CND 212: Digital Testing and Verification

FINAL PROJECT

UVM Verification for ALU8051

Section #: 19

Submitted by:

Student Name	ID
Aya Reda Osman	V23010305
Fatma Abdelmonsef	V23010243
Hadeer Khaled	V23010623
Hayat Gamal Abdel Hady	V23010201

Submitted to:

- **Dr: Ahmed Saeed**
- **TA: Randa Aboudeif**

1. Introduction:

- This project focuses on developing a Universal Verification Methodology (UVM) environment for verifying the Arithmetic Logic Unit (ALU) of the 8051 microcontroller. The 8051 ALU is a crucial component responsible for performing arithmetic and logic operations, essential for the microcontroller's functionality. By leveraging UVM, a standardized verification framework, we aim to create a robust, reusable, and scalable verification environment that ensures the ALU operates correctly under various conditions. This project will enhance the reliability of the ALU8051, contributing to the overall performance and dependability of systems utilizing this microcontroller.

2. Test Plan:

- we have made a test plan for all operations at which we dedicated a test for each problem then we randomize the inputs in a random way to improve the total coverage:

Tested Features	OP_CODE	Src1	Src2	Src3	SrcCy	SrcAc	bit_in	des1	des2	des_acc	sub_result	desCy	desAc	desOv
NOP	4'b0000	8'b11001001	8'b10010010	8'b10110010	0	1	0	8'b11001001	8'b10110011	8'b01011011	8'b00110111	1	0	1
ADD	4'b0001	8'b11010001	8'b01110010	8'b10010000	0	1	1	8'b00000000	8'b00000000	8'b01011111	8'b01011111	0	1	1
SUB	4'b0010	8'b11110110	8'b11011011	8'b11000011	1	1	0	8'b11110110	8'b11100010	8'b00000010	8'b00011010	0	0	1
MUL	4'b0011	8'b10001111	8'b10111111	8'b10101001	1	0	0	8'b10001111	8'b00000000	8'b00010111	8'b10100100	0	0	0
DIV	4'b0100	8'b10110001	8'b00001101	8'b10001010	0	1	1	8'b10110001	8'b00000000	8'b00010111	8'b10100100	1	0	0
DA (Decimal Adjustment)	4'b0101	8'b11111101	8'b00011001	8'b00110011	1	0	0	8'b00000010	8'b00000000	8'b00000010	8'b11100011	0	0	0
NOT	4'b0110	8'b01011110	8'b00000100	8'b11101011	1	1	0	8'b00000100	8'b00000000	8'b00000100	8'b01011001	0	0	0
AND	4'b0111	8'b00000010	8'b10001011	8'b01011111	0	0	0	8'b10001001	8'b00000000	8'b10001001	8'b01110111	0	0	0
XOR	4'b1000	8'b11001000	8'b10001110	8'b01101011	1	0	0	8'b11001110	8'b00000000	8'b11001110	8'b00111001	1	0	0
OR	4'b1001	8'b00001101	8'b01111101	8'b00001001	1	0	0	8'b00001101	8'b00000000	8'b00011010	8'b10001111	1	0	0
RL (Rotate Left)	4'b1010	8'b01010110	8'b11111101	8'b00100100	1	0	1	8'b01010110	8'b01100101	8'b10101101	8'b01011000	0	0	0
RLC (Rotate Left with Carry)	4'b1011	8'b10011000	8'b01101111	8'b01101000	0	0	1	8'b10011000	8'b00000000	8'b01001100	8'b11010001	0	0	0
RR (Rotate Right)	4'b1100	8'b10011110	8'b01101101	8'b01011101	0	1	1	8'b10011110	8'b00000000	8'b01001111	8'b00110001	0	0	0
RRC (Rotate Right with Carry)	4'b1101	8'b10111100	8'b11100011	8'b11000100	1	1	1	8'b10111011	8'b11100011	8'b10111011	8'b11011000	0	0	0
INC	4'b1110	8'b10111101	8'b11001011	8'b00100001	0	0	0	8'b10111011	8'b11001101	8'b10111011	8'b11110010	0	0	0
XCH (Exchange)	4'b1111	8'b11111001	8'b11000010	8'b11001101	0	0	1	8'b11111001	8'b11000010	8'b11111001	8'b00110111	0	0	0

3. UVM Testbench :

We are now introducing our UVM testbench for the ALU8051. A comprehensive UVM testbench consists of several key components: the test, sequence, environment, agent, sequencer, driver, monitor, interface, scoreboard, and coverage. These components work together to create a robust and efficient verification environment. In the following sections, we will introduce each of these components and explain their roles and functionalities within our UVM testbench for the ALU8051.

