

# 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

```
17
18 //ASSERTIONS
19
20 // -----
21 // tx_output_req is one-hot
22 // -----
23 property tx_output_req_one_hot;
24   @(posedge clk) disable iff(!rst_n)
25   (|tx_output_req| -> $onehot(tx_output_req));
26 endproperty
27
28 assert_tx_output_req_one_hot : assert property(tx_output_req_one_hot) ..... //assert the property
29 else
30   $error("HTAX_TX_INF ERROR : tx_output request is not one hot encoded");
31
32 // -----
33 // no tx_output_req without tx_vc_req
34 // -----
35 property tx_output_req_without_tx_vc_req;
36   @(posedge clk) disable iff(!rst_n)
37   $rose(|tx_vc_req| -> $rose(|tx_output_req|);
38 endproperty
39
40 assert_tx_output_req_without_tx_vc_req : assert property(tx_output_req_without_tx_vc_req) ..... //assert the property
41 else
42   $error("HTAX_TX_INF ERROR : tx_output_req high without tx_vc_req ");
43
44
45
46 // -----
47 // no tx_vc_req without tx_output_req
48 // -----
49 property tx_vc_req_without_tx_output_req;
50   @(posedge clk) disable iff(!rst_n)
51   $rose(|tx_output_req| -> $rose(|tx_vc_req|);
52 endproperty
53
54 assert_tx_vc_req_without_tx_output_req : assert property(tx_vc_req_without_tx_output_req) ..... //assert the property
55 else
56   $error("HTAX_TX_INF ERROR : tx_vc_req high without tx_vc_req tx_output_req");
57
58
59 // -----
60 // tx_vc_gnt is subset of vc_request
61 // -----
62 property tx_vc_gnt_subset_vc_request;
63   @(posedge clk) disable iff(!rst_n)
64   $rose(|tx_vc_req| -> (tx_vc_gnt == (tx_vc_gnt & tx_vc_req));
65 endproperty
66
67 assert_tx_vc_gnt_subset_vc_request : assert property(tx_vc_gnt_subset_vc_request) ..... //assert the property
68 else
69   $error("HTAX_TX_INF ERROR : tx_vc_gnt didn't rise or fall within the tx_vc_req");
70
71
72 // -----
73 // no tx_sot without previous tx_vc_gnt
74 // -----
75 property tx_sot_without_tx_vc_gnt(int i);
76   @(posedge clk) disable iff(!rst_n)
77   $rose(tx_sot[i]) -> $past(tx_vc_gnt[i]);
78 endproperty
79
80
81 assert_tx_sot_without_tx_vc_gnt : assert property(tx_sot_without_tx_vc_gnt(0)) ..... //assert the property
82 else
83   $error("HTAX_TX_INF ERROR : tx_sot high when tx_vc_gnt");
84
85
86
```

```

87 // -----
88 // no tx_eot without previous tx_vc_gnt
89 // -----
90 property tx_eot_without_previous_tx_vc_gnt;
91   @(posedge clk) disable iff(!rst_n)
92   $rose(tx_eot) |-> (tx_eot | $past(tx_vc_gnt));
93   //@(posedge (|tx_vc_gnt)) $rose(tx_eot) |-> 1;
94   //$rose(tx_eot) |-> tran_on;
95 endproperty
96
97 assert_tx_eot_without_previous_tx_vc_gnt : assert property(tx_eot_without_previous_tx_vc_gnt) .....//assert the property
98 else
99   $error("HTAX_TX_INF ERROR : tx_eot high when tx_vc_gnt");
100
101 // -----
102 // tx_eot is asserted for a single clock cycle
103 // -----
104 property tx_eot_single_clk_cycle;
105   @(posedge clk) disable iff(!rst_n)
106   $rose(tx_eot) |> $fell(tx_eot);
107 endproperty
108
109 assert_tx_eot_single_clk_cycle : assert property(tx_eot_single_clk_cycle) .....//assert the property
110 else
111   $error("HTAX_TX_INF ERROR : tx_eot not high for a single clk cycle");
112
113 // -----
114 // tx_release_gnt for pkt(t) one clock cycle or same clock cycle with tx_eot of pkt(t)
115 // -----
116 property tx_release_gnt_pkt_tx_eot;
117   @(posedge clk) disable iff(!rst_n)
118   $rose(tx_release_gnt) |-> ##[0:1] $rose(tx_eot);
119 endproperty

```

```

122 assert_tx_release_gnt_pkt_tx_eot : assert property(tx_release_gnt_pkt_tx_eot) .....//assert the property
123 else
124   $error("HTAX_TX_INF ERROR : tx_release_gnt not high for one cycle for the pkt");
125
126 // -----
127 // No tx_sot of p(t+1) without tx_eot for p(t)
128 // -----
129 property tx_sot_without_tx_eot(int i);
130   @(posedge clk) disable iff(!rst_n)
131   $rose(|tx_sot[i]) |-> (|tx_sot[i] | $past(tx_eot));
132 endproperty
133
134 assert_tx_sot_without_tx_eot : assert property(tx_sot_without_tx_eot(i)) .....//assert the property
135 else
136   $error("HTAX_TX_INF ERROR : tx_sot high without tx_eot");
137
138 // -----
139 // Valid packet transfer : rise of tx_output_req followed by a tx_vc_gnt followed by tx_sot
140 // followed by tx_release_gnt followed by tx_eot. Consider the right timings between each event.
141 // -----
142 property valid_packet_transfer;
143   @(posedge clk) disable iff (!rst_n)
144   ($past(tx_output_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));
145 endproperty
146
147 assert_valid_packet_transfer : assert property(valid_packet_transfer) .....//assert the property
148 else
149   $error("HTAX_TX_INF ERROR : Valid Packet is not transferring properly");
150 endinterface : htax_tx_interface
151
152
153
154
155
156

```

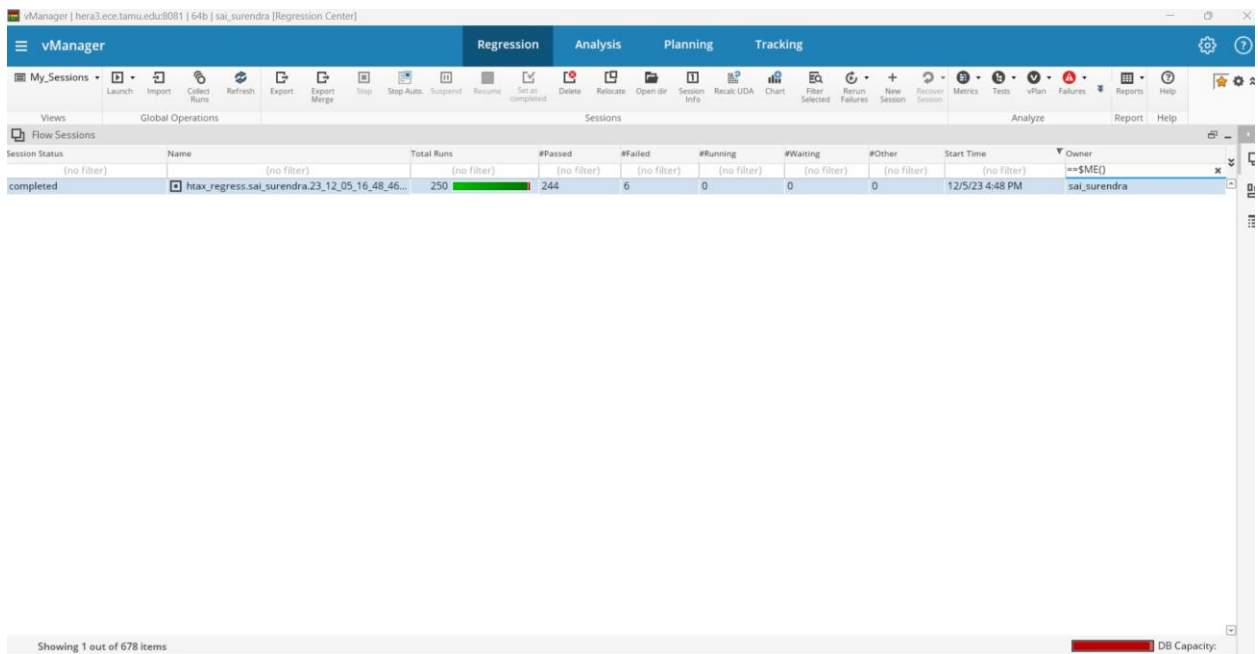
For coverage analysis, added some cover points and cross cover points in htax\_rx\_monitor\_c

```

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
30

```

The regression test result is given below:



The screenshot shows the vManager Analysis tab. The main table displays test results for the test suite `/all_test/test9`. The table has columns: Index, Name, Status, Duration (sec), and Top Files. The status column shows green checkmarks for passed tests and red X marks for failed tests. The right pane shows details for a failed test, including error messages and logs.

Index	Name	Status	Duration (sec)	Top Files
229	/all_test/test9	passed	16	n/a
230	/all_test/test9	passed	16	n/a
231	/all_test/test9	passed	15	n/a
233	/all_test/test9	passed	17	n/a
234	/all_test/test9	passed	16	n/a
235	/all_test/test9	passed	16	n/a
236	/all_test/test9	passed	15	n/a
237	/all_test/test9	passed	15	n/a
238	/all_test/test9	passed	15	n/a
239	/all_test/test9	passed	14	n/a
240	/all_test/test9	passed	16	n/a
241	/all_test/test9	passed	15	n/a
242	/all_test/test9	passed	15	n/a
243	/all_test/test9	passed	16	n/a
244	/all_test/test9	passed	15	n/a
245	/all_test/test9	passed	16	n/a
246	/all_test/test9	passed	16	n/a
247	/all_test/test9	passed	16	n/a
248	/all_test/test9	passed	16	n/a
249	/all_test/test9	passed	15	n/a
250	/all_test/test9	passed	16	n/a
86	/all_test/test9	failed	8	n/a
97	/all_test/test9	failed	17	n/a
155	/all_test/test9	failed	9	n/a
196	/all_test/test9	failed	11	n/a
222	/all_test/test9	failed	16	n/a
232	/all_test/test9	failed	11	n/a

The right pane shows details for a failed test, including error messages and logs. The error message is: `Assertion top_inst_htax_rx_intf[3] assert_eq`. The logs show the test execution details.

