CSCE 616: Introduction to Hardware Design Verification

Lab-10: HTAX Regression

Coverage Closure Report

Name: Sai Surendra Reddy Yaraballi

UIN: 333002926

For Coverage analysis, added assertions in htax_tx_interface

```
//ASSERTIONS

//---------------------------
    // tx_outport_req is one-hot
    //---------------------------
    property tx_outport_req_one_hot;
        @(posedge clk) disable iff(!rst_n)
        (|tx_outport_req) |-> $onehot(tx_outport_req);
    endproperty

    assert_tx_outport_req_one_hot : assert property(tx_outport_req_one_hot)                 //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_outport request is not one hot encoded");

    //-----------------------------------
    // no tx_outport_req without tx_vc_req
    //-----------------------------------
    property tx_outport_req_without_tx_vc_req;
        @(posedge clk) disable iff(!rst_n)
        $rose(|tx_vc_req) |-> $rose(|tx_outport_req);
    endproperty

    assert_tx_outport_req_without_tx_vc_req : assert property(tx_outport_req_without_tx_vc_req)        //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_outport_req high without tx_vc_req ");



    //-----------------------------------
    // no tx_vc_req without tx_outport_req
    //-----------------------------------
    property tx_vc_req_without_tx_outport_req;
        @(posedge clk) disable iff(!rst_n)
        $rose(|tx_outport_req) |-> $rose(|tx_vc_req);
    endproperty
```

```

    assert_tx_vc_req_without_tx_outport_req : assert property(tx_vc_req_without_tx_outport_req)          //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_vc_req high without tx_vc_req tx_outport_req");


    //-----------------------------------
    // tx_vc_gnt is subset of vc_request
    //-----------------------------------
    property tx_vc_gnt_subset_vc_request;
        @(posedge clk) disable iff(!rst_n)
        $rose(|tx_vc_req) |-> (tx_vc_gnt == (tx_vc_gnt & tx_vc_req));
    endproperty

    assert_tx_vc_gnt_subset_vc_request : assert property(tx_vc_gnt_subset_vc_request)                   //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_vc_gnt didn't rise or fall within the tx_vc_req");


    //-----------------------------------
    // no tx_sot without previous tx_vc_gnt
    //-----------------------------------
    property tx_sot_without_tx_vc_gnt(int i);
        @(posedge clk) disable iff(!rst_n)
        $rose(tx_sot[i]) |-> $past(tx_vc_gnt[i]);
    endproperty


    assert_tx_sot_without_tx_vc_gnt : assert property(tx_sot_without_tx_vc_gnt(0))                      //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_sot high when tx_vc_gnt");
```

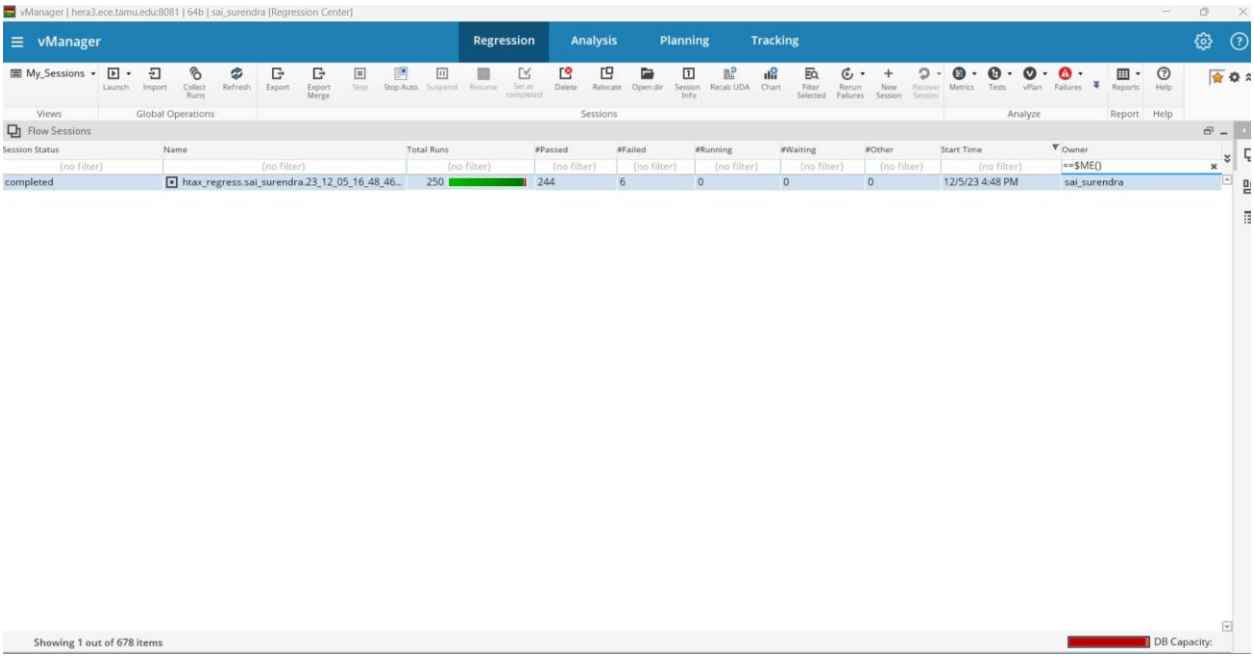
```
    // ------------------------------------
    // no tx_eot without previous tx_vc_gnt
    // ------------------------------------
    property tx_eot_without_previous_tx_vc_gnt;
        @(posedge clk) disable iff(!rst_n)
        $rose(tx_eot) |-> (tx_eot | $past(tx_vc_gnt));
        //@(posedge (|tx_vc_gnt)) $rose(tx_eot) |-> 1;
        //$rose(tx_eot) -> tran_on;
    endproperty

    assert_tx_eot_without_previous_tx_vc_gnt : assert property(tx_eot_without_previous_tx_vc_gnt)          //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_eot high when tx_vc_gnt");


    // ------------------------------------------
    // tx_eot is asserted for a single clock cycle
    // ------------------------------------------
    property tx_eot_single_clk_cycle;
        @(posedge clk) disable iff(!rst_n)
        $rose(tx_eot) |=> $fell(tx_eot) ;
    endproperty

    assert_tx_eot_single_clk_cycle : assert property(tx_eot_single_clk_cycle)                             //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_eot not high for a single clk cycle");


    // ---------------------------------------------------------------
    // tx_release_gnt for pkt(t) one clock cycle or same clock cycle with tx_eot of pkt(t)
    // ---------------------------------------------------------------
    property tx_release_gnt_pkt_tx_eot;
        @(posedge clk) disable iff(!rst_n)
        $rose(tx_release_gnt) |-> ##[0:1] $rose(tx_eot);
    endproperty
```

