

Lab 10

HTAX Regression

Reminder:

Please take a few minutes of your time to evaluate the course anonymously on <https://tamu.aefis.net/> - Your feedback is valuable.

Note: the deadline to submit your course evaluation is **December 4th, 2024**.

Due date

Dec 2, 2024 11:59 PM

Table of content

Academic Integrity.....	2
Introduction.....	2
Design Under Test.....	2
Environment Setup.....	2
To-do.....	3
Deliverables.....	4

Academic Integrity


The following actions are strictly prohibited and violate the honor code. The minimum penalty for plagiarism is a grade of zero and a report to the Aggie honor system office.

- Sharing your solutions with a classmate.
- Uploading assignments to external websites or tutoring websites
- Copying solutions from external websites or tutoring websites
- Copying code from a classmate or unauthorized sources and submitting it as yours

Introduction

In this lab, you will make use of the knowledge you gained from the previous labs to run regression to close coverage in your testbench and identify the bug in the design.

Design Under Test

The DUT is the HyperTransport Advanced X-Bar whose specifications are mentioned in this document:  HyperTransport Advanced X-Bar HTAX Specification.pdf .

Environment Setup

1. Accept the assignment's repository on GitHub Classroom:

<https://classroom.github.com/a/ShXRrPCp>

2. Source the setup file.

```
source setupX.bash
cd work
```

3. We have now a full UVM-TB for HTAX design in place. Additional files:

tb/htax_virtual_sequencer_c.sv - Virtual Sequencer for HTAX Env., contains 4 instance of

HTAX Sequencer

tb/htax_vseqs.sv - Contains base Virtual Sequence

test/multiport_sequential_random_test.sv - This test executes a virtual sequence "simple_random_vsequence" which executes 10 TXNs on ports {0,1,2,3} randomly

.

4. Adding More Tests: Before you kick off regression, add new tests into your test directory. Use test/multiport_sequential_random_test.sv as template.

- Include the virtual sequence class in the test file itself, after your test class.
- Use “uvm_do_on_with” to add constraints to the req (htax_packet_c instance).
- p_sequencer.htax_seqr[<i>] will initiate the transaction on port <i>. Replace <i> with any of {0,1,2,3}.
- Use fork-join to initiate multiple transactions in parallel
- Include your test in test library file - test/test_lib.svh .

5. Including Tests for Regression: Open sim/run_vm.vsif. Include the tests under the “Add Tests Here” marker. Use the template provided in the file. Change count to run a test multiple number of times.

6. Run Regression: Open Vmanager and launch the run_vm.vsif. Follow the instructions provided in [vManager Setup Fall2023 manual](#).

7. Coverage Closure: Map the final coverage of the regression with your v-plan in Vmanager.

8. Finding Bug: Find and fix bug in HTAX design. You should hit this bug if you run a sufficient

number of random Long and Short packets. (Consider all possibilities, play around with your tests and sequences, create new tests/sequences).

To-do

1. Add new tests in your tests directory and run the regression. Find the bug.
2. Prepare the coverage closure report. Requirements:

- 2.1. DUT code coverage screenshots (block coverage should be 95% closure, expression - 95%, toggle - 90%)
 - 2.2. Functional coverage (100% closure) individual screenshots for
 - 2.2.1. Fields of packet like vc, length, dest_port etc. Consider their cross and transition coverage.
 - 2.2.2. tx_outport_req has covered all the values 0001,0010,0100,1000 at least once
 - 2.2.3. All the virtual channels are requested at least once. Ignore what is not allowed, or put it as illegal.
 - 2.2.4. All the virtual channels are granted at least once.
 - 2.2.5. Optional - Additionally, add all the cover-groups/cover-points relevant to the HTAX design as per your v-plan.
 - 2.3. Clear explanation of any Code Coverage/ Functional Coverage Holes.
3. Prepare a Bug report which contains the following:
 - 3.1. Cadence xrun log (xrun.log) showing the failing case and resulting checker errors, and overall vManager regression pass/fail rate
 - 3.2. Clear explanation of how you debugged the failure and isolated the bad DUT code (including waveforms used to debug)
 - 3.3. Recommended fix for the bad DUT code
4. Prepare a lab report that includes all the code you have written, UVM summary/simulation output that shows that there are no UVM_FATAL/UVM_ERROR/Assertion failures. Attach your Coverage Closure report and Bug report to this. Name it lab_report.pdf.

(Note. Format your report properly with proper titles and labels to gain maximum marks)

Deliverables

Commit and push all your changes to your remote repository.

Your repository must include the following:

- The design directory **(DO NOT COMMIT THE DESIGN WITH THE BUG FIX)**
- The test directory

- The sim directory containing lab_report.pdf (Format is already present in the sim directory)
- The tb directory containing updated files.

Important note: To get full credit, you must upload all the required files and directories and strictly name your files according to the requirements.