3.1. Test bench:

```
`include "uvm_macros.svh"
import uvm_pkg::*;
`include "interface.sv"
`include "test.sv"

module tbench_top;
  //clock and reset signal declaration
  bit clk;
  bit rst;

  //clock generation
  always #5 clk = ~clk;
  //reset Generation
  initial begin
    rst = 1;
    #5 rst =0;
  end

  alu_inf intf(clk,rst);
```

```
    oc8051_alu DUT (
      .clk(intf.clk),
      .rst(intf.rst),
      .src1(intf.src1),
      .src2(intf.src2),
      .src3(intf.src3),
      .srcCy(intf.srcCy),
      .srcAc(intf.srcAc),
      .bit_in(intf.bit_in),
      .des1(intf.des1),
      .des2(intf.des2),
      .des_acc(intf.des_acc),
      .desCy(intf.desCy),
      .desAc(intf.desAc),
      .desOv(intf.desOv),
      .sub_result(intf.sub_result),
      .op_code(intf.op_code));

  initial begin
    uvm_config_db#(virtual alu_inf)::set(null,"*","vif",intf);
    $dumpfile("dump.vcd");
    $dumpvars;
  end
```

```
  initial begin
    run_test("test");
  end
endmodule
```

3.2. Test:

```
`include "environment.sv"

class test extends uvm_test;
  `uvm_component_utils(test)

  alu_sequence seq;
  environment ENV;

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

  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);

    seq=alu_sequence::type_id::create("seq");
    ENV=environment::type_id::create("ENV",this);
  endfunction : build_phase

    task run_phase(uvm_phase phase);
      phase.raise_objection(this);

      seq.start(ENV.agnt.alu_sqncr);

      phase.drop_objection(this);
    endtask:run_phase

endclass:test
```

3.3. Sequence:

```
class alu_sequence extends uvm_sequence#(seq_item);
  `uvm_object_utils(alu_sequence)


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

  virtual task body();
    repeat(2000)begin
    req=seq_item::type_id::create("req");
      start_item(req);
      assert(req.randomize());
    finish_item(req);
    end
  endtask

endclass
```

3.4. Interface:

```
interface alu_inf( input logic clk, rst);

   logic [7:0] src1;
   logic [7:0] src2;
   logic [7:0] src3;
   logic       srcCy;
   logic       srcAc;
   logic       bit_in;
   logic [3:0] op_code;
   logic       desCy;
   logic       desAc;
   logic       desOv;
   logic [7:0] des1;
   logic [7:0] des2;
   logic [7:0] des_acc;
   logic [7:0] sub_result;
```

```
 clocking driver_cb @(posedge clk);
   default input #1 output #1;
output  src1;
output  src2;
output  src3;
output  srcCy;
output  srcAc;
output  bit_in;
output  op_code;
input   des1;
input   des2;
input   des_acc;
input   sub_result;
input   desCy;
input   desAc;
input   desOv;
endclocking
```

```
 clocking monitor_cb @(negedge clk);
   default input #1 output #1;
input  src1;
input  src2;
input  src3;
input  srcCy;
input  srcAc;
input  bit_in;
input  op_code;
input  des1;
input  des2;
input  des_acc;
input  sub_result;
input  desCy;
input  desAc;
input  desOv;
endclocking

modport DRIVER (clocking driver_cb, input clk,rst);
modport MONITOR (clocking monitor_cb, input clk,rst);

   endinterface
```

3.5. Sequence Itmes:

```
class seq_item extends uvm_sequence_item;
    rand  bit [7:0] src1;
    rand  bit [7:0] src2;
    rand  bit [7:0] src3;
    randc  bit      srcCy;
    randc  bit      srcAc;
    randc bit [3:0] op_code;
          bit       bit_in;
          bit       desCy;
          bit       desAc;
          bit       desOv;
          bit [7:0] des1;
          bit [7:0] des2;
          bit [7:0] des_acc;
          bit [7:0] sub_result;
```

```
  `uvm_object_utils_begin(seq_item)
  `uvm_field_int(src1,UVM_ALL_ON)
  `uvm_field_int(src2,UVM_ALL_ON)
  `uvm_field_int(src3,UVM_ALL_ON)
  `uvm_field_int(srcCy,UVM_ALL_ON)
  `uvm_field_int(srcAc,UVM_ALL_ON)
  `uvm_field_int(des1,UVM_ALL_ON)
  `uvm_field_int(des2,UVM_ALL_ON)
  `uvm_field_int(des_acc,UVM_ALL_ON)
  `uvm_field_int(desCy,UVM_ALL_ON)
  `uvm_field_int(desAc,UVM_ALL_ON)
  `uvm_field_int(desOv,UVM_ALL_ON)
  `uvm_field_int(bit_in,UVM_ALL_ON)
  `uvm_field_int(op_code,UVM_ALL_ON)
  `uvm_field_int(sub_result,UVM_ALL_ON)
  `uvm_object_utils_end

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

endclass
```

3.6. Sequencer:

```
class sequencer extends uvm_sequencer#(seq_item);
  `uvm_component_utils(sequencer)

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction
endclass
```

3.7. Driver:

```
`define DRIV_IF vif.DRIVER.driver_cb

class driver extends uvm_driver#(seq_item);
  `uvm_component_utils(driver)

  virtual alu_inf vif;

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction

  //////////////   BUILD PHASE   ////////////////////////////
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db#(virtual alu_inf)::get(this,"", "vif",vif))
      `uvm_fatal("No vif found", {"virtual interface must be set for: ",get_full_name(),".vif"});
  endfunction: build_phase

  //////////////   RUN PHASE   ////////////////////////////
  virtual task run_phase(uvm_phase phase);
    forever begin
      seq_item_port.get_next_item(req);
      drive();
      seq_item_port.item_done();
    end
  endtask: run_phase
```

```
  //////////////   DRIVE() TASK   ////////////////////////////

  virtual task drive();
    @(negedge vif.clk);
    vif.src1<=req.src1;
    vif.src2<=req.src2;
    vif.src3<=req.src3;
    vif.srcAc<=req.srcAc;
    vif.srcCy<=req.srcCy;
    vif.bit_in<=req.bit_in;
    vif.op_code<=req.op_code;

    @(negedge vif.DRIVER.clk);
    req.des1<=vif.des1;
    req.des2<=vif.des2;
    req.des_acc<=vif.des_acc;
    req.sub_result<=vif.sub_result;
    req.desCy<=vif.desCy;
    req.desAc<=vif.desAc;
    req.desOv<=vif.desOv;
  endtask: drive

endclass: driver
```

3.8. Monitor:

```
`define MON_IF vif.monitor_cb

class monitor extends uvm_monitor;
  `uvm_component_utils(monitor)

  virtual alu_inf vif;
  uvm_analysis_port#(seq_item) item_collected_port;
  seq_item seq_collected;

  function new(string name, uvm_component parent);
    super.new(name,parent);
    item_collected_port=new("item_collected_port",this);
    seq_collected=new();
  endfunction: new

    //////////////   BUILD PHASE   ////////////////////////////
    function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db#(virtual alu_inf)::get(this,"", "vif",vif))
      `uvm_fatal("No vif found", {"virtual interface must be set for: ",get_full_name(),".vif"});
  endfunction: build_phase
```

```
  //////////////   RUN PHASE   ////////////////////////////
    virtual task run_phase(uvm_phase phase);
       seq_item item_collected;

      forever begin
        @(negedge vif.clk);
        seq_collected.src1=vif.src1;
        seq_collected.src2=vif.src2;
        seq_collected.src3=vif.src3;
        seq_collected.srcAc=vif.srcAc;
        seq_collected.srcCy=vif.srcCy;
        seq_collected.bit_in=vif.bit_in;
        seq_collected.op_code=vif.op_code;

        seq_collected.des1=vif.des1;
        seq_collected.des2=vif.des2;
        seq_collected.des_acc=vif.des_acc;
        seq_collected.sub_result=vif.sub_result;
        seq_collected.desCy=vif.desCy;
        seq_collected.desAc=vif.desAc;
        seq_collected.desOv=vif.desOv;


        $cast(item_collected, seq_collected.clone());
        item_collected_port.write(item_collected);

      end
  endtask: run_phase

endclass: monitor
```

3.9. Agent:

```
`include "sequence_items.sv"
`include "sequencer.sv"
`include "sequence.sv"
`include "Driver.sv"
`include "monitor.sv"

class agent extends uvm_agent;
  `uvm_component_utils(agent)

  sequencer alu_sqncr;
  driver alu_driv;
  monitor alu_mon;

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction: new

    //////////////   BUILD PHASE   ////////////////////////////
    virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
        //ACTIVE
      alu_sqncr=sequencer::type_id::create("alu_sqncr",this);
      alu_driv=driver::type_id::create("alu_driver",this);
      alu_mon=monitor::type_id::create("alu_mon",this);
    endfunction:build_phase

    //////////////   CONNECT PHASE   ////////////////////////////
  virtual function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    alu_driv.seq_item_port.connect(alu_sqncr.seq_item_export);
  endfunction: connect_phase

endclass: agent
```

