

Lab 10 report

Coverage closure report:

Functional coverage is 100% for both TX and RX interface

Initially 100% coverage was not achieved, in order to hit 100% coverage additional testcases were written. It can be seen that all possible values for the fields of packet like VC, tx_outport_req, length and dest_port are being covered.

vManager

Regression

Analysis

Planning

Tracking

All_Vplan

Runs Query

Metrics

Tests

vPlan

Reload vPlan

Reload Coverage

Refresh Runs

New vPlan

Edit vPlan

Block

Expression

Toggle

Statement

FSM

Cover Group

Assertion

Instance

Analyze Runs

Views

Context Operations

Planning

Analyze

Analysis

vPlan Hierarchy

All Runs 1
htax_regress.din...

Tests

Runs Analy: x


vPlan (ProjectVeri... x

CSCE 616 Project Fall2022

ISc	Unit	Name	Overall Average Grade	Overall Covered	Assertion Status Grade
		(no filter)	(no filter)	(no filter)	(no filter)
		1.3.2 Assertions_slash_Checkers for RX in	100%	3 / 3 (100%)	0%
		1.4 Burst Mode	0%	0 / 2 (0%)	0%
		1.5 HTOC Protocol	0%	0 / 2 (0%)	0%
		1.6 Functional Coverage	40%	411 / 414 (99.28%)	0%
		1.6.1 System Interface	40%	411 / 414 (100%)	0%
		1.6.2 TX Interface	40%	406 / 406 (100%)	n/a
		1.6.2.1 In port	100%	4 / 4 (100%)	n/a
		1.6.2.2 Output Request	100%	4 / 4 (100%)	n/a
		1.6.2.3 OUTPORT_REQ	100%	4 / 4 (100%)	n/a
		1.6.2.4 X_DEST_PORT_LENGTH	100%	64 / 64 (100%)	n/a
		1.6.2.5 OUTPORT_REQ_1	100%	4 / 4 (100%)	n/a
		1.6.2.6 OUTPORT_REQ_2	100%	4 / 4 (100%)	n/a
		1.6.2.7 X_DEST_PORT_LENGTH_2	100%	64 / 64 (100%)	n/a
		1.6.2.8 X_DEST_PORT_VC_2	100%	12 / 12 (100%)	n/a
		1.6.2.9 LENGTH_2	100%	16 / 16 (100%)	n/a
		1.6.2.10 DEST_PORT_2	100%	4 / 4 (100%)	n/a
		1.6.2.11 X_DEST_PORT_LENGTH_1	100%	64 / 64 (100%)	n/a
		1.6.2.12 X_DEST_PORT_VC_1	100%	12 / 12 (100%)	n/a
		1.6.2.13 LENGTH_1	100%	16 / 16 (100%)	n/a
		1.6.2.14 VC_1	100%	3 / 3 (100%)	n/a
		1.6.2.15 DEST_PORT_1	100%	4 / 4 (100%)	n/a
		1.6.2.16 X_DEST_PORT_VC	100%	12 / 12 (100%)	n/a
		1.6.2.17 LENGTH	100%	16 / 16 (100%)	n/a
		1.6.2.18 VC	100%	3 / 3 (100%)	n/a
		1.6.2.19 DEST_PORT	100%	4 / 4 (100%)	n/a
		1.6.2.20 Packet Data Length	100%	16 / 16 (100%)	n/a
		1.6.2.21 First Added TX interface cove	100%	12 / 12 (100%)	n/a
		1.6.2.22 Second added TX interface cc	100%	64 / 64 (100%)	n/a
		1.6.3 RX Interface	100%	5 / 5 (100%)	n/a
		1.6.3.1 First Added RX interface cover	100%	1 / 1 (100%)	n/a
		1.6.3.2 VC_GNT	100%	1 / 1 (100%)	n/a
		1.6.3.3 VC_GNT_1	100%	1 / 1 (100%)	n/a
		1.6.3.4 VC_GNT_2	100%	1 / 1 (100%)	n/a
		1.6.3.5 VC_GNT_3	100%	1 / 1 (100%)	n/a
		1.6.4 Burst Mode	0%	0 / 1 (0%)	0%
		1.6.5 HTOC Protocol	0%	0 / 1 (0%)	0%

Code coverage:

From the below screenshot, it can be seen that the code coverage is above 95% for block, toggle, expression, and FSM.

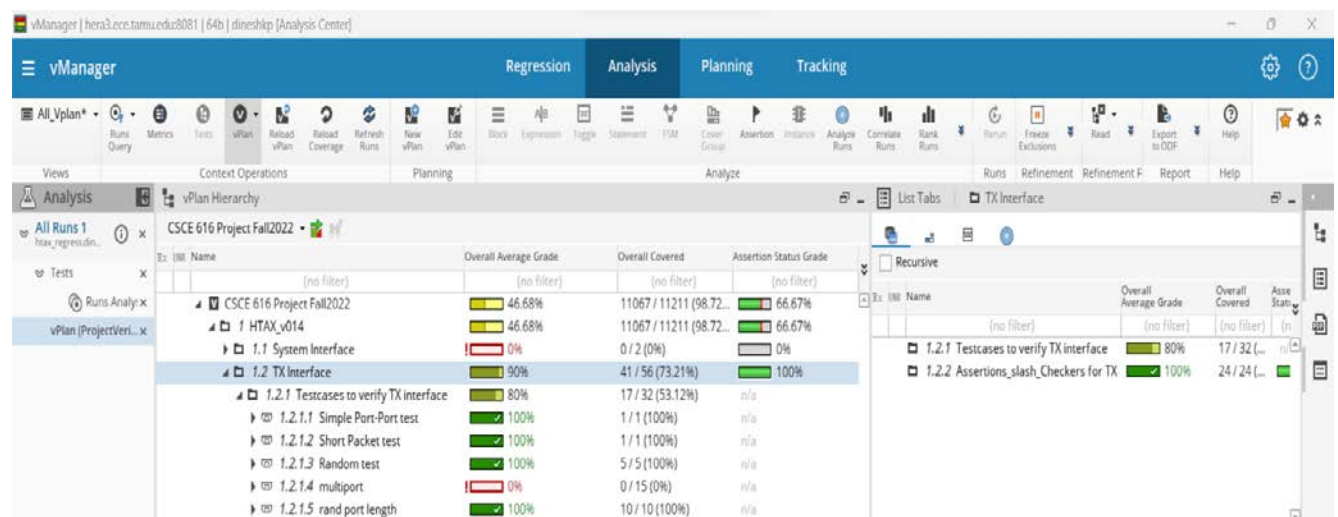


1.7 Code Coverage	96.78%	10596 / 10716 (98.88%)	n/a
1.7.1 Block	96.78%	2649 / 2679 (98.88%)	n/a
1.7.2 Expression	96.78%	2649 / 2679 (98.88%)	n/a
1.7.3 Toggle	96.78%	2649 / 2679 (98.88%)	n/a
1.7.4 FSM	96.78%	2649 / 2679 (98.88%)	n/a

Bug report:

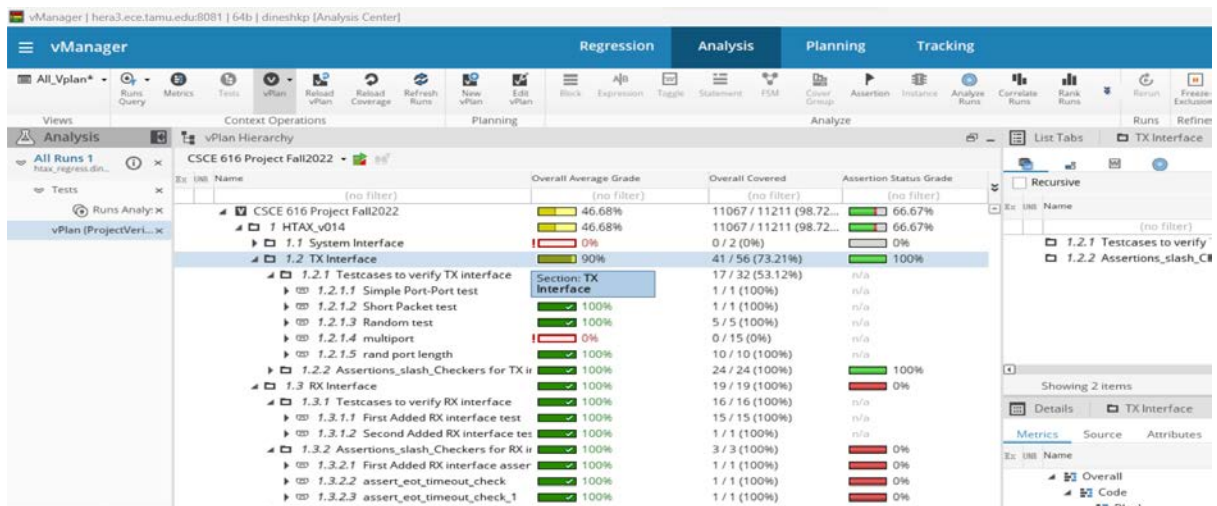
Additional tests were included to identify the bug. The htax_random_test file contains the testcases that were used to identify the bug in the design. In this, we generate packets to all the 4 ports parallel using fork-join. Additional constraints were written such that all dest_ports are also covered. Also, packet length and delay has to be kept constant for all 4 packets to hit this bug.

