The University of Alabama in Huntsville, Dept. of Electrical & Computer Engineering

CPE 426/526 Spring 2022

Homework #4

Due April 10, 2022

Based on the previous work (Spring 2021; arbiter.vhd), take the provided VHDL model of the newly updated "prio_enc_arb.vhd" and its submodule "prio_enc_recur.vhd" which performs a priority encoding of the bit vector at its input (choosing the least significant '1' in the vector and reporting its location). Think of prio_enc_arb as an upgrade to arbiter, and the assignment is to use last year's solution as a template for how to solve this homework, with a few noted exceptions.

The template includes an interface for the arbiter, a "Packet" class random object defined within the package "mine", 4 test modules (one for each request device) and one test program that randomly generates resets for the system. This time, the assignment expands the test to 8 instances of request devices with 8 accompanying test modules/programs. Also, due to the nature of the upgrade, the constraints for the random variables must be changed as follows:

1. Number of cycles consumed before a device generates a request: 101 – 500
2. Number of cycles a request is kept high: 1 – 40
3. Number of cycles consumed before a reset is generated: 501 - 800
4. Number of cycles the reset is asserted: 4 – 7

This year, instead of calling p.randomize(); multiple times in the testbench to ensure different results from different driver instances, seeds will be used. For the 8 different instances, use seed=1 through seed=8. Prior to the "repeat(200)" statement, use p.srandom(seed); and use only one p.randomize(); call within the repeat block. **ALSO:** since this is an arbiter, we should set up an assertion to verify that the GRANT signal in the arb_if interface has either 0 or 1 bit high at most across all 8 bits of GRANT. Please add the property "onehot" according to the example on Slide 42 of the SV lecture deck and an assertion to monitor that property.

Compile all your files with all forms of coverage enabled. Optimize using the command vopt top -o opttop +cover=sbecf from the transcript window, where top is your top-level module. Run 200 different test cases for each test program. Generate a coverage report for all types of coverage using Tools -> Coverage Report -> Text. Select Details and all kinds of Coverage. Additional notes are as follows:

It is recommended to create a project in QuestaSim, adding the above files to the project. Prior to compiling (each file, or altogether), right-click the file names and select "Compile Properties". On the Coverage tab, check all the Source code coverage boxes, the Enable 0/1/Z Toggle Coverage radio button, Optimization level 4, and all other coverage checkboxes (I hit enter since the OK button somehow didn't register).

After doing the above, please run "vopt top -o opttop +cover=sbecf" on the Transcript console. Finally, do Start Simulation and select work>opttop to simulate, and also go to the Other table and check "Enable code coverage". (Or instead of Start Simulation, equivalently type in "vsim opttop -coverage"

Turn in all your SystemVerilog source files and your coverage report file (mine was 1215 lines of text).