3.10. Environment:

```
`include "agent.sv"
`include "scoreboard.sv"
`include "coverage.sv"

class environment extends uvm_env;
  `uvm_component_utils(environment)
   agent agnt;
   scoreboard scb;
   coverage cov;

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction:new

  //////////////////BUILD PHASE//////////////////////
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);

    agnt=agent::type_id::create("agnt",this);
    scb=scoreboard::type_id::create("scb",this);
    cov=coverage::type_id::create("cov",this);
  endfunction: build_phase

  //////////////CONNECT PHASE////////////////////////
  function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    agnt.alu_mon.item_collected_port.connect(scb.sb_fifo.analysis_export);
    agnt.alu_mon.item_collected_port.connect(cov.analysis_export);

  endfunction: connect_phase

endclass:environment
```

3.11. Scoreboard:

```
class scoreboard extends uvm_scoreboard;
 `uvm_component_utils(scoreboard)

   uvm_analysis_export   #(seq_item) sb_export;
   uvm_tlm_analysis_fifo #(seq_item) sb_fifo;
   seq_item  seq_item_sb;

   logic      desCy_ref, desAc_ref, desOv_ref;
   logic [7:0] des1_ref, des2_ref, des_acc_ref, sub_result_ref;

   //add :internal variables
   bit [4:0] add1, add2, add3, add4;
   bit [3:0] add5, add6, add7, add8;
   bit [1:0] add9, adda, addb, addc;

   //sub :internal variables
   bit [4:0] sub1, sub2, sub3, sub4;
   bit [3:0] sub5, sub6, sub7, sub8;
   bit [1:0] sub9, suba, subb, subc;
   bit [7:0] sub_result;

   //da :internal variables
   bit da_tmp, da_tmp1;

   // inc :internal variables
   bit [15:0] inc, dec;
   bit enable_mul;
   bit enable_div;
   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);
    ref_model(seq_item_sb);

    if (seq_item_sb.desCy!= desCy_ref|| seq_item_sb.desAc != desAc_ref || seq_item_sb.desOv != desOv_ref || seq_item_sb.des1!= des1_ref
     ||seq_item_sb.des2 != des2_ref || seq_item_sb.des_acc != des_acc_ref ||seq_item_sb.sub_result != sub_result_ref) begin
          `uvm_error("run_phase", $sformatf ("comparsion failed, transaction received by DUT :%s, while ****,
                     desCy_ref=0b%0b,desAc_ref=0b%0b,desOv_ref=0b%0b,des1_ref=0b%0b,des2_ref=0b%0b,
                     des_acc_ref=0b%0b,sub_result_ref=0b%0b",seq_item_sb.convert2string(),desCy_ref,desAc_ref,desOv_ref,des1_ref,
                     des2_ref,des_acc_ref,sub_result_ref));
        error_count++;
    end
    else begin
        `uvm_info("run_phase", $sformatf ("correct ALSU_out: %s",seq_item_sb.convert2string()),UVM_HIGH);
         correct_count++;
    end
 end
 endtask
```

```
task ref_model (seq_item seq_item_chk);
    add1 = {1'b0,seq_item_sb.src1[3:0]};
    add2 = {1'b0,seq_item_sb.src2[3:0]};
    add3 = {3'b000,seq_item_sb.srcCy};
    add4 = add1+add2+add3;

    add5 = {1'b0,seq_item_sb.src1[6:4]};
    add6 = {1'b0,seq_item_sb.src2[6:4]};
    add7 = {1'b0,1'b0,1'b0,add4[4]};
    add8 = add5+add6+add7;

    add9 = {1'b0,seq_item_sb.src1[7]};
    adda = {1'b0,seq_item_sb.src2[7]};
    addb = {1'b0,add8[3]};
    addc = add9+adda+addb;

    sub1 = {1'b1,seq_item_sb.src1[3:0]};
    sub2 = {1'b0,seq_item_sb.src2[3:0]};
    sub3 = {1'b0,1'b0,1'b0,seq_item_sb.srcCy};
    sub4 = sub1-sub2-sub3;

    sub5 = {1'b1,seq_item_sb.src1[6:4]};
    sub6 = {1'b0,seq_item_sb.src2[6:4]};
    sub7 = {1'b0,1'b0,1'b0, !sub4[4]};
    sub8 = sub5-sub6-sub7;

    sub9 = {1'b1,seq_item_sb.src1[7]};
    suba = {1'b0,seq_item_sb.src2[7]};
    subb = {1'b0,!sub8[3]};
    subc = sub9-suba-subb;

    sub_result = {subc[0],sub8[2:0],sub4[3:0]};

    inc = {seq_item_sb.src2, seq_item_sb.src1} + {15'h0, 1'b1};
    dec = {seq_item_sb.src2, seq_item_sb.src1} - {15'h0, 1'b1};
```

```
case (seq_item_sb.op_code)
    //operation add
    `OC8051_ALU_ADD: begin
    des_acc_ref = {addc[0],add8[2:0],add4[3:0]};
    des1_ref = seq_item_sb.src1;
    des2_ref = seq_item_sb.src3+ {7'b0, addc[1]};
    desCy_ref = addc[1];
    desAc_ref = add4[4];
    desOv_ref = addc[1] ^ add8[3];
    enable_mul = 1'b0;
    enable_div = 1'b0;
    end
    //operation subtract
    `OC8051_ALU_SUB: begin
    des_acc_ref = sub_result_ref;
    des1_ref = 8'h00;
    des2_ref = 8'h00;
    desCy_ref = !subc[1];
    desAc_ref = !sub4[4];
    desOv_ref = !subc[1] ^ !sub8[3];
    enable_mul = 1'b0;
    enable_div = 1'b0;
    end
    //operation decimal adjustment
    `OC8051_ALU_DA: begin
    if (seq_item_sb.srcAc==1'b1 | seq_item_sb.src1[3:0]>4'b1001)
        {da_tmp, des_acc_ref[3:0]} = {1'b0, seq_item_sb.src1[3:0]}+ 5'b00110;
    else
        {da_tmp, des_acc_ref[3:0]} = {1'b0, seq_item_sb.src1[3:0]};

    if (seq_item_sb.srcCy | da_tmp | seq_item_sb.src1[7:4]>4'b1001)
        {da_tmp1, des_acc_ref[7:4]} = {seq_item_sb.srcCy, seq_item_sb.src1[7:4]}+ 5'b00110 + {4'b0, da_tmp};
    else
        {da_tmp1, des_acc_ref[7:4]} = {seq_item_sb.srcCy, seq_item_sb.src1[7:4]} + {4'b0, da_tmp};

    desCy_ref = da_tmp | da_tmp1;
```

```
desCy_ref = da_tmp | da_tmp1;
des1_ref = seq_item_sb.src1;
des2_ref = 8'h00;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
//operation not
// bit operation not
`OC8051_ALU_NOT: begin
des_acc_ref = ~seq_item_sb.src1;
des1_ref = ~seq_item_sb.src1;
des2_ref = 8'h00;
desCy_ref = !seq_item_sb.srcCy;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
//operation and
//bit operation and
`OC8051_ALU_AND: begin
des_acc_ref = seq_item_sb.src1 & seq_item_sb.src2;
des1_ref = seq_item_sb.src1 & seq_item_sb.src2;
des2_ref = 8'h00;
desCy_ref = seq_item_sb.srcCy & seq_item_sb.bit_in;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
```

```
`OC8051_ALU_XOR: begin
des_acc_ref = seq_item_sb.src1 ^ seq_item_sb.src2;
des1_ref = seq_item_sb.src1 ^ seq_item_sb.src2;
des2_ref = 8'h00;
desCy_ref = seq_item_sb.srcCy ^ seq_item_sb.bit_in;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
//operation or
// bit operation or
`OC8051_ALU_OR: begin
des_acc_ref = seq_item_sb.src1 | seq_item_sb.src2;
des1_ref = seq_item_sb.src1 | seq_item_sb.src2;
des2_ref = 8'h00;
desCy_ref = seq_item_sb.srcCy | seq_item_sb.bit_in;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
//operation rotate left
// bit operation cy= cy or (not ram)
`OC8051_ALU_RL: begin
des_acc_ref = {seq_item_sb.src1[6:0], seq_item_sb.src1[7]};
des1_ref = seq_item_sb.src1 ;
des2_ref = 8'h00;
desCy_ref = seq_item_sb.srcCy | !seq_item_sb.bit_in;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
```

```
`OC8051_ALU_RLC: begin
des_acc_ref = {seq_item_sb.src1[6:0], seq_item_sb.srcCy};
des1_ref = seq_item_sb.src1 ;
des2_ref = {seq_item_sb.src1[3:0], seq_item_sb.src1[7:4]};
desCy_ref = seq_item_sb.src1[7];
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
//operation rotate right
`OC8051_ALU_RR: begin
des_acc_ref = {seq_item_sb.src1[0], seq_item_sb.src1[7:1]};
des1_ref = seq_item_sb.src1 ;
des2_ref = 8'h00;
desCy_ref = seq_item_sb.srcCy & !seq_item_sb.bit_in;
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
//operation rotate right with carry
`OC8051_ALU_RRC: begin
des_acc_ref = {seq_item_sb.srcCy, seq_item_sb.src1[7:1]};
des1_ref = seq_item_sb.src1 ;
des2_ref = 8'h00;
desCy_ref = seq_item_sb.src1[0];
desAc_ref = 1'b0;
desOv_ref = 1'b0;
enable_mul = 1'b0;
enable_div = 1'b0;
end
```

```
        `OC8051_ALU_INC: begin
        if (seq_item_sb.srcCy) begin
            des_acc_ref = dec[7:0];
            des1_ref = dec[7:0];
            des2_ref = dec[15:8];
        end
        else begin
            des_acc_ref = inc[7:0];
            des1_ref = inc[7:0];
            des2_ref = inc[15:8];
        end
        desCy_ref = 1'b0;
        desAc_ref = 1'b0;
        desOv_ref = 1'b0;
        enable_mul = 1'b0;
        enable_div = 1'b0;
        end
        //operation exchange
        //if carry = 0 exchange low order digit
        `OC8051_ALU_XCH: begin
        if (seq_item_sb.srcCy)
        begin
            des_acc_ref = seq_item_sb.src2;
            des1_ref = seq_item_sb.src2;
            des2_ref = seq_item_sb.src1;
        end
        else begin
            des_acc_ref = {seq_item_sb.src1[7:4],seq_item_sb.src2[3:0]};
            des1_ref = {seq_item_sb.src1[7:4],seq_item_sb.src2[3:0]};
            des2_ref = {seq_item_sb.src2[7:4],seq_item_sb.src1[3:0]};
        end
```

```
        desCy_ref = 1'b0;
        desAc_ref = 1'b0;
        desOv_ref = 1'b0;
        enable_mul = 1'b0;
        enable_div = 1'b0;
        end
        `OC8051_ALU_NOP: begin
        des_acc_ref = seq_item_sb.src1;
        des1_ref = seq_item_sb.src1;
        des2_ref = seq_item_sb.src2;
        desCy_ref = seq_item_sb.srcCy;
        desAc_ref = seq_item_sb.srcAc;
        desOv_ref = 1'b0;
        enable_mul = 1'b0;
        enable_div = 1'b0;
  end
endcase

  endtask


  function void report_phase(uvm_phase  phase);
      super.report_phase(phase);
      `uvm_info("report_phase", $sformatf("total succesful transactions: %0d",correct_count),UVM_MEDIUM);
      `uvm_info("report_phase", $sformatf("total failled transactions: %0d",error_count),UVM_MEDIUM);
  endfunction

endclass
```

