UVM Verification of ALU Design

Submitted to:

Dr. Ahmed Saeed

Eng. Randa Aboudeif

—

Fagr Ahmed Abd Elrazek “AUC section 12” ID:V23010586

Marwa Moaz “AUC section 14” ID:V23009897

Mohamed Yasser “AUC section 16” ID:V23010495

shurooq Mohammad Hassan Alshal “AUC section 12” ID:V23010529

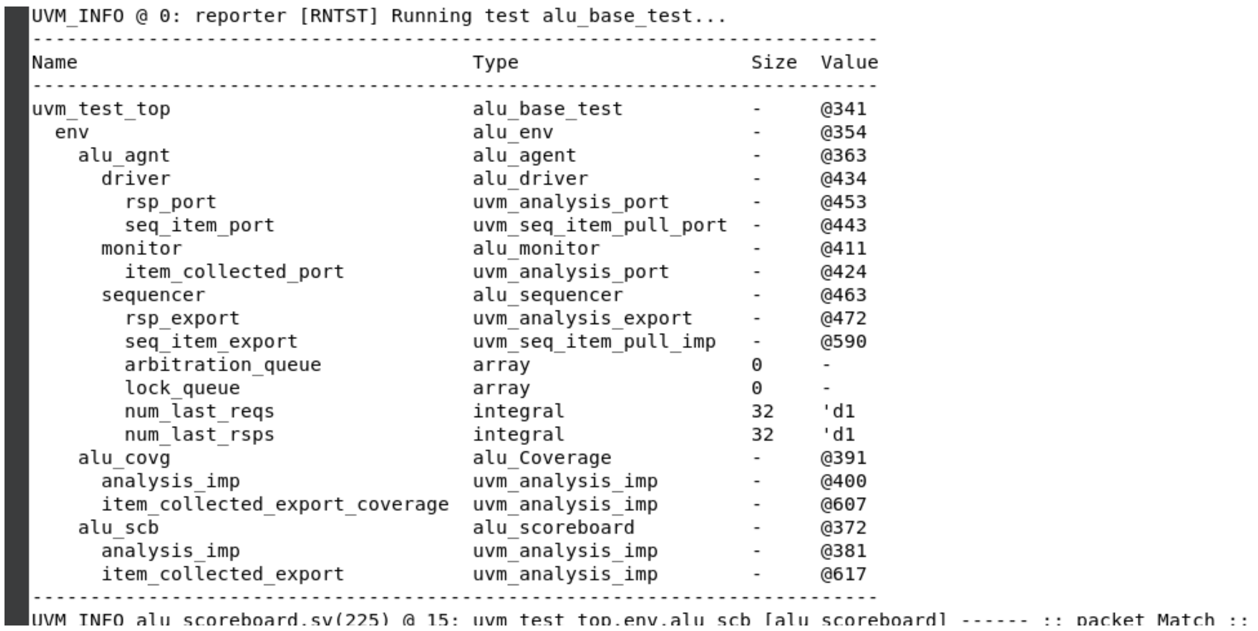
Overview

the complete verification flow to generate a working UVM testbench for an open-source ALU design available at: https://shorturl.at/lvwX6 using the UVM (Universal Verification Methodology) Framework.

Verification Plan

1. The Design Specs and testbench architecture:

   The design is an alu that performs multiplication and division in four clock cycles and the rest of the operations in one clock cycle. These commands include arithmetic commands such as addition and subtraction and increment and also logic commands such as AND, OR and NOT. In addition to NOP command. The inputs to the design are clk, rst,op_code “4 bits”, src1 “8 bits”, src2 “8 bits”, src3 “8 bits”, srcCy, srcAc, and bit_in. The outputs from the design are des1 “8 bits”, des2 “8 bits”, des_acc, desCy, desAc, desOv,and sub_result “8 bits”. The test bench architecture is as follows



```
UVM_INFO @ 0: reporter [RNTST] Running test alu_base_test...
-------------------------------------------------------------------------
Name                                  Type                    Size  Value
-------------------------------------------------------------------------
uvm_test_top                          alu_base_test           -     @341
  env                                 alu_env                 -     @354
    alu_agnt                          alu_agent               -     @363
      driver                          alu_driver              -     @434
        rsp_port                      uvm_analysis_port       -     @453
        seq_item_port                 uvm_seq_item_pull_port  -     @443
      monitor                         alu_monitor             -     @411
        item_collected_port           uvm_analysis_port       -     @424
      sequencer                       alu_sequencer           -     @463
        rsp_export                    uvm_analysis_export     -     @472
        seq_item_export               uvm_seq_item_pull_imp   -     @590
        arbitration_queue             array                   0     -
        lock_queue                    array                   0     -
        num_last_reqs                 integral                32    'd1
        num_last_rsps                 integral                32    'd1
    alu_covg                          alu_Coverage            -     @391
      analysis_imp                    uvm_analysis_imp        -     @400
      item_collected_export_coverage  uvm_analysis_imp        -     @607
    alu_scb                           alu_scoreboard          -     @372
      analysis_imp                    uvm_analysis_imp        -     @381
      item_collected_export           uvm_analysis_imp        -     @617
-------------------------------------------------------------------------
UVM_INFO alu_scoreboard.sv(225) @ 15: uvm_test_top.env.alu_scb [alu_scoreboard] ------ :: packet Match ::
```

2. DUT Interface:

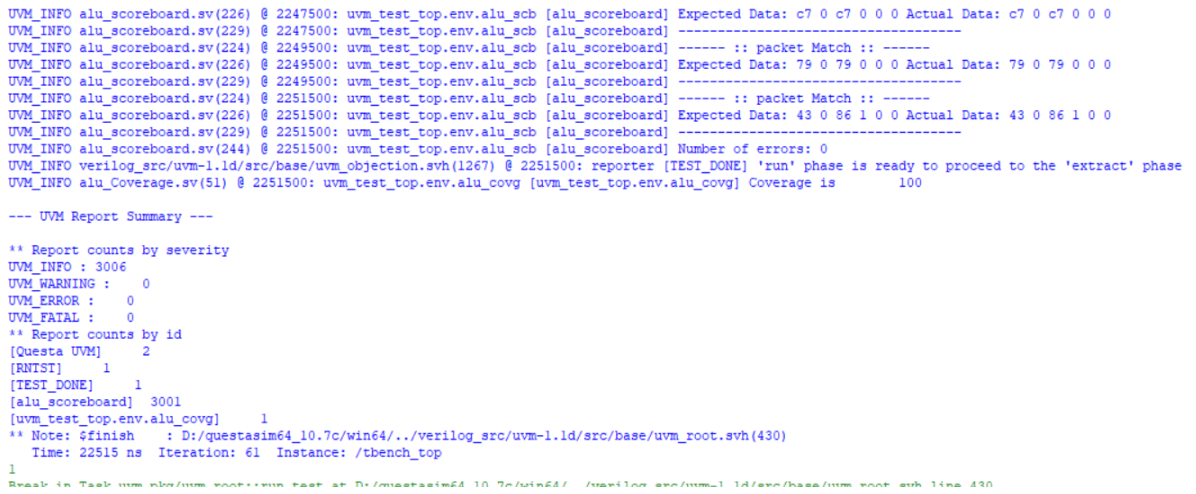
   The interface connects the DUT with both the driver and the monitor.

3. The testing Scenarios and functional cover points:

We randomized 1000 iterations for different opcodes to guarantee that all opcodes and input carries are covered.

Simulation Results

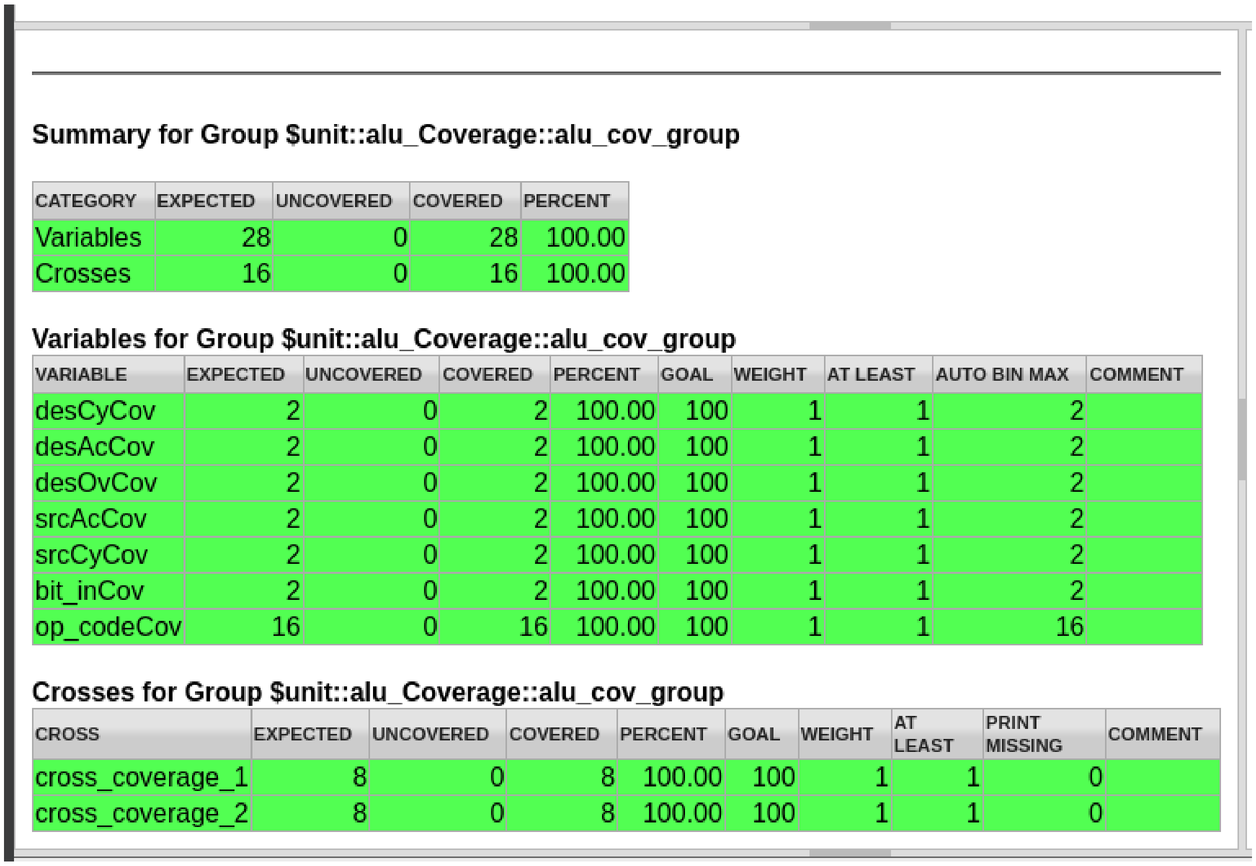
VCS and Questasim



```
UVM_INFO alu_scoreboard.sv(226) @ 2247500: uvm_test_top.env.alu_scb [alu_scoreboard] Expected Data: c7 0 c7 0 0 0 Actual Data: c7 0 c7 0 0 0
UVM_INFO alu_scoreboard.sv(229) @ 2247500: uvm_test_top.env.alu_scb [alu_scoreboard] ------------------------------------
UVM_INFO alu_scoreboard.sv(224) @ 2249500: uvm_test_top.env.alu_scb [alu_scoreboard] ------ :: packet Match :: ------
UVM_INFO alu_scoreboard.sv(226) @ 2249500: uvm_test_top.env.alu_scb [alu_scoreboard] Expected Data: 79 0 79 0 0 0 Actual Data: 79 0 79 0 0 0
UVM_INFO alu_scoreboard.sv(229) @ 2249500: uvm_test_top.env.alu_scb [alu_scoreboard] ------------------------------------
UVM_INFO alu_scoreboard.sv(224) @ 2251500: uvm_test_top.env.alu_scb [alu_scoreboard] ------ :: packet Match :: ------
UVM_INFO alu_scoreboard.sv(226) @ 2251500: uvm_test_top.env.alu_scb [alu_scoreboard] Expected Data: 43 0 86 1 0 0 Actual Data: 43 0 86 1 0 0
UVM_INFO alu_scoreboard.sv(229) @ 2251500: uvm_test_top.env.alu_scb [alu_scoreboard] ------------------------------------
UVM_INFO alu_scoreboard.sv(244) @ 2251500: uvm_test_top.env.alu_scb [alu_scoreboard] Number of errors: 0
UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 2251500: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO alu_Coverage.sv(51) @ 2251500: uvm_test_top.env.alu_covg [uvm_test_top.env.alu_covg] Coverage is        100

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO : 3006
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[Questa UVM]     2
[RNTST]     1
[TEST_DONE]     1
[alu_scoreboard]  3001
[uvm_test_top.env.alu_covg]     1
** Note: $finish    : D:/questasim64_10.7c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
   Time: 22515 ns  Iteration: 61  Instance: /tbench_top
1
Break in Task uvm_pkg/uvm_root::run_test at D:/questasim64_10.7c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```



Summary for Group $unit::alu_Coverage::alu_cov_group

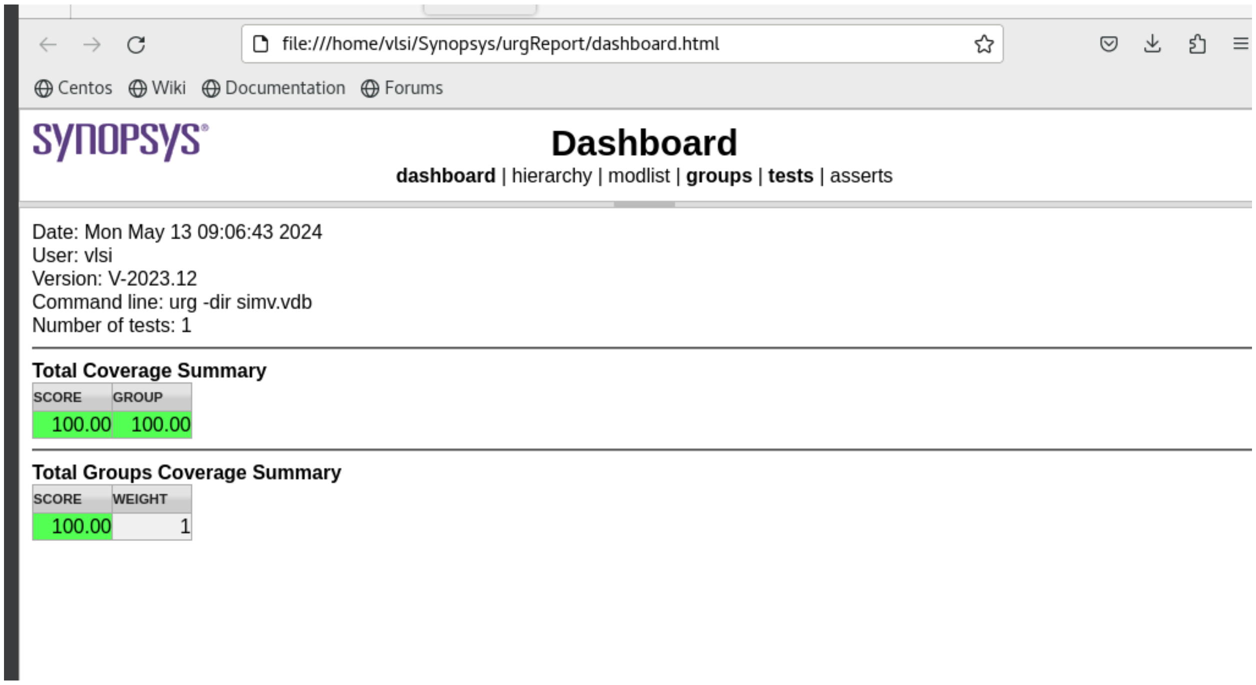
CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
Variables	28	0	28	100.00
Crosses	16	0	16	100.00

Variables for Group $unit::alu_Coverage::alu_cov_group

VARIABLE	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL	WEIGHT	AT LEAST	AUTO BIN MAX	COMMENT
desCyCov	2	0	2	100.00	100	1	1	2	
desAcCov	2	0	2	100.00	100	1	1	2	
desOvCov	2	0	2	100.00	100	1	1	2	
srcAcCov	2	0	2	100.00	100	1	1	2	
srcCyCov	2	0	2	100.00	100	1	1	2	
bit_inCov	2	0	2	100.00	100	1	1	2	
op_codeCov	16	0	16	100.00	100	1	1	16	

Crosses for Group $unit::alu_Coverage::alu_cov_group

CROSS	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL	WEIGHT	AT LEAST	PRINT MISSING	COMMENT
cross_coverage_1	8	0	8	100.00	100	1	1	0	
cross_coverage_2	8	0	8	100.00	100	1	1	0	



Dashboard

dashboard | hierarchy | modlist | **groups** | **tests** | asserts

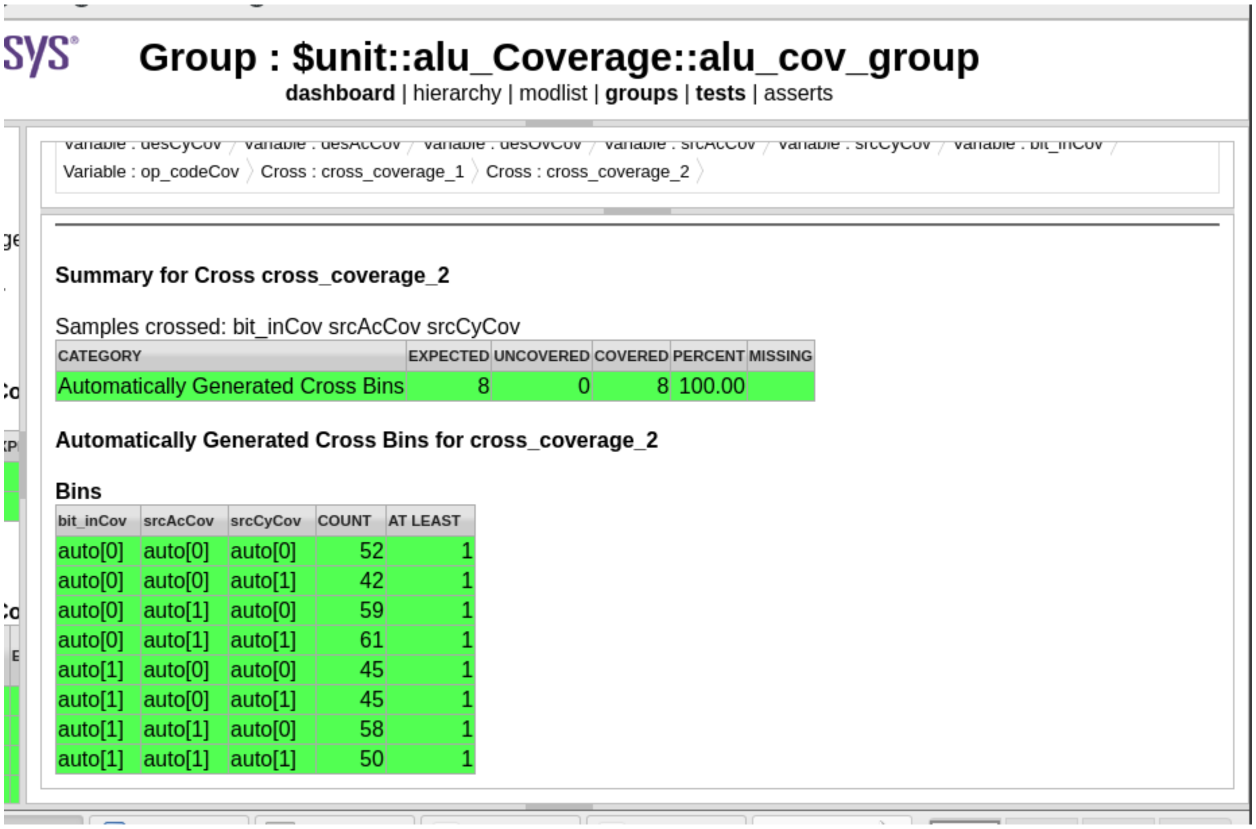
Date: Mon May 13 09:06:43 2024
User: vlsi
Version: V-2023.12
Command line: urg -dir simv.vdb
Number of tests: 1

Total Coverage Summary

SCORE	GROUP
100.00	100.00

Total Groups Coverage Summary

SCORE	WEIGHT
100.00	1



Group : $unit::alu_Coverage::alu_cov_group

dashboard | hierarchy | modlist | **groups** | **tests** | asserts

Variable : op_codeCov 〉 Cross : cross_coverage_1 〉 Cross : cross_coverage_2 〉

Summary for Cross cross_coverage_2

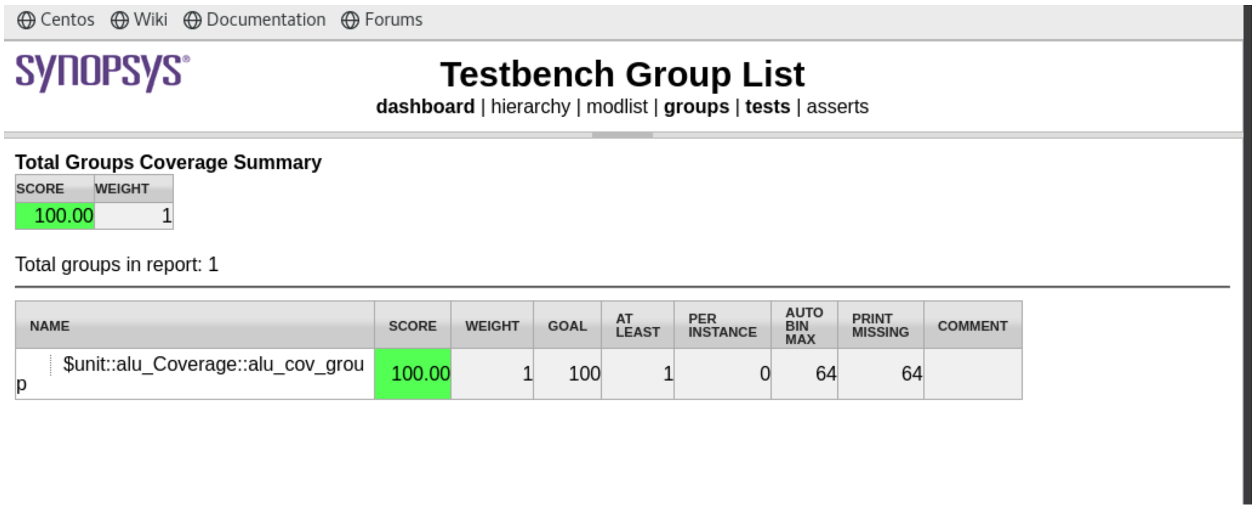
Samples crossed: bit_inCov srcAcCov srcCyCov

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT	MISSING
Automatically Generated Cross Bins	8	0	8	100.00	

Automatically Generated Cross Bins for cross_coverage_2

Bins

bit_inCov	srcAcCov	srcCyCov	COUNT	AT LEAST
auto[0]	auto[0]	auto[0]	52	1
auto[0]	auto[0]	auto[1]	42	1
auto[0]	auto[1]	auto[0]	59	1
auto[0]	auto[1]	auto[1]	61	1
auto[1]	auto[0]	auto[0]	45	1
auto[1]	auto[0]	auto[1]	45	1
auto[1]	auto[1]	auto[0]	58	1
auto[1]	auto[1]	auto[1]	50	1



Centos Wiki Documentation Forums

Testbench Group List

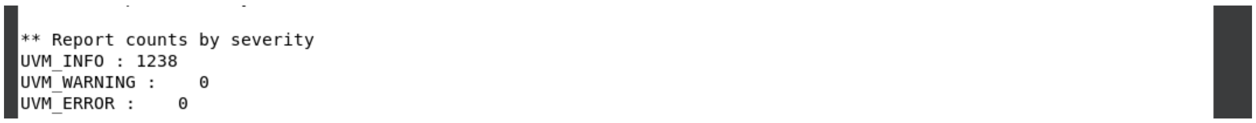
dashboard | hierarchy | modlist | **groups** | **tests** | asserts

Total Groups Coverage Summary

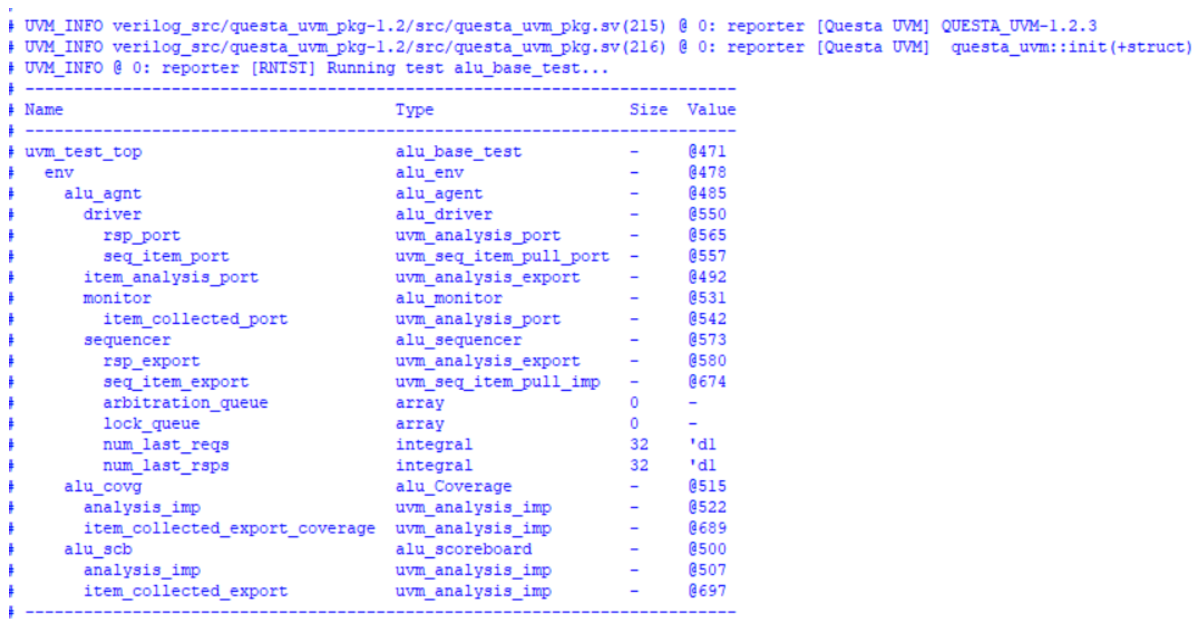
SCORE	WEIGHT
100.00	1

Total groups in report: 1

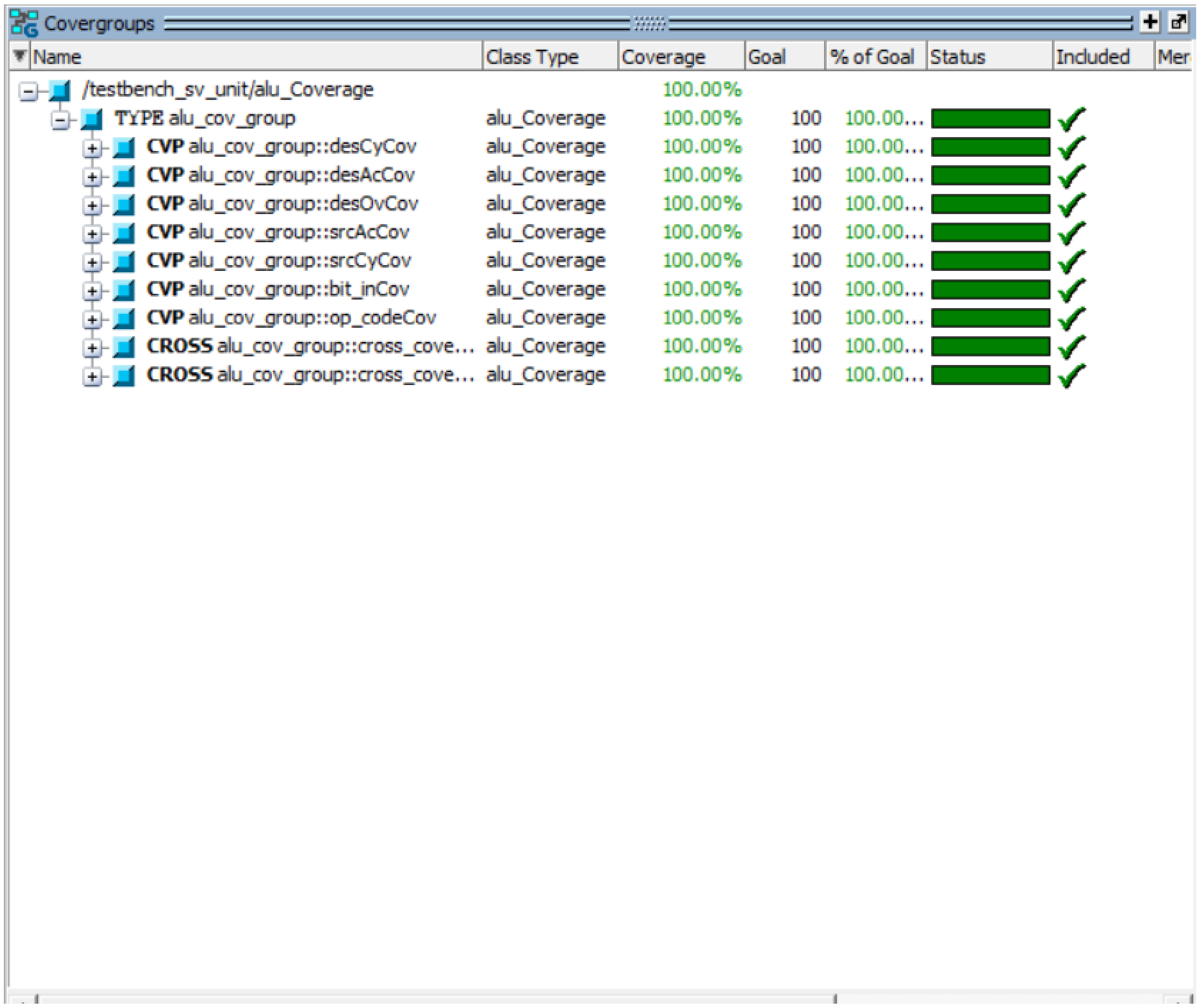
NAME	SCORE	WEIGHT	GOAL	AT LEAST	PER INSTANCE	AUTO BIN MAX	PRINT MISSING	COMMENT
$unit::alu_Coverage::alu_cov_group	100.00	1	100	1	0	64	64	



```
** Report counts by severity
UVM_INFO : 1238
UVM_WARNING :    0
UVM_ERROR :    0
```



```
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(216) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test alu_base_test...
# ------------------------------------------------------------------------
# Name                                  Type                       Size  Value
# ------------------------------------------------------------------------
# uvm_test_top                          alu_base_test              -     @471
#   env                                 alu_env                    -     @478
#     alu_agnt                          alu_agent                  -     @485
#       driver                          alu_driver                 -     @550
#         rsp_port                      uvm_analysis_port          -     @565
#         seq_item_port                 uvm_seq_item_pull_port     -     @557
#       item_analysis_port              uvm_analysis_export        -     @492
#       monitor                         alu_monitor                -     @531
#         item_collected_port           uvm_analysis_port          -     @542
#       sequencer                       alu_sequencer              -     @573
#         rsp_export                    uvm_analysis_export        -     @580
#         seq_item_export               uvm_seq_item_pull_imp      -     @674
#         arbitration_queue             array                      0     -
#         lock_queue                    array                      0     -
#         num_last_reqs                 integral                   32    'd1
#         num_last_rsps                 integral                   32    'd1
#     alu_covg                          alu_Coverage               -     @515
#       analysis_imp                    uvm_analysis_imp           -     @522
#       item_collected_export_coverage  uvm_analysis_imp           -     @689
#     alu_scb                           alu_scoreboard             -     @500
#       analysis_imp                    uvm_analysis_imp           -     @507
#       item_collected_export           uvm_analysis_imp           -     @697
# ------------------------------------------------------------------------
```



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Mer
/testbench_sv_unit/alu_Coverage		100.00%					
TYPE alu_cov_group	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::desCyCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::desAcCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::desOvCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::srcAcCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::srcCyCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::bit_inCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CVP alu_cov_group::op_codeCov	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CROSS alu_cov_group::cross_cove...	alu_Coverage	100.00%	100	100.00...	🟩	✓	
CROSS alu_cov_group::cross_cove...	alu_Coverage	100.00%	100	100.00...	🟩	✓	

SystemVerilog codes

Testbench:

```
`include "uvm_macros.svh"
import uvm_pkg::*;
//including interface and testcase files
`include "alu_interface.sv"
`include "alu_test.sv" //alu_base_test file
`include "oc8051_alu.v"
//----------------------------------------------------------------
module tbench_top;
  //---------------------------------------
  //clock and reset signal declaration
  //---------------------------------------
  bit clk;
  bit rst;

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


  //---------------------------------------
  //interface instance
  //----------------------------------------
  alu_interface intf(clk,rst);

  //----------------------------------------
  //alu instance
  //----------------------------------------
  oc8051_alu DUT (
    .clk(intf.clk),
    .rst(intf.rst),
    .op_code(intf.op_code),
    .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)
   );
//----------------------------------------
  //passing the interface handle to UVM config database
  //----------------------------------------
  initial begin
    uvm_config_db#(virtual
alu_interface)::set(uvm_root::get(),"*","vif",intf);
  End



  //----------------------------------------
  //calling test
  //----------------------------------------
  initial begin
```

```
    run_test("alu_base_test");
  end
Endmodule
```

Interface

```
/*`include "oc8051_timescale.v"*/
interface alu_interface(input logic clk,rst);

  //---------------------------------------
  //declaring the signals
  //---------------------------------------


 logic        srcCy, srcAc, bit_in;
logic  [3:0] op_code;
logic  [7:0] src1, src2, src3;
logic      desCy, desAc, desOv;
logic [7:0] des1, des2, des_acc, sub_result;
  //---------------------------------------
  //driver clocking block
  //---------------------------------------
  clocking driver_cb @(posedge clk);
    default input #1 output #1;
    output srcCy;
    output srcAc;
    output bit_in;
    output  op_code;
      output  src1,src2,src3;
    input  desCy, desAc, desOv;
      input  des1, des2, des_acc, sub_result;
  endclocking

  //---------------------------------------
  //monitor clocking block
  //---------------------------------------
  clocking monitor_cb @(posedge clk);
    default input #1 output #1;
    input srcCy;
    input srcAc;
    input bit_in;
    input  op_code;
      input  src1,src2,src3;
    input  desCy, desAc, desOv;
      input  des1, des2, des_acc, sub_result;
  endclocking

  //---------------------------------------
  //driver modport
  //---------------------------------------
  modport DRIVER  (clocking driver_cb,input clk,rst);

  //---------------------------------------
  //monitor modport
  //---------------------------------------
  modport MONITOR (clocking monitor_cb,input clk,rst);

endinterface
```

The test

```
`include "alu_env.sv"
`include "sequence.sv"
class alu_base_test extends uvm_test;
  `uvm_component_utils(alu_base_test)
  //---------------------------------------
  // env instance
  //---------------------------------------
  alu_env env;
  my_sequence seq;
  //uvm_domain domain1;
  //---------------------------------------
  // constructor
  //---------------------------------------
  function new(string name = "alu_base_test",uvm_component parent=null);
    super.new(name,parent);
  endfunction : new
  //---------------------------------------
  // build_phase
  //---------------------------------------
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    // Create the env
    env = alu_env::type_id::create("env", this);
  endfunction : build_phase
  task run_phase(uvm_phase phase);
      super.run_phase(phase);
      phase.raise_objection(this);
      seq=my_sequence::type_id::create("seq");
      seq.start(env.alu_agnt.sequencer);
      phase.drop_objection(this);
  endtask : run_phase
  //---------------------------------------
  // end_of_elobaration phase
  //---------------------------------------
  virtual function void end_of_elaboration();
    //print's the topology
    print();
  endfunction
endclass : alu_base_test
```

Test Environment

```
import uvm_pkg::*;
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"
`endif
 `include "alu_agent.sv"
`include "alu_scoreboard.sv"
`include "alu_Coverage.sv"
class alu_env extends uvm_env;

  //---------------------------------------
  // agent and scoreboard instance
  //---------------------------------------
      alu_agent      alu_agnt;
      alu_scoreboard alu_scb;
      alu_Coverage alu_covg;
  `uvm_component_utils(alu_env)

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

  //---------------------------------------
  // build_phase - create the components
  //---------------------------------------
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    alu_agnt = alu_agent::type_id::create("alu_agnt", this);
      alu_scb  = alu_scoreboard::type_id::create("alu_scb", this);
      alu_covg=alu_Coverage::type_id::create("alu_covg",this);
  endfunction : build_phase

  //---------------------------------------
  // connect_phase - connecting monitor and scoreboard port
  //---------------------------------------
  function void connect_phase(uvm_phase phase);
      alu_agnt.item_analysis_port.connect(alu_scb.item_collected_export);
                  alu_agnt.item_analysis_port.connect(alu_covg.item_collected
      _export_coverage);

  endfunction : connect_phase
endclass : alu_env
```

The agent

```
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"
`endif
`include "sequencer.sv"
`include "alu_driver.sv"
`include "alu_monitor.sv"


import uvm_pkg::*;
class alu_agent extends uvm_agent;
  //---------------------------------------
  // component instances
  //---------------------------------------
  alu_driver    driver;
  alu_sequencer sequencer;
  alu_monitor   monitor;
  uvm_analysis_export#(seq_item) item_analysis_port;
  `uvm_component_utils(alu_agent)
  //---------------------------------------
  // constructor
  //---------------------------------------
  function new (string name, uvm_component parent);
    super.new(name, parent);
    item_analysis_port=new("item_analysis_port",this);
  endfunction : new
  //---------------------------------------
  // build_phase
  //---------------------------------------
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    monitor = alu_monitor::type_id::create("monitor", this);
    //creating driver and sequencer only for ACTIVE agent
    if(get_is_active() == UVM_ACTIVE) begin
      driver    = alu_driver::type_id::create("driver", this);
      sequencer = alu_sequencer::type_id::create("sequencer", this);
    end
  endfunction : build_phase
  //---------------------------------------
  // connect_phase - connecting the driver and sequencer port
  //---------------------------------------
  function void connect_phase(uvm_phase phase);
    if(get_is_active() == UVM_ACTIVE) begin
      driver.seq_item_port.connect(sequencer.seq_item_export); //driver and
sequencer by default have TLM ports define
    end
    monitor.item_collected_port.connect(item_analysis_port);
 endfunction : connect_phase
endclass : alu_agent
```

Sequencer

```
import uvm_pkg::*;
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"

`endif
class alu_sequencer extends uvm_sequencer#(seq_item);
`uvm_component_utils(alu_sequencer)
function new(string name="sequencer", uvm_component parent);

super.new(name,parent);
endfunction

endclass
```

The Sequence

```
import uvm_pkg::*;
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"
`endif
class my_sequence extends uvm_sequence#(seq_item);
`uvm_object_utils(my_sequence)
seq_item req;
function new(string name="my_sequence");
super.new(name);
endfunction

task body();
req=seq_item::type_id::create("req");
repeat(1000) begin
assert(req.randomize());
`uvm_send(req);
end
endtask
endclass
```

Sequence Item

```
import uvm_pkg::*;
`include "uvm_macros.svh"
`define flag 1
class seq_item  extends uvm_sequence_item;
rand bit       srcCy, srcAc, bit_in,clk,rst;
randc bit [3:0] op_code;
rand bit  [7:0] src1, src2, src3;
bit     desCy, desAc, desOv;
bit [7:0] des1, des2, des_acc, sub_result;

constraint srcCy_dist {srcCy dist {0:/50, 1:/50};}
constraint srcAc_dist {srcAc dist {0:/50, 1:/50};}
constraint bit_in_dist {bit_in dist {0:/50, 1:/50};}
constraint op_code_dist {op_code dist {4'd0:/6, 4'd1:/6,4'd2:/6,
4'd3:/6,4'd4:/6,4'd5:/6,4'd6:/6,4'd7:/6,4'd8:/6,4'd9:/6,4'd10:/6,
4'd11:/6,4'd12:/7,4'd13:/7,4'd14:/7,4'd15:/7};}
`uvm_object_utils_begin(seq_item)
    `uvm_field_int(op_code, UVM_ALL_ON)
    `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(bit_in, 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(sub_result, UVM_ALL_ON)
  `uvm_object_utils_end

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

The Driver

```
`define DRIV_IF vif.DRIVER.driver_cb
import uvm_pkg::*;
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"

`endif
```

```
class alu_driver extends uvm_driver #(seq_item);
  //--------------------------------------
  // Virtual Interface
  //--------------------------------------
  virtual alu_interface vif;
  `uvm_component_utils(alu_driver)
  //--------------------------------------
  // Constructor
  //--------------------------------------
  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);
     if(!uvm_config_db#(virtual alu_interface)::get(this, "", "vif", vif))
       `uvm_fatal("NO_VIF",{"virtual interface must be set for:
",get_full_name(),".vif"});
  endfunction: build_phase
  //--------------------------------------
  // run phase
  //--------------------------------------
  virtual task run_phase(uvm_phase phase);
    repeat(1000) begin
      seq_item_port.get_next_item(req);
      drive();
      seq_item_port.item_done();
    end
  endtask : run_phase



//--------------------------------------
  // drive - transaction level to signal level
  // drives the value's from seq_item to interface signals
  //--------------------------------------
  virtual task drive();
```

```
    @(posedge vif.DRIVER.clk);

    `DRIV_IF.src1 <= req.src1;
    `DRIV_IF.src2 <= req.src2;
    `DRIV_IF.src3 <= req.src3;
    `DRIV_IF.srcCy <= req.srcCy;
    `DRIV_IF.srcAc <= req.srcAc;
    `DRIV_IF.bit_in <= req.bit_in;
      `DRIV_IF.op_code <= req.op_code;
    if(req.op_code==4'h3 || req.op_code==4'h4)
      begin
      @(posedge vif.DRIVER.clk);
      @(posedge vif.DRIVER.clk);
      @(posedge vif.DRIVER.clk);
      end
      else
      @(posedge vif.DRIVER.clk);

      req.des1 = `DRIV_IF.des1;
      req.des2 = `DRIV_IF.des2;
      req.des_acc = `DRIV_IF.des_acc;
      req.desCy = `DRIV_IF.desCy;
      req.desAc = `DRIV_IF.desAc;
      req.desOv = `DRIV_IF.desOv;
      req.sub_result = `DRIV_IF.sub_result;
  endtask : drive
endclass : alu_driver
```

Coverage:

```
import uvm_pkg::*;
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"
`endif
class alu_Coverage extends uvm_subscriber #(seq_item);
```

```
`uvm_component_utils(alu_Coverage)
seq_item coverage_txn;
uvm_analysis_imp #(seq_item,alu_Coverage)
item_collected_export_coverage;
//seq_item pkt_qu [$];
real cov;
//---------------------------------------------------------------//
      /////////////////Constructor/////////////////
      covergroup alu_cov_group;
            desCyCov: coverpoint coverage_txn.desCy;
            desAcCov: coverpoint coverage_txn.desAc;
            desOvCov: coverpoint coverage_txn.desOv;
            srcAcCov: coverpoint coverage_txn.srcAc;
            srcCyCov: coverpoint coverage_txn.srcCy;
            bit_inCov: coverpoint coverage_txn.bit_in;
            op_codeCov: coverpoint coverage_txn.op_code;
            cross_coverage_1: cross desAcCov,desCyCov,desOvCov;
            cross_coverage_2: cross bit_inCov,srcAcCov,srcCyCov;
      endgroup

      function new(string name, uvm_component parent);
            super.new(name, parent);
            alu_cov_group  =new();
      endfunction
      virtual function void write(seq_item t);
            coverage_txn=t;
            alu_cov_group.sample();
      endfunction

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

      item_collected_export_coverage=new("item_collected_export_coverag
e",this);
      endfunction : build_phase
```

```
    function void extract_phase(uvm_phase phase);
        cov=alu_cov_group.get_coverage();
    endfunction

    function void report_phase(uvm_phase phase);
        `uvm_info(get_full_name(),$sformatf("Coverage is
%d",cov),UVM_MEDIUM);
    endfunction
endclass
```

Monitor

```
import uvm_pkg::*;
`include "uvm_macros.svh"
`ifndef flag
`include "seq_item.sv"

`endif
class alu_monitor extends uvm_monitor;

  //---------------------------------------
  // Virtual Interface
  //---------------------------------------
  virtual alu_interface vif;

  //---------------------------------------
  // analysis port, to send the transaction to scoreboard
  //---------------------------------------


   uvm_analysis_port#(seq_item) item_collected_port;

  seq_item trans_collected;

  `uvm_component_utils(alu_monitor)

  //---------------------------------------
  // new - constructor
  //---------------------------------------
  function new (string name, uvm_component parent);
    super.new(name, parent);
    trans_collected = new();

    //TODO: Step 2: Create Analysis Port
      item_collected_port = new("item_collected_port", this);
  endfunction : new

  //---------------------------------------
  // build_phase - getting the interface handle
  //---------------------------------------
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db#(virtual alu_interface)::get(this, "", "vif", vif))
       `uvm_fatal("NOVIF",{"virtual interface must be set for:
",get_full_name(),".vif"});
  endfunction: build_phase
//---------------------------------------
  // run_phase - convert the signal level activity to transaction level.
  // i.e, sample the values on interface signal ans assigns to transaction
class fields
  //---------------------------------------
  virtual task run_phase(uvm_phase phase);
    repeat(1000) begin
      @(posedge vif.MONITOR.clk);
            trans_collected.srcCy=vif.monitor_cb.srcCy;
            trans_collected.srcAc=vif.monitor_cb.srcAc;
            trans_collected.bit_in=vif.monitor_cb.bit_in;
            trans_collected.op_code=vif.monitor_cb.op_code;
            trans_collected.src1=vif.monitor_cb.src1;
            trans_collected.src2=vif.monitor_cb.src2;
            trans_collected.src3=vif.monitor_cb.src3;
             if(vif.monitor_cb.op_code==4'h3 || vif.monitor_cb.op_code==4'h4)
            begin
                  @(posedge vif.DRIVER.clk);
                  @(posedge vif.DRIVER.clk);
```

```
                    @(posedge vif.DRIVER.clk);
              end
              else
                    @(posedge vif.DRIVER.clk);
              trans_collected.desCy=vif.monitor_cb.desCy;
              trans_collected.desOv=vif.monitor_cb.desOv;
              trans_collected.des1=vif.monitor_cb.des1;
          trans_collected.des2 = vif.monitor_cb.des2;
          trans_collected.des_acc = vif.monitor_cb.des_acc;
          trans_collected.desAc = vif.monitor_cb.desAc;
              trans_collected.sub_result=vif.monitor_cb.sub_result;


              item_collected_port.write(trans_collected);
     end
    endtask : run_phase

endclass : alu_monitor
```

Scoreboard

```
`ifndef flag
`include "seq_item.sv"
`endif
import uvm_pkg::*;
`include "uvm_macros.svh"
class alu_scoreboard extends uvm_subscriber #(seq_item);

  `uvm_component_utils(alu_scoreboard)

      seq_item pkt_qu [$];
      uvm_analysis_imp #(seq_item,alu_scoreboard)
item_collected_export;
  logic [7:0] des1_alu, des2_alu, des_acc_alu, sub_result_alu;
     bit desCy_alu, desAc_alu, desOv_alu;
      logic        srcCy, srcAc, bit_in;
logic  [3:0] op_code;
logic  [7:0] src1, src2, src3;
bit temp_carry,temp_carry2,desAc_alu_temp,da_tmp,da_tmp1;
bit [3:0] temp_4;
bit [6:0] out_temp;
bit [15:0] inc,dec;
int error;
  //---------------------------------------
  //constructor
  //---------------------------------------
  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
     item_collected_export=new("item_collected_export",this);
  endfunction : build_phase

  //---------------------------------------
  // write method
  //---------------------------------------

  virtual function void write(seq_item t);
    pkt_qu.push_back(t);
  endfunction



//---------------------------------------
  // run_phase - compare's the read data with the expected data(stored
in local memory)
  // local memory will be updated on the write operation.
  //---------------------------------------
  virtual task run_phase(uvm_phase phase);
    seq_item alu_pkt;
     repeat(1000) begin
     wait(pkt_qu.size() > 0);
```

```
          alu_pkt=pkt_qu.pop_front();
          src1=alu_pkt.src1;
          src2=alu_pkt.src2;
          src3=alu_pkt.src3;
          srcCy=alu_pkt.srcCy;
          srcAc=alu_pkt.srcAc;
          bit_in=alu_pkt.bit_in;
          op_code=alu_pkt.op_code;
    if(alu_pkt.op_code==4'h0) begin  //NOP
        des_acc_alu = src1;
          des1_alu = src1;
          des2_alu = src2;
          desCy_alu = srcCy;
          desAc_alu = srcAc;
          desOv_alu = 1'b0;
     end
       else if(alu_pkt.op_code==4'h1) begin  //ADD
          {temp_carry,des_acc_alu} = src1+src2+srcCy;
          {temp_carry2,out_temp}=src1[6:0]+src2[6:0]+srcCy;
          des1_alu = src1;
          desCy_alu=temp_carry;
          des2_alu = src3+ {7'b0, temp_carry};
          {desAc_alu,temp_4}=src1[3:0]+src2[3:0]+srcCy;
          desOv_alu= temp_carry ^ temp_carry2;
       end
       else if(alu_pkt.op_code==4'h2) begin  //SUB

          {temp_carry,des_acc_alu} = {1'b1,src1}-{1'b0,src2}-
{3'b0,srcCy};
          {temp_carry2,out_temp}={1'b1,src1[6:0]}-{1'b0,src2[6:0]}-
{3'b0,srcCy};
          des1_alu = 8'h00;
          des2_alu = 8'h00;
          desCy_alu=!temp_carry;
```

```
            {desAc_alu_temp,temp_4}={1'b1,src1[3:0]}-{1'b0,src2[3:0]}-
{3'b0,srcCy};
            desAc_alu=!desAc_alu_temp;
            desOv_alu= !temp_carry ^ !temp_carry2;
        end
        else if(alu_pkt.op_code==4'h3) begin  //<Multiply
            {des_acc_alu,des2_alu} = src1*src2;
            des1_alu = src1;
            desCy_alu=1'b0;
            desAc_alu=1'b0;
            desOv_alu= |des_acc_alu;
        end
         else if(alu_pkt.op_code==4'h4) begin  //<divide
                if(src2==0) begin
                      des_acc_alu=src1;
                      des2_alu=8'hFF;
                end
                else begin
                      des_acc_alu=src1%src2;
                      des2_alu = src1/src2;
                end
                      des1_alu = src1;
                      desCy_alu=1'b0;
                      desAc_alu=1'b0;
                      desOv_alu= src2==8'h00;
        end
        else if(alu_pkt.op_code==4'h5) begin  //<operational decimal
adjustment


        if (srcAc==1'b1 | src1[3:0]>4'b1001) {da_tmp, des_acc_alu[3:0]}
= {1'b0, src1[3:0]}+ 5'b00110;
        else {da_tmp, des_acc_alu[3:0]} = {1'b0, src1[3:0]};


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

```
      else {da_tmp1, des_acc_alu[7:4]} = {srcCy, src1[7:4]} + {4'b0,
da_tmp};
           des1_alu = src1;
           des2_alu = 8'h00;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
           desCy_alu=da_tmp |da_tmp1;

        end
        else if(alu_pkt.op_code==4'h6) begin  //Not
           des_acc_alu = ~src1;
           des1_alu = ~src1;
           des2_alu = 8'h00;
           desCy_alu= !srcCy;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'h7) begin  //and
           des_acc_alu = src1 & src2;
           des1_alu = src1 & src2;
           des2_alu = 8'h00;
           desCy_alu = srcCy & bit_in;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;

        end
        else if(alu_pkt.op_code==4'h8) begin  //xor
           des_acc_alu = src1 ^ src2;
           des1_alu = src1 ^ src2;
           des2_alu = 8'h00;
           desCy_alu = srcCy ^ bit_in;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'h9) begin  //or
```

```
           des_acc_alu = src1 | src2;
           des1_alu = src1 | src2;
           des2_alu = 8'h00;
           desCy_alu = srcCy | bit_in;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'hA) begin  //rotate left
           des_acc_alu = {src1[6:0], src1[7]};
           des1_alu = src1 ;
           des2_alu = 8'h00;
           desCy_alu = srcCy | !bit_in;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'hB) begin  //rotate left with carry
swap nibbles
           des_acc_alu = {src1[6:0], srcCy};
           des1_alu = src1 ;
           des2_alu = {src1[3:0], src1[7:4]};
           desCy_alu = src1[7];
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'hC) begin  //rotate right
           des_acc_alu = {src1[0], src1[7:1]};
           des1_alu = src1 ;
           des2_alu = 8'h00;
           desCy_alu = srcCy & !bit_in;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'hD) begin  //rotate right with carry
           des_acc_alu = {srcCy, src1[7:1]};
           des1_alu = src1 ;
           des2_alu = 8'h00;
           desCy_alu = src1[0];
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'hE) begin  //operation pcs Add
           dec={src2, src1} - {15'h0, 1'b1};
           inc = {src2, src1} + {15'h0, 1'b1};
           if (srcCy) begin
               des_acc_alu = dec[7:0];
               des1_alu = dec[7:0];
               des2_alu = dec[15:8];
           end
           else begin
               des_acc_alu = inc[7:0];
               des1_alu = inc[7:0];
               des2_alu = inc[15:8];
           end
           desCy_alu = 1'b0;
           desAc_alu = 1'b0;
           desOv_alu = 1'b0;
        end
        else if(alu_pkt.op_code==4'hF) begin  //operation exchange
           if (srcCy)
           begin
               des_acc_alu = src2;
               des1_alu = src2;
               des2_alu = src1;
           end
           else begin
               des_acc_alu = {src1[7:4],src2[3:0]};
               des1_alu = {src1[7:4],src2[3:0]};
               des2_alu = {src2[7:4],src1[3:0]};
           end
           desCy_alu = 1'b0;
```

```
            desAc_alu = 1'b0;
            desOv_alu = 1'b0;
        end
       if(des1_alu==alu_pkt.des1 && des2_alu==alu_pkt.des2 &&
des_acc_alu==alu_pkt.des_acc &&
         descy_alu==alu_pkt.desCy && desAc_alu==alu_pkt.desAc &&
desOv_alu==alu_pkt.desOv)
            begin
                  `uvm_info(get_type_name(),$sformatf("------ :: packet
Match :: ------"),UVM_LOW)

                  `uvm_info(get_type_name(),$sformatf("Expected Data:
%0h %0h %0h %0h %0h %0h Actual Data: %0h %0h %0h %0h %0h %0h"

      ,des1_alu,des2_alu,des_acc_alu,desCy_alu,desAc_alu,desOv_alu,

      alu_pkt.des1,alu_pkt.des2,alu_pkt.des_acc,alu_pkt.desCy,alu_pkt.d
esAc,alu_pkt.desOv),UVM_LOW)
                  `uvm_info(get_type_name(),"---------------------------
---------",UVM_LOW)

            end
            else
            begin
                  `uvm_error("MYERR",$sformatf("------ :: packet
Mismatch :: ------"))
                  `uvm_info(get_type_name(),$sformatf("inputs srcCy %d
srcAc %d bit_in %d op_code %d src1 %d src2 %d src3 %d",srcCy, srcAc,
bit_in,op_code,src1, src2, src3),UVM_LOW)
                  `uvm_info(get_type_name(),$sformatf("Expected Data:
%0h %0h %0h %0h %0h %0h Actual Data: %0h %0h %0h %0h %0h %0h"

      ,des1_alu,des2_alu,des_acc_alu,desCy_alu,desAc_alu,desOv_alu,

      alu_pkt.des1,alu_pkt.des2,alu_pkt.des_acc,alu_pkt.desCy,alu_pkt.d
esAc,alu_pkt.desOv),UVM_LOW)

                  `uvm_info(get_type_name(),"---------------------------
---------",UVM_LOW)
                  error=error+1;
            end
        end
         `uvm_info(get_type_name(),$sformatf("Number of errors: %0d "
                    ,error),UVM_LOW)
         endtask
        Endclass
```

DUT Codes

available at: https://shorturl.at/lvwX6

Used commands in VCS & Questasim

VCS

Compilation

vcs – sverilog -ntb_opts uvm-1.2 testbench.sv

Simulation

./simv simv.log +UVM_TESTNAME=alu_base_test

Coverage

./simv -cm line+cond+fsm -cm_name testbench

urg -dir ./simv.vdb -format both

Questasim

Compilation

vlog -cover bcs -work work testbench.sv

Simulation

vsim -voptargs=+acc -coverage work.tbench_top -sv_seed 30

Adding wave

```
add wave -r /*
```

Running

```
run -all
```