The coverage report is

The screenshot shows the vManager Analysis tab. The main table displays code coverage metrics for the test suite `CSCE 616 Project Fall2022`. The table has columns: Ex, Name, Overall Average Grade, Overall Covered, and Assertion Status Grade. The right pane shows details for a specific test case, including code snippets and coverage metrics.

Ex	Name	Overall Average Grade	Overall Covered	Assertion Status Grade
1	HTAX_v014	99.13%	4592 / 4624 (99.31%)	100%
1.1	System Interface	n/a	0 / 0 (n/a)	n/a
1.2	TX Interface	100%	290 / 290 (100%)	100%
1.2.1	Testcases to verify TX interface	100%	250 / 250 (100%)	n/a
1.2.2	Assertions_slash_Checkers for TX	100%	40 / 40 (100%)	100%
1.3	RX Interface	100%	205 / 205 (100%)	100%
1.3.1	Testcases to verify RX interface	100%	201 / 201 (100%)	n/a
1.3.2	Assertions_slash_Checkers for RX	100%	4 / 4 (100%)	100%
1.4	Burst Mode	n/a	0 / 0 (n/a)	n/a
1.5	HTOC Protocol	n/a	0 / 0 (n/a)	n/a
1.6	Functional Coverage	100%	788 / 788 (100%)	n/a
1.6.1	System Interface	n/a	0 / 0 (n/a)	n/a
1.6.2	TX Interface	100%	724 / 724 (100%)	n/a
1.6.2.1	Dest Port	100%	16 / 16 (100%)	n/a
1.6.2.2	Output Request	100%	16 / 16 (100%)	n/a
1.6.2.3	Packet Data Length	100%	64 / 64 (100%)	n/a
1.6.2.4	VC Req	100%	12 / 12 (100%)	n/a
1.6.2.5	VC Gnt	100%	12 / 12 (100%)	n/a
1.6.2.6	DEST_PORT+VC	100%	48 / 48 (100%)	n/a
1.6.2.7	DEST_PORT+LENGTH	100%	256 / 256 (100%)	n/a
1.6.2.8	VC+LENGTH	100%	192 / 192 (100%)	n/a
1.6.2.9	Output Request+VC Req	100%	48 / 48 (100%)	n/a
1.6.2.10	Output Request+VC Gnt	100%	48 / 48 (100%)	n/a
1.6.2.11	VC	100%	12 / 12 (100%)	n/a
1.6.3	RX Interface	100%	64 / 64 (100%)	n/a
1.6.3.1	Packet Data Length	100%	64 / 64 (100%)	n/a
1.6.4	Burst Mode	n/a	0 / 0 (n/a)	n/a
1.6.5	HTOC Protocol	n/a	0 / 0 (n/a)	n/a

The right pane shows details for a specific test case, including code snippets and coverage metrics. The code snippets show the test execution details.

The code coverage is given below

1.7 Code Coverage	<div><div></div></div> 96.5%	3309 / 3341 (99.04...	n/a
1.7.1 Block	<div><div></div></div> 96.41%	213 / 221 (96.38%)	n/a
1.7.2 Expression	<div><div></div></div> 93.21%	88 / 96 (91.67%)	n/a
1.7.3 Toggle	<div><div></div></div> 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:

vManager   hera3.ecn.tamu.edu:8081   64b   sai_surendra [Regression Center]										
vManager										
Regression Analysis Planning Tracking										
My_Sessions Launch Import Collect Runs Refresh Export Export Merge Stop Stop Auto Suspend Resume Set as completed Delete Relocate Open dir Session Info Recall LDA Chart Filter Selected Run Failures New Session Recover Session Metrics Tests vPlan Failures Reports Help										
Views Global Operations Sessions Analyze Report Help										
Flow Sessions										
Session Status	Name	Total Runs	#Passed	#Failed	#Running	#Waiting	#Other	Start Time	Owner	
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	==SMEI	
completed	htax_regress.sai_surendra.23_12_05_22_45_45...	250	<div><div></div></div> 250	0	0	0	0	12/5/23 10:45 PM	sai_surendra	
completed	htax_regress.sai_surendra.23_12_05_16_48_46...	250	<div><div></div></div> 244	6	0	0	0	12/5/23 4:48 PM	sai_surendra	

Showing 2 out of 663 items

DB Capacity: