Lab 7

Prachi Anil More

UIN: 434007051

htax_tx_monitor_c.sv Code

htax_tx_monitor_c.sv htax_rx_monitor_c.sv

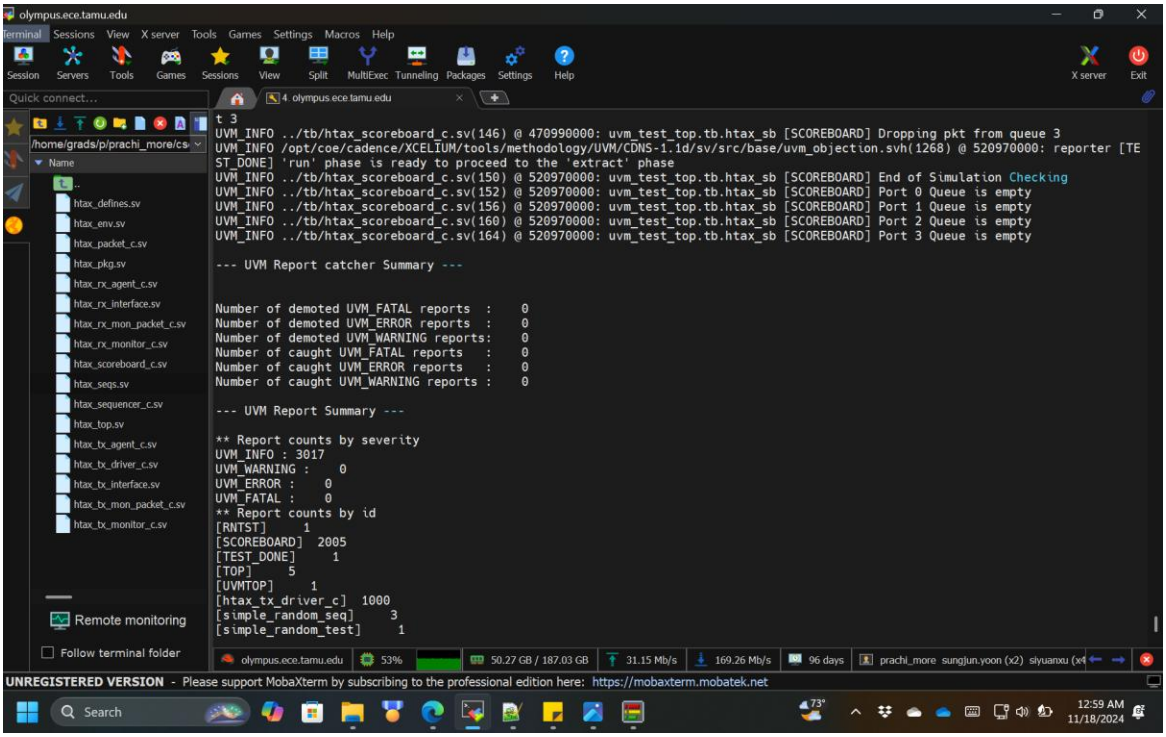
```
                                                    }

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

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



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

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

        // TO DO : VC cross LENGTH
        VC_LENGTH : 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 outport_rq[] = {4'b0001, 4'b0010, 4'b0100, 4'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 vc_req[] = {[1:7]}; // Legal VCs
            illegal_bins illegal_request = {0};
        }

        // 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 vc_gnt[] = {[1:7]};
        }
```

htax_rx_monitor_c.sv Code

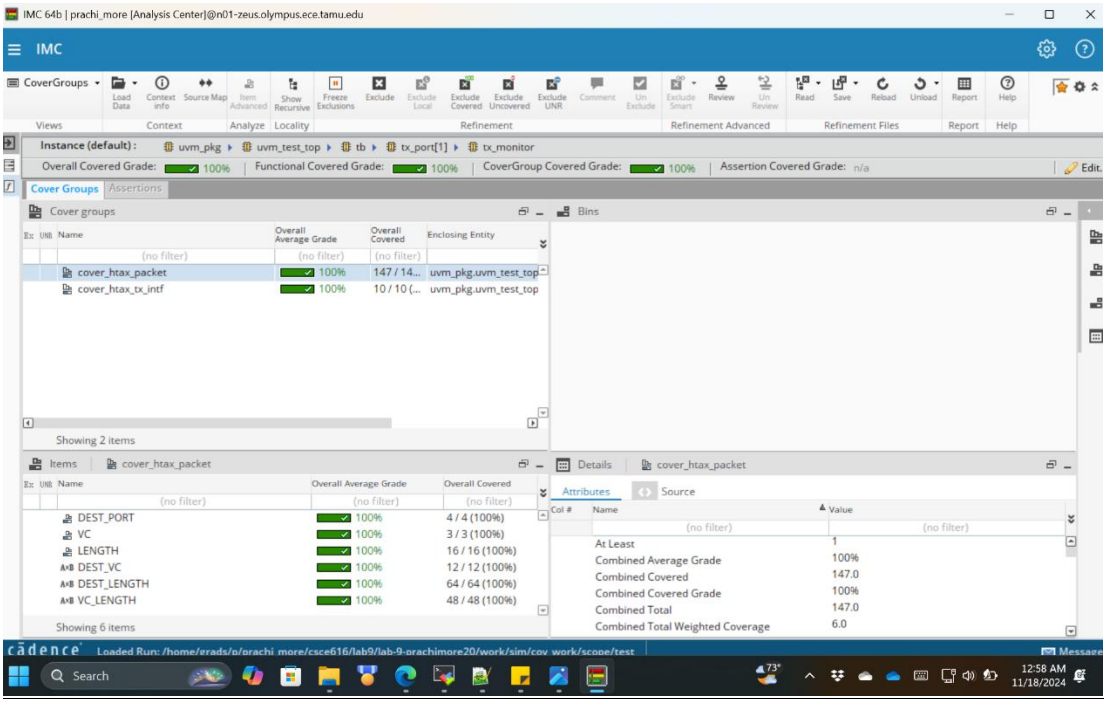
htax_tx_monitor_c.sv | htax_rx_monitor_c.sv

```
1  ////////////////////////////////////////////////////////////////////////////////
2  // Texas A&M University
3  // CSCE 616 Hardware Design Verification
4  // Created by  : Prof. Quinn and Saumil Gogri
5  ////////////////////////////////////////////////////////////////////////////////
6
7  class htax_rx_monitor_c extends uvm_monitor;
8
9      `uvm_component_utils(htax_rx_monitor_c)
10
11     //Analysis port to communicate with Scoreboard
12     uvm_analysis_port #(htax_rx_mon_packet_c) rx_collect_port;
13
14     virtual interface htax_rx_interface htax_rx_intf;
15     htax_rx_mon_packet_c rx_mon_packet;
16     int pkt_len;
17
18     // TO DO : Create covergroup for htax_rx_inf and add at least one coverpoint
19     covergroup cover_htax_rx_intf;
20         option.per_instance = 1;
21         option.name = "cover_htax_rx_intf";
22
23         // Coverpoint for rx_sot: Captures start of transmission signals
24         RX_SOT : coverpoint htax_rx_intf.rx_sot {
25             bins sot_active[] = {1'b1};
26         }
27
28         // Coverpoint for rx_data: Tracks data values received
29         RX_DATA : coverpoint htax_rx_intf.rx_data {
30             bins data_values[] = {[0:255]};
31         }
32         // Coverpoint for rx_eot: Captures end of transmission signals
33         RX_EOT : coverpoint htax_rx_intf.rx_eot {
34             bins eot_active[] = {1'b1}; // Capture active end signals
35
36         }
37     endgroup
38
```

Simulation Output



htax_packet covergroup



htax_intf covergroup