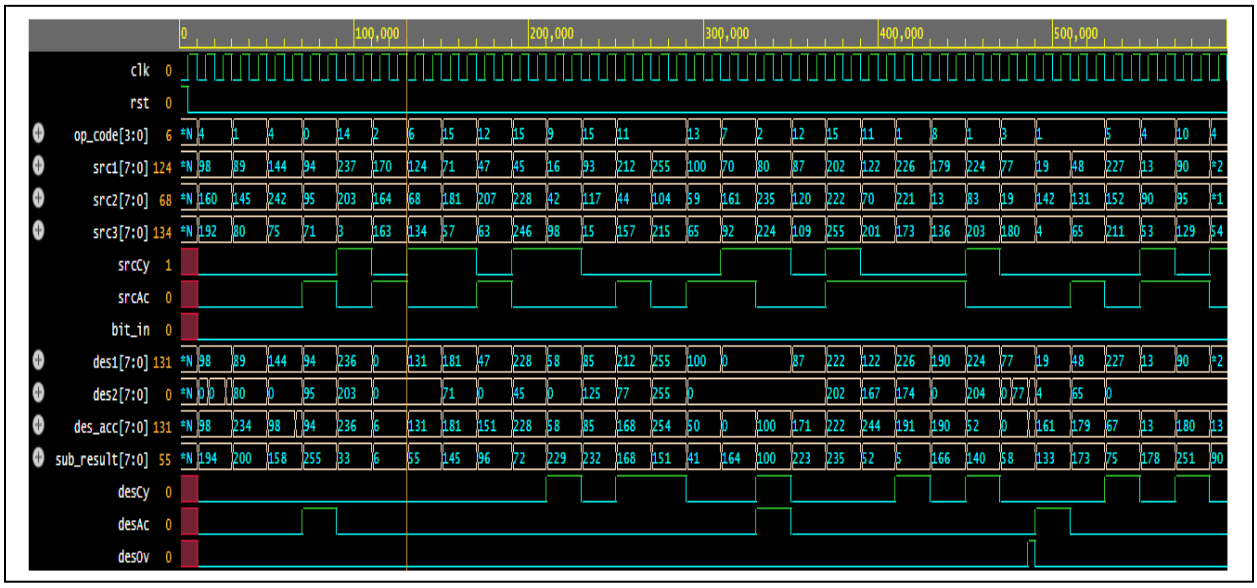
3.12. Coverage:

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

class coverage extends uvm_subscriber #(seq_item);
`uvm_component_utils (ALU_coverage)
      uvm_analysis_export #(ALU_seq_item) cov_export;
      uvm_tlm_analysis_fifo #(ALU_seq_item) cov_fifo;
      ALU_seq_item seq_item_cov;


        covergroup cvr_grp  ;
          rst_g:         coverpoint seq_item_cov.rst;
          srcCy_g:       coverpoint seq_item_cov.srcCy;
          srcAc_g:       coverpoint seq_item_cov.srcAc;
          bit_in_g:      coverpoint seq_item_cov.bit_in;
          op_code_g:     coverpoint seq_item_cov. op_code{ illegal_bins mul={4'b0011 }; illegal_bins div={4'b0100};}
          src1_g:        coverpoint seq_item_cov.src1;
          src2_g:        coverpoint seq_item_cov.src2;
          src3_g:        coverpoint seq_item_cov.src3;
          desCy_g:       coverpoint seq_item_cov.desCy;
          desAc_g:       coverpoint seq_item_cov.desAc;
          desOv_g:       coverpoint seq_item_cov.desOv;
          des1_g:        coverpoint seq_item_cov.des1;
          des2_g:        coverpoint seq_item_cov.des2;
          des_acc_g:     coverpoint seq_item_cov.des_acc;
          sub_result_g:  coverpoint seq_item_cov.sub_result;
        endgroup;

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

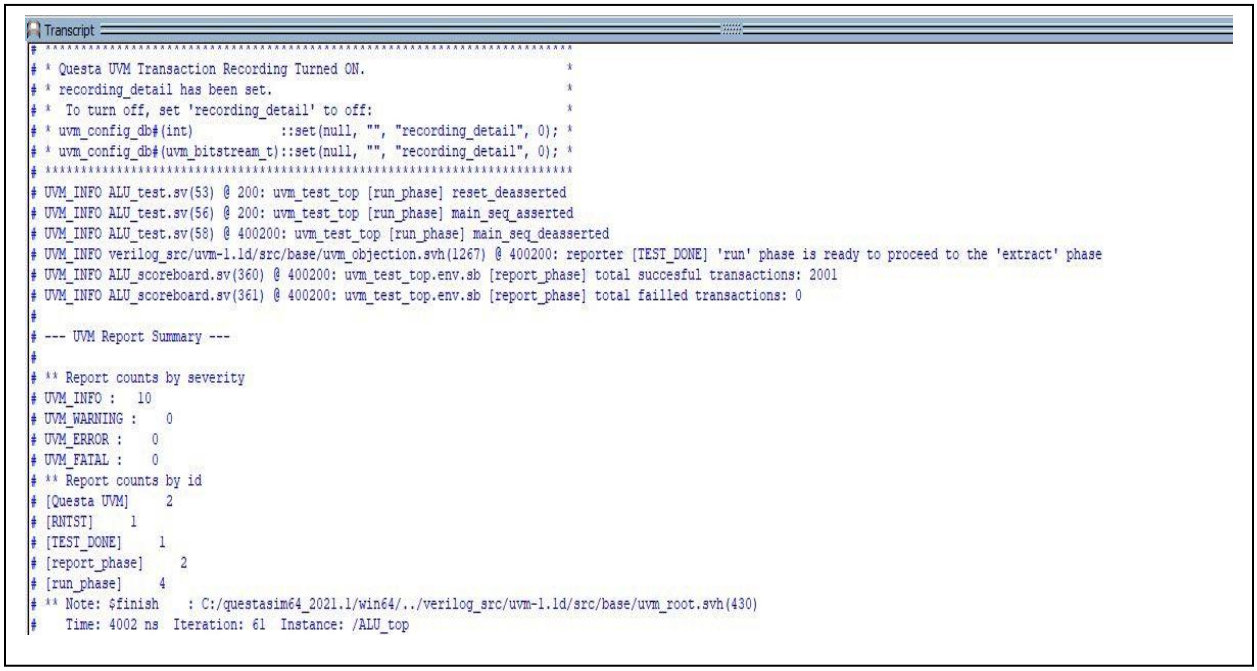
```
    function void build_phase (uvm_phase phase);
      super.build_phase (phase);
      cov_export = new("cov_export", this);
      cov_fifo = new("cov_fifo", this);
    endfunction

     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);
            cvr_grp.sample();
        end
     endtask

  endclass
```

4. Simulation Results:

4.1. Waveform:



4.2. UVM Report Summary:



```
Transcript
# ********************************************************************************
# * 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 ALU_test.sv(53) @ 200: uvm_test_top [run_phase] reset_deasserted
# UVM_INFO ALU_test.sv(56) @ 200: uvm_test_top [run_phase] main_seq_asserted
# UVM_INFO ALU_test.sv(58) @ 400200: uvm_test_top [run_phase] main_seq_deasserted
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 400200: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO ALU_scoreboard.sv(360) @ 400200: uvm_test_top.env.sb [report_phase] total succesful transactions: 2001
# UVM_INFO ALU_scoreboard.sv(361) @ 400200: uvm_test_top.env.sb [report_phase] total failled transactions: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   10
# 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]     4
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 4002 ns  Iteration: 61  Instance: /ALU_top
```

4.3. Coverage Results:

Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/ALU_coverage_pk...		100.00%							
TYPE cvr_grp		100.00%	100	100.00...	✓		auto(1)		
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				
CVP cvr_gr...		100.00%	100	100.00...	✓				

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

-----------------------------------------------------------------------------------------------------------
 TYPE /ALU_coverage_pkg/ALU_coverage/cvr_grp        100.00%       100          -    Covered
    covered/total bins:                                476        476          -
    missing/total bins:                                  0        476          -
    % Hit:                                         100.00%        100          -
    Coverpoint rst_g                               100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                   1959          1          -    Covered
        bin auto[1]                                     42          1          -    Covered
    Coverpoint srcCy_g                             100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                   1008          1          -    Covered
        bin auto[1]                                    993          1          -    Covered
    Coverpoint srcAc_g                             100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                    941          1          -    Covered
        bin auto[1]                                   1060          1          -    Covered
    Coverpoint bit_in_g                            100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                   1025          1          -    Covered
        bin auto[1]                                    976          1          -    Covered
    Coverpoint op_code_g                           100.00%        100          -    Covered
        covered/total bins:                             14         14          -
        missing/total bins:                              0         14          -
        % Hit:                                     100.00%        100          -
```

bin auto[12:15]	30	1	-	Covered
bin auto[16:19]	30	1	-	Covered
bin auto[20:23]	25	1	-	Covered
bin auto[24:27]	25	1	-	Covered
bin auto[28:31]	25	1	-	Covered
bin auto[32:35]	29	1	-	Covered
bin auto[36:39]	34	1	-	Covered
bin auto[40:43]	38	1	-	Covered
bin auto[44:47]	26	1	-	Covered
bin auto[48:51]	42	1	-	Covered
bin auto[52:55]	36	1	-	Covered
bin auto[56:59]	32	1	-	Covered
bin auto[60:63]	36	1	-	Covered
bin auto[64:67]	39	1	-	Covered
bin auto[68:71]	34	1	-	Covered
bin auto[72:75]	30	1	-	Covered
bin auto[76:79]	27	1	-	Covered
bin auto[80:83]	31	1	-	Covered
bin auto[84:87]	27	1	-	Covered
bin auto[88:91]	37	1	-	Covered
bin auto[92:95]	33	1	-	Covered
bin auto[96:99]	31	1	-	Covered
bin auto[100:103]	34	1	-	Covered
bin auto[104:107]	33	1	-	Covered
bin auto[108:111]	27	1	-	Covered
bin auto[112:115]	28	1	-	Covered
bin auto[116:119]	28	1	-	Covered
bin auto[120:123]	37	1	-	Covered
bin auto[124:127]	27	1	-	Covered
bin auto[128:131]	33	1	-	Covered
bin auto[132:135]	24	1	-	Covered
bin auto[136:139]	29	1	-	Covered
bin auto[140:143]	29	1	-	Covered
bin auto[144:147]	39	1	-	Covered
bin auto[148:151]	33	1	-	Covered
bin auto[152:155]	33	1	-	Covered
bin auto[156:159]	28	1	-	Covered

bin auto[132:135]	24	1	-	Covered
bin auto[136:139]	29	1	-	Covered
bin auto[140:143]	29	1	-	Covered
bin auto[144:147]	39	1	-	Covered
bin auto[148:151]	33	1	-	Covered
bin auto[152:155]	33	1	-	Covered
bin auto[156:159]	28	1	-	Covered
bin auto[160:163]	25	1	-	Covered
bin auto[164:167]	32	1	-	Covered
bin auto[168:171]	37	1	-	Covered
bin auto[172:175]	38	1	-	Covered
bin auto[176:179]	26	1	-	Covered
bin auto[180:183]	29	1	-	Covered
bin auto[184:187]	41	1	-	Covered
bin auto[188:191]	36	1	-	Covered
bin auto[192:195]	30	1	-	Covered
bin auto[196:199]	32	1	-	Covered
bin auto[200:203]	27	1	-	Covered
bin auto[204:207]	43	1	-	Covered
bin auto[208:211]	31	1	-	Covered
bin auto[212:215]	35	1	-	Covered
bin auto[216:219]	33	1	-	Covered
bin auto[220:223]	44	1	-	Covered
bin auto[224:227]	23	1	-	Covered
bin auto[228:231]	25	1	-	Covered
bin auto[232:235]	27	1	-	Covered
bin auto[236:239]	30	1	-	Covered
bin auto[240:243]	31	1	-	Covered
bin auto[244:247]	32	1	-	Covered
bin auto[248:251]	30	1	-	Covered
bin auto[252:255]	22	1	-	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1