Out: November 25th, 2015
Due: December 6th, 2015

Goal

To gain experience building a module component to complete the automated checking capabilities of a testbench, then adding practical functional coverage and analyzing coverage results from a random packet generation test.

Procedure

1. This lab package includes the "phase 2b" SBIU testbench that we are developing in class (sbiu_phase_2b.zip). Extract it to a convenient location for reference. The design of the PDM testbench closely parallels the SBIU testbench, so you can leverage the latter as you work on this lab.

2. Extract the lab 6 PDM testbench shell (pdm_tb.zip) to your working folder. The usual .do files have been provided for your convenience:

    a. compile_all.do – Compiles the DUT and all of your testbench files, excluding the testcase file. Run this script every time you make a change to your testbench code that you want to simulate. NOTE - before you compile for the first time, issue the following command: vlib work

    b. run_sanity_test.do – Compiles the sanity test, invokes the simulator, and brings up a wave trace window in the same format as shown in the PDM specification so you can easily verify that your testbench is working properly. Run this script every time you change sanity_test.sv or want to run/re-run the sanity test.

    c. run_rand_test.do – Same as in b), but for the random test (rand_test.sv).

3. Complete the PDM module component (pdm_tb/pdm_module/pdm_module.sv). Follow the "TO DO" comments as a guide to the tasks and functions that you need to create. The key protocol checks to implement are:

    a. Check that the destination port of the input TR matches the port of the output TR (confirming that packet routing was correct).

    b. Check that the payload in the input TR matches the payload in the output TR (confirming that no internal data corruption occurred).

    c. Check that the value on newdata_len_X correctly reflects the size of the packet that was driven into the input interface.

4. All of the components, including the pdm_module, are already instantiated in the testbench env. Run the sanity test and confirm that your module component is active, producing records, and not reporting any errors. Then, uncomment one-at-a-time each of the three

error injection routines in the sanity test and re-run the test for each. Confirm that your module component can correctly detect and report each injected error condition.

5. Add functional coverage groups, with associated coverpoints, to the module component. Instantiate and sample the coverage groups. Again, the "TO DO" comments guide you through the specific code to write. Use the sbiu_module.sv and the functional coverage lecture notes for reference as needed.

6. Load the random test (run_rand_test.do). Before running the sim, open the Cover Groups browser (View -> Coverage -> Covergroups). Browse through the coverage groups and their constituent coverage items that you created. Confirm that the names, number of bins, etc. reflect what you were asked to implement.

7. The random test generates a number of random packets with constraints to ensure that they are generated with valid destination ports and payload sizes. Run the test until all packets have been driven then view your functional coverage results.

Deliverable

A compressed archive (.zip or .tar.gz) of your completed lab 6 testbench environment. Include your name, NSID, and student number in the filename, and email to the course lab instructor Chandler Janzen (chandler.janzen@usask.ca) by midnight on December 6th.