Testcases for TX and RX interface in Vmanager:

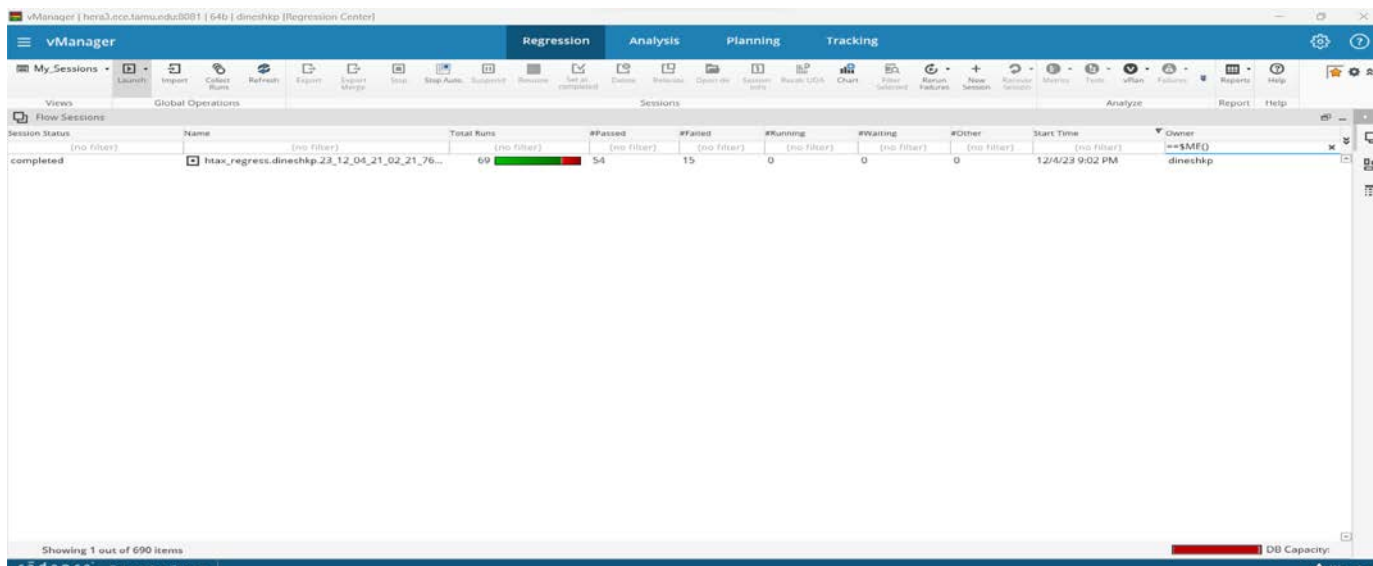


Ex	Item Name	Overall Average Grade	Overall Covered	Assertion Status Grade
	CSCE 616 Project Fall2022	46.68%	11067 / 11211 (98.72%)	66.67%
	1 HTAX_v014	46.68%	11067 / 11211 (98.72%)	66.67%
	1.1 System Interface	0%	0 / 2 (0%)	0%
	1.2 TX Interface	90%	41 / 56 (73.21%)	100%
	1.2.1 Testcases to verify TX interface	80%	17 / 32 (53.12%)	n/a
	1.2.1.1 Simple Port-Port test	100%	1 / 1 (100%)	n/a
	1.2.1.2 Short Packet test	100%	1 / 1 (100%)	n/a
	1.2.1.3 Random test	100%	5 / 5 (100%)	n/a
	1.2.1.4 multiport	0%	0 / 15 (0%)	n/a
	1.2.1.5 rand port length	100%	10 / 10 (100%)	n/a

Ex	Item Name	Overall Average Grade	Overall Covered	Assertion Status Grade
	1.2.1 Testcases to verify TX interface	80%	17 / 32 (53.12%)	n/a
	1.2.2 Assertions_slash_Checkers for TX	100%	24 / 24 (100%)	n/a



In the above figure, only the testcase that contains the bug tests fails.



The test that includes the testcases for finding the bug has been called 15 times. So these tests 15 fails and other tests pass.

The above test is executed using the irun command and the assertion fails.

```
xmsim: *F,ASRTST (../tb/htax_rx_interface.sv,56): (time 20430 NS) Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed
Memory Usage - Current physical: 91.2M, Current virtual: 175.1M
CPU Usage - 0.2s system + 0.2s user = 0.4s total (62.7% cpu)
Simulation terminated via $fatal(2) at time 20430 NS + 2
../tb/htax_rx_interface.sv:56 $fatal("HTAX_RX_INF ERROR : TIMEOUT rx_eot did not occur within 1000 cycles after rx_sot");
xcelium> exit

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

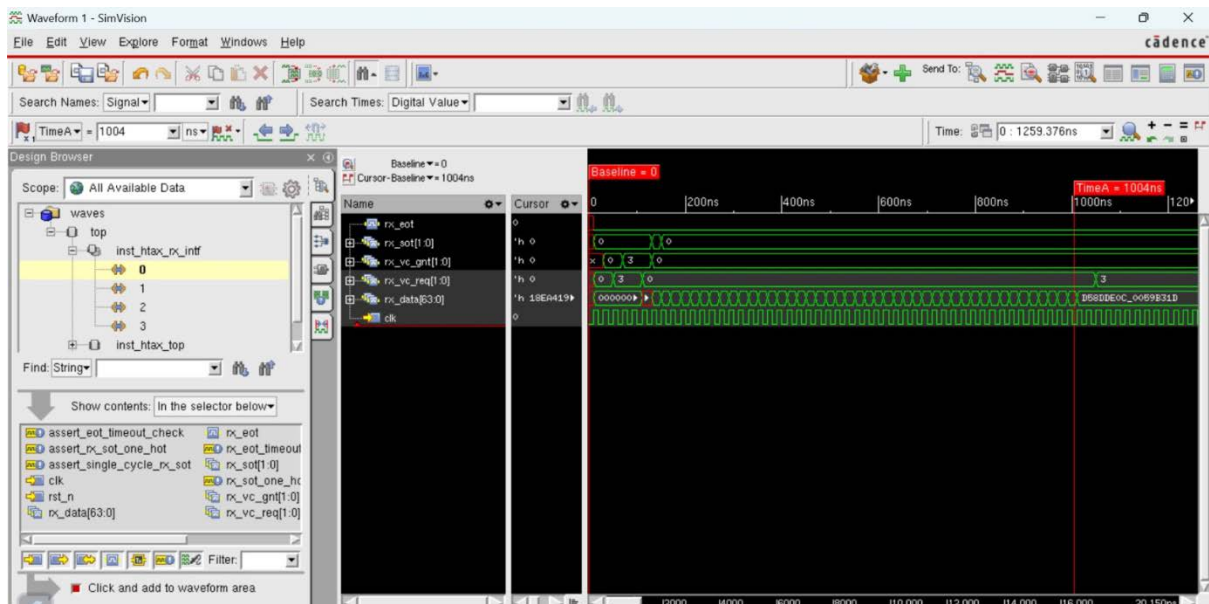
coverage files:
model(design data) : ./cov_work/scope/icc_4e8e3c4e_7afba4d.ucm (reused)
data : ./cov_work/scope/test/icc_4e8e3c4e_7afba4d.ucd
TOOL: xrun 22.03-s004: Exiting on Dec 04, 2023 at 21:21:09 CST (total: 00:00:07)
```

Bug in the code:

When packets are generated in all ports the eot_in become 1. The complement of it produces a 0 and thus the AND with it makes eot always 0.

```
always @( * )
begin
    (* full_case *) (* parallel_case *)
    casex (inport_sel)
        4'b1xxx: selected_sot = sot_in[((4*VC)-1):(3*VC)];
        4'b01xx: selected_sot = sot_in[((3*VC)-1):(2*VC)];
        4'b001x: selected_sot = sot_in[((2*VC)-1):(1*VC)];
        4'b0001: selected_sot = sot_in[((1*VC)-1):(0*VC)];
    endcase
    assign selected_eot = |(eot_in & inport_sel_reg) & ~(eot_in));
end
```

Waveform with eot not asserted:



From the above waveform, it is clear that the rx_eot signal is not asserted for any packet that is transmitted. But, it is supposed to be asserted when the final packet is transmitted for each transaction. Tracing this bug in the design, it was found that a NOT of bitwise AND was performed on eot_in.

Due to this, whenever all the ports are accessed, all the bits become high and the bitwise AND becomes 1. Hence, a complement of this will give a 0. So the EOT signal is 0 at all times.

To fix this, we can remove the bitwise AND part and this fixes the bug.

Fixed code:

```

always @( * )
begin
    (* full_case *) (* parallel_case *)
    casex (inport_sel)
        4'b1xxx: selected_sot = sot_in[((4*VC)-1):(3*VC)];
        4'bx1xx: selected_sot = sot_in[((3*VC)-1):(2*VC)];
        4'bxx1x: selected_sot = sot_in[((2*VC)-1):(1*VC)];
        4'bxxx1: selected_sot = sot_in[((1*VC)-1):(0*VC)];
    endcase
end

assign selected_eot = |(eot_in & inport_sel_reg);

```

After fixing the code the UVM fatal does not occur:

```

--- 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 : 136
UVM_WARNING : 0
UVM_ERROR : 0
UVM_FATAL : 0
** Report counts by id
[END]

```

EOT gets asserted in the waveform after fixing the bug:

