Name: Sushan Patel
UIN: 334003274

Lab 09 Report

1. htax_tx_monitor_c.sv:

```
class htax_tx_monitor_c extends uvm_monitor;

        parameter PORTS = `PORTS;

        `uvm_component_utils(htax_tx_monitor_c)

        uvm_analysis_port #(htax_tx_mon_packet_c)    tx_collect_port;

        virtual interface htax_tx_interface htax_tx_intf;
        htax_tx_mon_packet_c tx_mon_packet;
        int pkt_len;

 covergroup cover_htax_packet;
  option.per_instance = 1;
  option.name = "cover_htax_packet";


  // Coverpoint for htax packet field : destination port
  DEST_PORT : coverpoint tx_mon_packet.dest_port  {
                                                        bins dest_port[] = {[0:3]};
                                                }

  // TO DO : Coverpoint for htax packet field : vc (include vc=0 in illegal bin)
  VC : coverpoint tx_mon_packet.vc  {
                                                                illegal_bins    illegal_vc[]
= {0};
                                                        bins vc[] = {[1:3]};
                                                }



  // TO DO : Coverpoint for htax packet field : length (Divide range [3:63] into 16 bins)
  LENGTH : coverpoint tx_mon_packet.length  {
                                                        bins length[16] = {[3:63]};
                                                }
```

```
                // Coverpoints for Crosses
                // TO DO : DEST_PORT cross VC
  VCxDEST_PORT : cross VC, DEST_PORT;



                // TO DO : DEST_PORT cross LENGTH
  LENGTHxDEST_PORT : cross LENGTH, DEST_PORT;


                // TO DO : VC cross LENGTH
  VCxLENGTH : cross VC, LENGTH;


 endgroup

        covergroup cover_htax_tx_intf;
  option.per_instance = 1;
  option.name = "cover_htax_tx_intf";


                // TO DO : Coverpoint for tx_outport_req: covered all the values 0001,0010,0100,1000

  TX_OUTPORT_REQ : coverpoint htax_tx_intf.tx_outport_req  {
                                                bins            tx_outport_req[4]           =
{'b0001,'b0010,'b0100,'b1000};
                                        }




                // TO DO : Coverpoint for tx_vc_req: All the VCs are requested atleast once. Ignore what
is not allowed, or put it as illegal

  TX_VC_REQ : coverpoint htax_tx_intf.tx_vc_req  {
                                                bins tx_vc_req[3] = {'b01,'b10,'b11};
                                                        illegal_bins
illegal_tx_vc_req = {'b0};
                                        }




                // TO DO : Coverpoint for tx_vc_gnt: All the virtual channels are granted atleast once.

  TX_VC_GNT : coverpoint htax_tx_intf.tx_vc_gnt  {
```

```
                                                bins tx_vc_gnt[3] = {'b01,'b10,'b11};
                                        }



        endgroup

        //constructor
        function new (string name, uvm_component parent);
                super.new(name,parent);
                tx_collect_port = new("tx_collect_port",this);
                tx_mon_packet           = new();

                //Instance for the covergroup cover_htax_packet
                this.cover_htax_packet = new();
                //Instance for the covergroup cover_htax_tx_intf
                this.cover_htax_tx_intf = new();
        endfunction : new

 //UVM build phase
 function void build_phase(uvm_phase phase);
  super.build_phase(phase);
                if(!uvm_config_db#(virtual htax_tx_interface)::get(this,"","tx_vif",htax_tx_intf))
                        `uvm_fatal("NO_TX_VIF",{"Virtual Interface needs to be set for ",
get_full_name(),".tx_vif"})
        endfunction : build_phase

        task run_phase(uvm_phase phase);
                forever begin
                        pkt_len=0;

                        //Assign tx_mon_packet.dest_port from htax_tx_intf.tx_outport_req
                        @(posedge |htax_tx_intf.tx_vc_gnt) begin

                                for(int i=0; i < PORTS; i++)
                                        if(htax_tx_intf.tx_outport_req[i]==1'b1)
                                                tx_mon_packet.dest_port = i;

                                //Assign tx_vc_req to tx_mon_packet.vc
                                tx_mon_packet.vc = htax_tx_intf.tx_vc_req;

                                cover_htax_tx_intf.sample();      //Sample Coverage on htax_tx_intf
                        end

                        @(posedge htax_tx_intf.clk)
                        //On consequtive cycles append htax_tx_intf.tx_data to tx_mon_packet.data[]
till htax_tx_intf.tx_eot pulse
```

```
                        while(htax_tx_intf.tx_eot==0) begin
                                @(posedge htax_tx_intf.clk)
                                tx_mon_packet.data = new[++pkt_len] (tx_mon_packet.data);
                                tx_mon_packet.data[pkt_len-1]=htax_tx_intf.tx_data;
                        end
                        //Assign pkt_len to tx_mon_packet.length
                        tx_mon_packet.length = pkt_len;
                        tx_collect_port.write(tx_mon_packet);
                        cover_htax_packet.sample();      //Sample Coverage on tx_mon_packet
                end
        endtask : run_phase

endclass : htax_tx_monitor_c
```

2. htax_rx_monitor_c.sv:

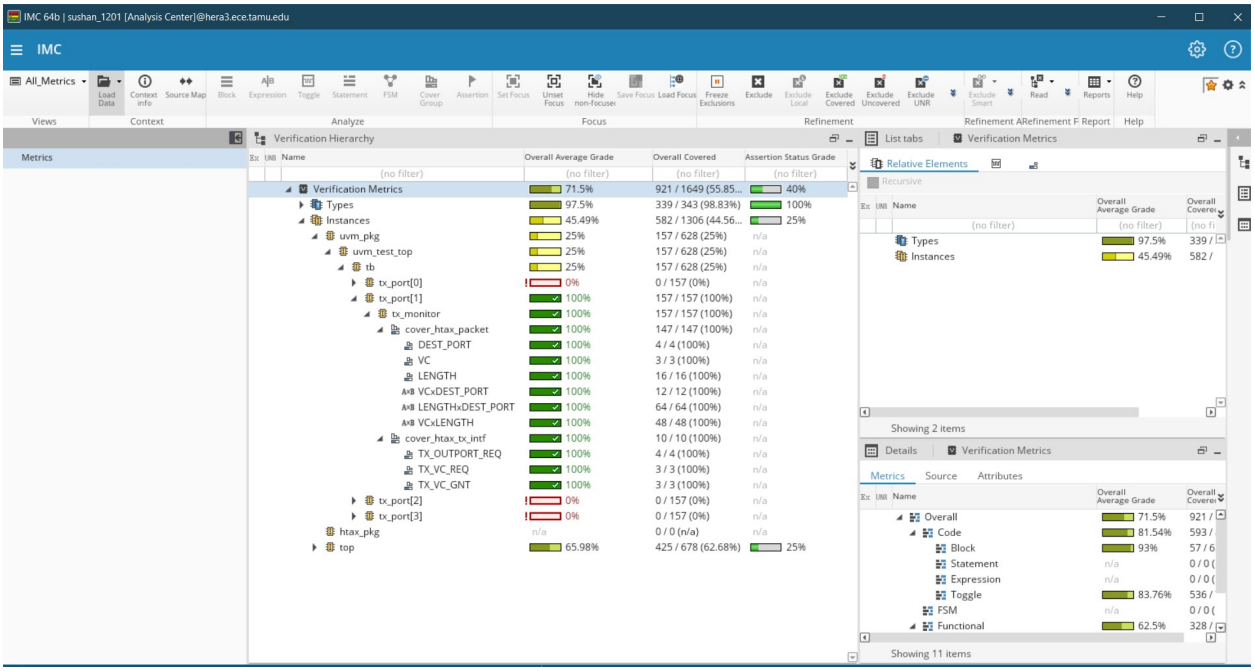
```
////////////////////////////////////////////////////////////////////////////
// Texas A&M University
// CSCE 616 Hardware Design Verification
// Created by  : Prof. Quinn and Saumil Gogri
////////////////////////////////////////////////////////////////////////////

class htax_rx_monitor_c extends uvm_monitor;

        `uvm_component_utils(htax_rx_monitor_c)

        //Analysis port to communicate with Scoreboard
        uvm_analysis_port #(htax_rx_mon_packet_c) rx_collect_port;

        virtual interface htax_rx_interface htax_rx_intf;
        htax_rx_mon_packet_c rx_mon_packet;
        int pkt_len;

  covergroup cover_htax_packet;
    option.per_instance = 1;
    option.name = "cover_htax_packet";

    TX_VC_REQ : coverpoint htax_rx_intf.rx_vc_req  {
                                                        bins rx_vc_req[3] = {'b01,'b10,'b11};
                                                                illegal_bins
illegal_rx_vc_req = {'b0};
                                                }

  endgroup

        function new (string name, uvm_component parent);
                super.new(name, parent);
                rx_collect_port = new ("rx_collect_port", this);
```

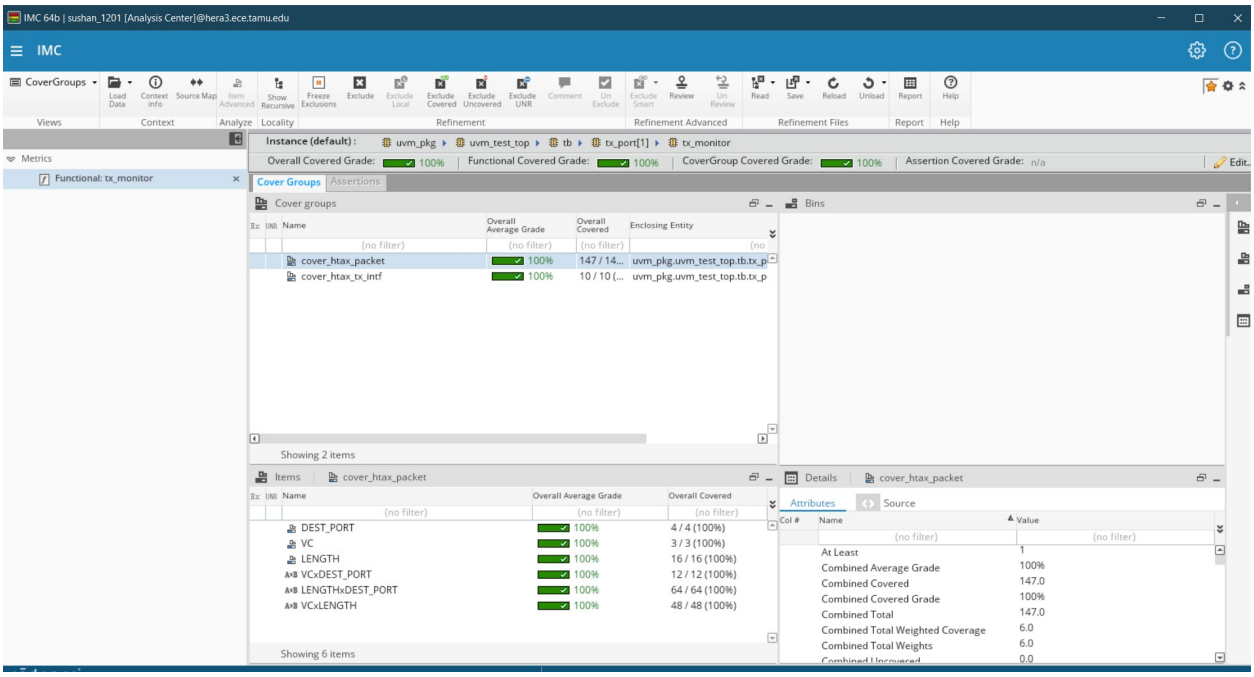
```
		rx_mon_packet		= new();
	endfunction : new

	function void build_phase (uvm_phase phase);
		super.build_phase(phase);
		if(!uvm_config_db#(virtual htax_rx_interface)::get(this,"","rx_vif",htax_rx_intf))
			`uvm_fatal("RX_VIF",{"Virtual Interface needs to be set for ",get_full_name(),".rx_vif"})
	endfunction : build_phase

	task run_phase(uvm_phase phase);
		forever begin
			pkt_len=0;
			//Wait for rising edge of htax_rx_intf.rx_sot
			@(posedge (|htax_rx_intf.rx_sot))

			//On consequtive cycles append htax_rx_intf.rx_data to rx_mon_packet.data[] till htax_rx_intf.rx_eot pulse
			while(!htax_rx_intf.rx_eot==1) begin
				@(posedge htax_rx_intf.clk);
				rx_mon_packet.data = new [++pkt_len] (rx_mon_packet.data);
				rx_mon_packet.data[pkt_len-1] = htax_rx_intf.rx_data;
			end

			//Write the rx_mon_packet on anaylysis port
			rx_collect_port.write(rx_mon_packet);

		end
	endtask : run_phase
endclass : htax_rx_monitor_c
```

3. Simulation Output:

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


Number of demoted UVM_FATAL reports  :    0
Number of demoted UVM_ERROR reports  :    0
Number of demoted UVM_WARNING reports:    0
Number of caught UVM_FATAL reports   :    0
Number of caught UVM_ERROR reports   :    0
Number of caught UVM_WARNING reports :    0

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO : 3017
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[SCOREBOARD]  2005
[TEST_DONE]     1
[TOP]     5
[UVMTOP]     1
[htax_tx_driver_c]  1000
[simple_random_seq]     3
[simple_random_test]     1
Simulation complete via $finish(1) at time 520970 NS + 45
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit

coverage setup:
  workdir  :  ./cov_work
  dutinst  :  top(top)
  scope    :  scope
  testname :  test

coverage files:
  model(design data) :  ./cov_work/scope/icc_1cfdfc8b_454bc9db.ucm (reused)
  data               :  ./cov_work/scope/test/icc_1cfdfc8b_454bc9db.ucd
TOOL:   xrun    22.03-s004: Exiting on Nov 11, 2023 at 17:18:36 CST  (total: 00:00:03)
```

4. Coverage Report:




IMC 64b | sushan_1201 [Analysis Center]@hera3.ece.tamu.edu
IMC
CoverGroups
Load Data
Context info
Source Map
Item Advanced
Show Recursive
Freeze Exclusions
Exclude
Exclude Local
Exclude Covered
Exclude Uncovered
Exclude UNR
Comment
Un Exclude
Exclude Smart
Review
Un Review
Read
Save
Reload
Unload
Report
Help
Views
Context
Analyze
Locality
Refinement
Refinement Advanced
Refinement Files
Report
Help
Metrics
Functional: tx_monitor
Instance (default) :
uvm_pkg
uvm_test_top
tb
tx_port[1]
tx_monitor
Overall Covered Grade: 100%
Functional Covered Grade: 100%
CoverGroup Covered Grade: 100%
Assertion Covered Grade: n/a
Edit..
Cover Groups
Assertions
Cover groups
Ex
UNR
Name
Overall Average Grade
Overall Covered
Enclosing Entity
(no filter)
cover_htax_packet
100%
147 / 14...
uvm_pkg.uvm_test_top.tb.tx_p
cover_htax_tx_intf
100%
10 / 10 (...
uvm_pkg.uvm_test_top.tb.tx_p
Showing 2 items
Bins
Items
cover_htax_packet
Overall Average Grade
Overall Covered
DEST_PORT
100%
4 / 4 (100%)
VC
100%
3 / 3 (100%)
LENGTH
100%
16 / 16 (100%)
VCxDEST_PORT
100%
12 / 12 (100%)
LENGTHxDEST_PORT
100%
64 / 64 (100%)
VCxLENGTH
100%
48 / 48 (100%)
Showing 6 items
Details
cover_htax_packet
Attributes
Source
Col #
Name
Value
At Least
1
Combined Average Grade
100%
Combined Covered
147.0
Combined Covered Grade
100%
Combined Total
147.0
Combined Total Weighted Coverage
6.0
Combined Total Weights
6.0
Combined Uncovered
0.0