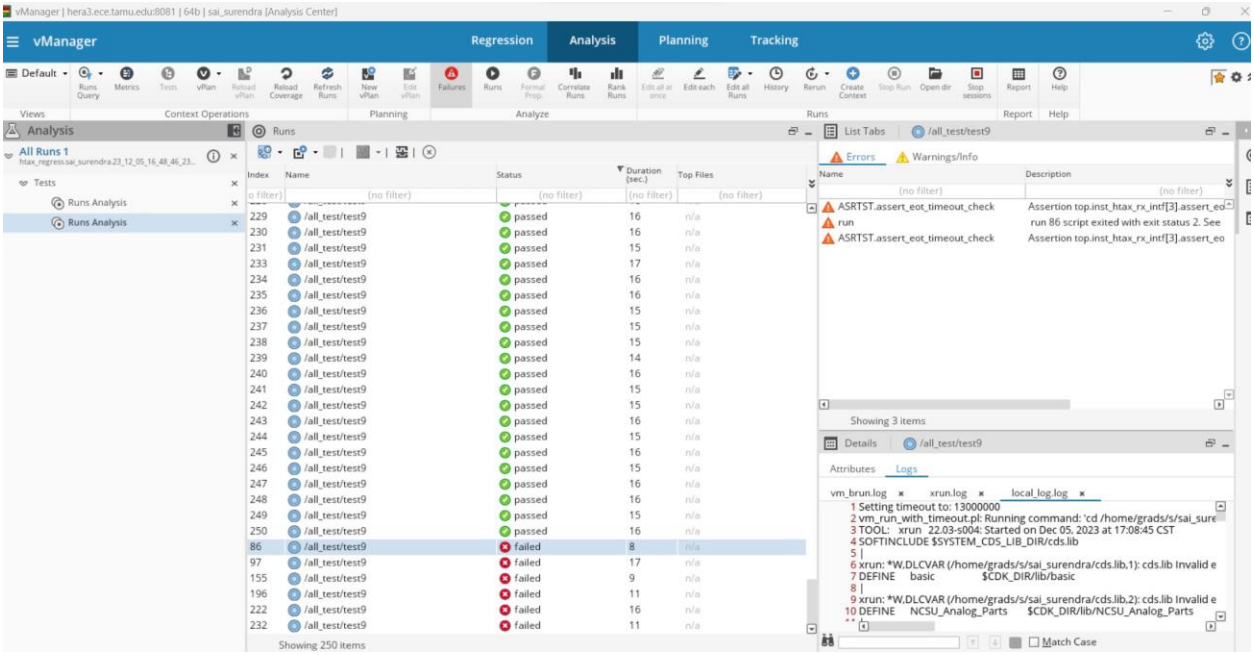
```
    assert_tx_release_gnt_pkt_tx_eot : assert property(tx_release_gnt_pkt_tx_eot)                         //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_release_gnt not high for one cycle for the pkt");


    // ---------------------------------------------------------
    // No tx_sot of p(t+1) without tx_eot for p(t)
    // ---------------------------------------------------------
    property tx_sot_without_tx_eot(int i);
        @(posedge clk) disable iff(!rst_n)
        $rose(|tx_sot[i]) |-> (|tx_sot[i] | $past(tx_eot));
    endproperty

    assert_tx_sot_without_tx_eot : assert property(tx_sot_without_tx_eot(1))                              //assert the property
    else
        $error("HTAX_TX_INF ERROR : tx_sot high without tx_eot");


    // ---------------------------------------------------------
    // Valid packet transfer - rise of tx_outport_req followed by a tx_vc_gnt followed by tx_sot
     // followed by tx_release_gnt followed by tx_eot. Consider the right timings between each event.
    // ---------------------------------------------------------
    property valid_packet_transfer;
        @(posedge clk) disable iff (!rst_n)


        ($past(tx_outport_req) && $past(tx_vc_req) && $past(tx_vc_gnt) && tx_data && $rose(tx_sot)) |->  (tx_data throughout (##[1:64] ##1 tx_release_gnt || tx_eot));
    endproperty

    assert_valid_packet_transfer : assert property(valid_packet_transfer)                                 //assert the property
    else
        $error("HTAX_TX_INF ERROR : Valid Packet is not transferring properly");
endinterface : htax_tx_interface
```

For coverage analysis, added some cover points and cross cover points in htax_rx_monitor_c

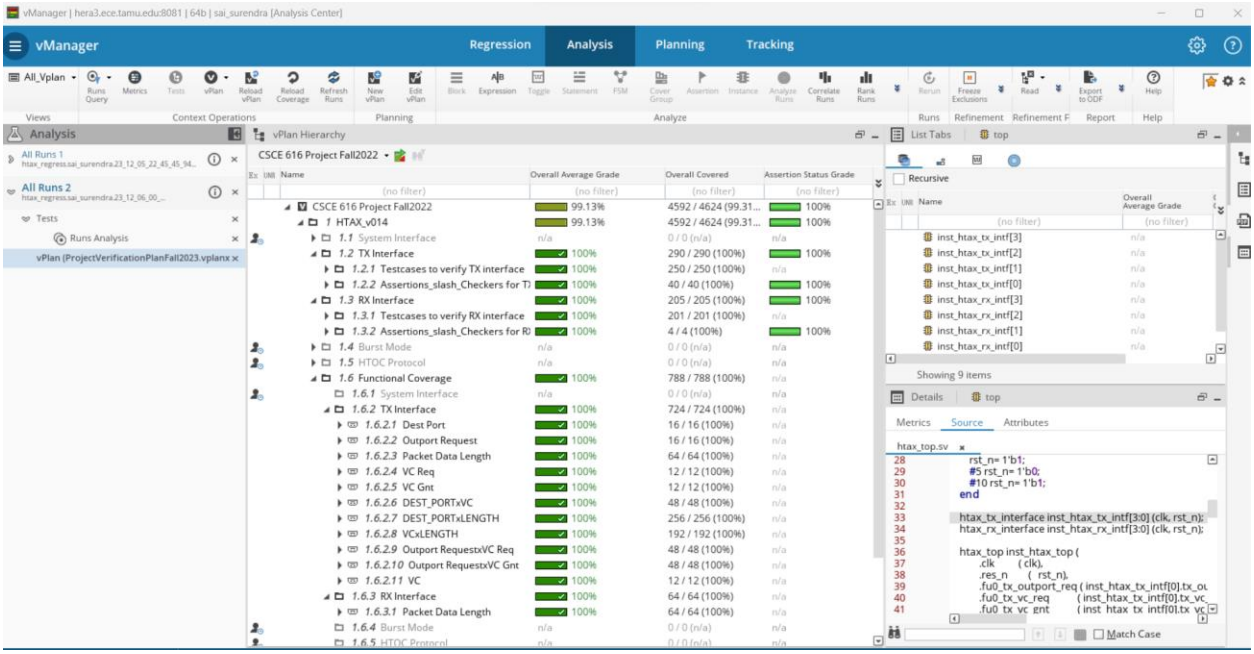
```
10
11      //Analysis port to communicate with Scoreboard
12      uvm_analysis_port #(htax_rx_mon_packet_c) rx_collect_port;
13
14      virtual interface htax_rx_interface htax_rx_intf;
15      htax_rx_mon_packet_c rx_mon_packet;
16      int pkt_len;
17
18      covergroup cover_htax_packet;
19      option.per_instance = 1;
20      option.name = "cover_htax_packet";
21
22
23      //Coverpoint for htax packet field : length
24      LENGTH : coverpoint rx_mon_packet.length  {
25                                                  bins length[16] = {[0:63]};
26                                                                          illegal_bins len = {[0:2]};
27                                                  }
28
29  endgroup
```

The regression test result is given below:





The coverage report is



The code coverage is given below

1.7 Code Coverage	96.5%	3309 / 3341 (99.04...	n/a
1.7.1 Block	96.41%	213 / 221 (96.38%)	n/a
1.7.2 Expression	93.21%	88 / 96 (91.67%)	n/a
1.7.3 Toggle	99.89%	3008 / 3024 (99.47...	n/a
1.7.4 FSM	n/a	0 / 0 (n/a)	n/a

Here the functional coverage and rx assertions are not 100% is due to the assertion failure and the bug. The reason for assertion failure and bug is given in bug report. Please refer to it.

The coverage analysis for bug report is given below: