Lab 7

HTAX TX/RX interface and SV Assertions

Due date

Nov 4, 2024 11:59 PM CST

Table of content

Academic Integrity........ 2
Introduction........ 2
Design Under Test........ 2
Environment Setup........ 2
To-do........ 5
Deliverables........ 6

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 your

Introduction

In this lab, you wil learn to write UVM's Driver and Monitor logic, gaining skills in designing UVM Drivers and monitoring transactions, all within the UVM framework.

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/JQLI2H2I

2. Source the setup file.

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

3. Similar to previous lab we have partial UVM-TB for HTAX design. Below is list of files in lab7/ directory with brief description

● *test/test_lib.svh* –	UVM Test library
● *test/base_test.sv* –	Base test template
● *test/simple_random_test.sv* –	Kicks of simple random sequence on port [1] sequencer
● *tb/htax_defines.sv* –	Design related defines
● *tb/htax_pkg.sv* -	Include UVM components and object related files
● *tb/htax_top.sv* -	TB Top module
● *tb/htax_packet_c.sv* -	HTAX packet class
● *tb/htax_tx_mon_packet_c.sv* -	HTAX TX Monitor packet class (NEW)
● *tb/htax_seqs.sv* -	Base sequence and simple random sequence code

- *tb/htax_env.sv* - UVM Environment (only UVM agent instantiated)
- *tb/htax_tx_agent_c.sv* - TX Agent (sequencer and driver instantiated)
- *tb/htax_tx_driver_c.sv* - We'll write the code in this lab
- *tb/htax_tx_monitor_c.sv* - We'll write the code in this lab (NEW)
- *tb/htax_sequencer_c.sv* - TX Sequencer code
- *tb/htax_tx_interface.sv* - TX Interface and SV Assertions
- *tb/htax_rx_interface.sv* - RX Interface and SV Assertions

3. Driver Code: Open tb/htax_tx_driver_c.sv. In run_phase() we initiate below tasks,

```
//UVM build phase
task run_phase(uvm_phase phase);
  rst_dut_and_signal();                //Reset the interface signals
  forever begin                                    //Till end of test
        seq_item_port.get_next_item(req);//Get next pkt from sequencer
        drive_thru_dut(req);           //Drive pkt to DUT via interface
        seq_item_port.item_done(); //Sends ACK to sequencer
  end
 endtask : run_phase
```

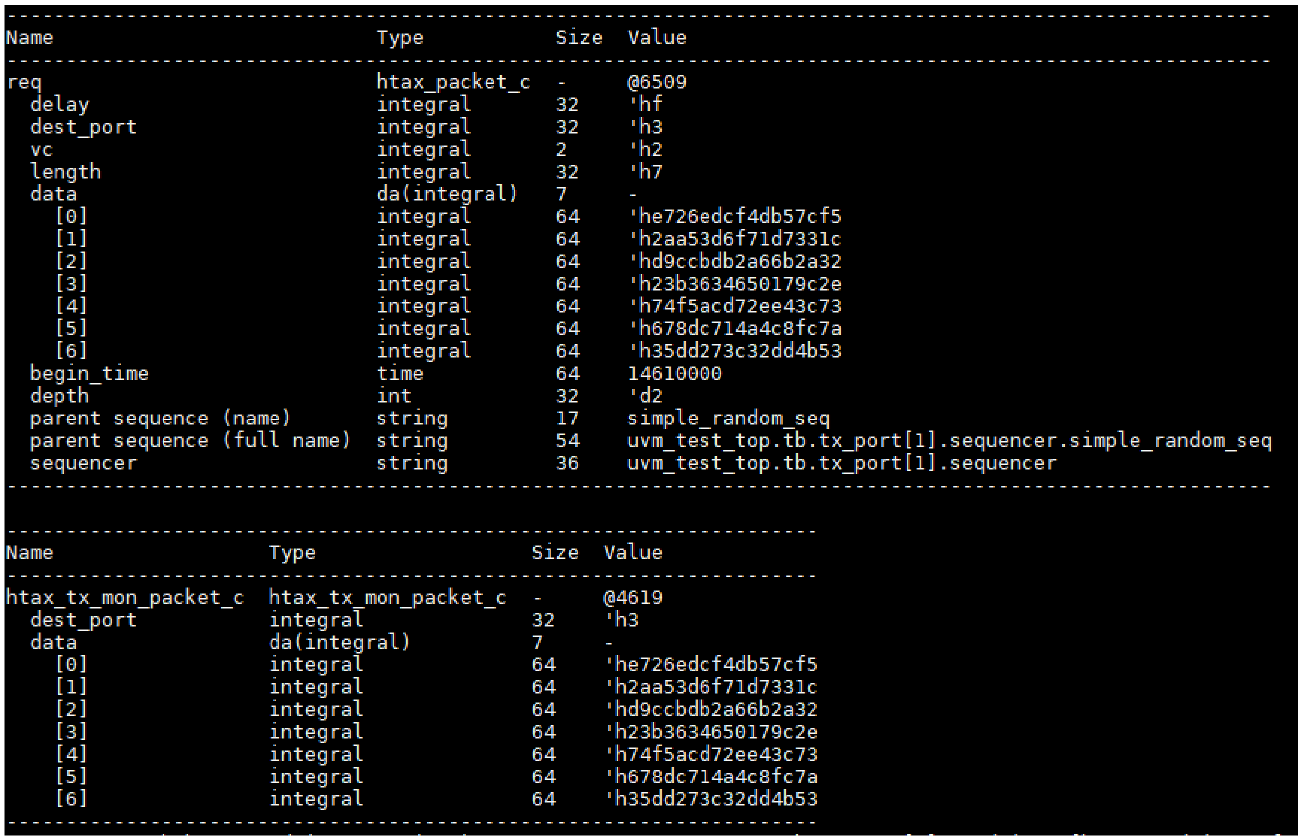
4. Monitor Code: Open tb/htax_tx_monitor_c.sv. You need to complete the TO DOs in the task run_phase. Simple random test: Run this test using below command:

```
cd sim
xrun -f run.f +UVM_TESTNAME=simple_random_test
```

The test is simulated successfully if there are no UVM_FATAL, UVM_ERROR and no Assertion failures with signature "ncsim: *E,ASRTST" in the summary at the end of simulation.

```
** Report counts by severity
UVM_INFO :   42
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
```

5 . HTAX Packet and HTAX TX Monitor Packet is printed in log. *Make sure dest_port and data match for both of them*.

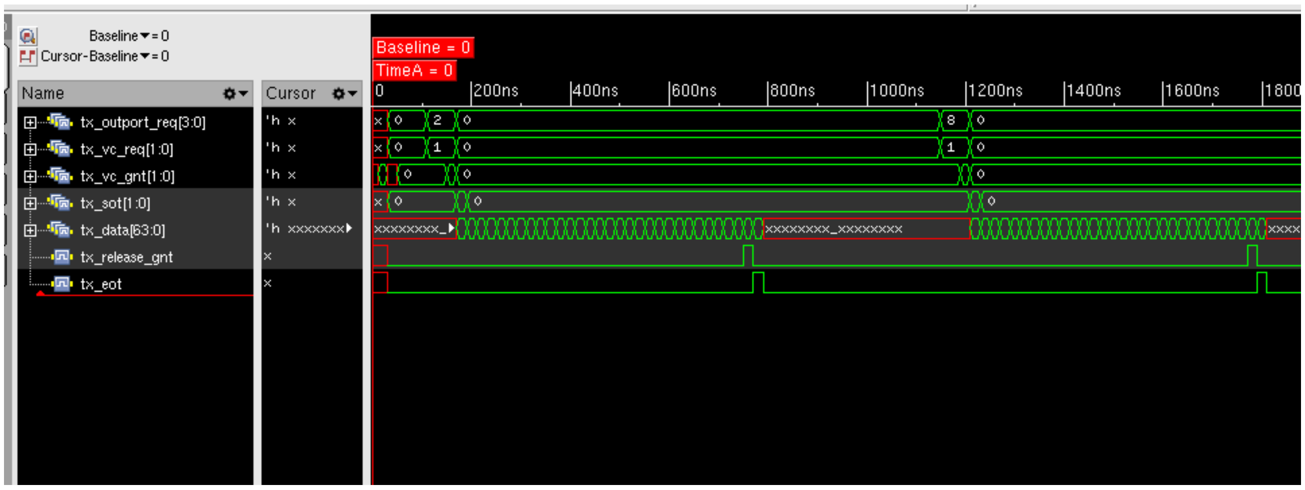

```
------------------------------------------------------------------------------------------------------
Name                           Type           Size  Value
------------------------------------------------------------------------------------------------------
req                            htax_packet_c  -     @6509
  delay                        integral       32    'hf
  dest_port                    integral       32    'h3
  vc                           integral       2     'h2
  length                       integral       32    'h7
  data                         da(integral)   7     -
    [0]                        integral       64    'he726edcf4db57cf5
    [1]                        integral       64    'h2aa53d6f71d7331c
    [2]                        integral       64    'hd9ccbdb2a66b2a32
    [3]                        integral       64    'h23b3634650179c2e
    [4]                        integral       64    'h74f5acd72ee43c73
    [5]                        integral       64    'h678dc714a4c8fc7a
    [6]                        integral       64    'h35dd273c32dd4b53
  begin_time                   time           64    14610000
  depth                        int            32    'd2
  parent sequence (name)       string         17    simple_random_seq
  parent sequence (full name)  string         54    uvm_test_top.tb.tx_port[1].sequencer.simple_random_seq
  sequencer                    string         36    uvm_test_top.tb.tx_port[1].sequencer
------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
Name                  Type                  Size  Value
-------------------------------------------------------------------
htax_tx_mon_packet_c  htax_tx_mon_packet_c  -     @4619
  dest_port           integral              32    'h3
  data                da(integral)          7     -
    [0]               integral              64    'he726edcf4db57cf5
    [1]               integral              64    'h2aa53d6f71d7331c
    [2]               integral              64    'hd9ccbdb2a66b2a32
    [3]               integral              64    'h23b3634650179c2e
    [4]               integral              64    'h74f5acd72ee43c73
    [5]               integral              64    'h678dc714a4c8fc7a
    [6]               integral              64    'h35dd273c32dd4b53
-------------------------------------------------------------------
```

6 . Open Waveform: open the waveform database using below command

```
simvision waves.shm/ &
```

7 . Send all htax_tx_intf signals to waveform window. You can see the waveform of all 15 TXN in the waveform window. Compare each signal's behavior against HTAX specification.



8 . Make sure no assertions are failing at the end of simulation. The assertion failure suggests that there is/are bug(s) in driver code which needs to be fix.

To-do

1. In this lab we have provided you the code for rst_dut_and_signal() in the driver . You need to complete the TO DOs in the task drive_thru_dut(req). Hints listed below,
   - 1.1. Set htax_tx_intf.tx_outport_req in one-hot fashion from pkt.dest_port
   - 1.2. Assign htax_tx_intf.tx_vc_req from pkt.vc
   - 1.3. Wait till htax_tx_intf.tx_vc_gnt is received
   - 1.4. Set any one bit tx_sot[vc-1:0] to 1 from the ones granted by htax_tx_intf.tx_vc_gnt
   - 1.5. Drive pkt.data[0] on htax_tx_intf.tx_data
   - 1.6. Reset htax_tx_intf.tx_outport_req and htax_tx_intf.tx_vc_req (to zero)
   - 1.7. On consecutive clk-posedges drive each of the packet's pkt.data on htax_tx_intf.tx_data
   - 1.8. Assign htax_tx_intf.tx_sot to zero after first cycle
   - 1.9. Assert htax_tx_intf.tx_release_gnt for one clock cycle when driving second last data packet
   - 1.10. Assert htax_tx_intf.tx_eot for one clock cycle when driving last data packet
   - 1.11. Replace XXX and YYY with appropriate values for a packet
   - 1.12. Assign htax_tx_intf.tx_data to X and htax_tx_intf.tx_eot to zero

2. In the monitor, complete the following tasks

    2.1. Assign tx_mon_packet.dest_port from htax_tx_intf.tx_outport_req
    2.2. On consequtive cycles append htax_tx_intf.tx_data to tx_mon_packet.data[] till htax_tx_intf.tx_eot pulse
    2.3. Replace XXX with appropriate condition in while loop

3. Make a lab report that includes your driver and monitor code, htax_tx_intf signals waveform (shown in step 7) , simulation output that shows that there are no UVM_FATAL/UVM_ERROR/Assertion failures and HTAX Packet / HTAX TX Monitor Packet (like it is shown in step 5) and name it lab_report.pdf.

Note: We will write the TX driver code for “Single Transmit Request” scenario only. For this course we won’t verify the design for “Multiple Transmit Request” or “MLF”.

Deliverables

Commit and push all your changes to your remote repository.

Your repository must include the following:

- The test directory
- The sim directory containing lab_report.pdf
- 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.