Embedded Systems Modeling and Design
ECPS 203
Fall 2017

Assignment 3

Posted: October 18, 2017
Due: October 25, 2017 at 6pm

Topic: Introduction to SystemC language and simulation

1. Setup:

We will use the same Linux account and the same remote servers as for the previous assignments. A current version of the open-source SystemC library from the Accellera Systems Initiative has been installed on the servers in this directory:

```
/opt/pkg/systemc-2.3.1/
```

For this assignment, create a new working directory, so that you can properly submit your deliverables in the end.

```
mkdir hw3
cd hw3
```

For your convenience, we also provide a **`Makefile`** suitable for this assignment that you should copy into your working directory.

```
cp ~ecps203/public/Makefile .
```

2. Capture and simulate the introductory example provided by Doulos

We will use the introductory example from Doulos discussed in Lecture 6 as an initial SystemC model. Please refer to the posted presentation slides as reference:

`https://eee.uci.edu/17f/16905/DAC15_SystemC_Training.pdf`

Step 1: Structural model

Review and study the structural model shown on slide 25. Note that there are three modules instantiated inside the **`Top`** module, namely the stimulus **`Stim`**, the design-under-test (aka. unit-under-test) **`Mult`**, and the monitor **`Mon`**. The modules are connected by SystemC signals.

We will capture the source codes for all the modules and then simulate the entire SystemC model in this assignment.

Step 2: Source file structure

We will implement the same source file structure as shown on slide 32. While the header file **`systemc.h`** is provided by the SystemC installation, you will need to capture the other eight source files by use of your favorite text editor. Make sure you use exactly the filenames indicated in the slide. Otherwise the provided **`Makefile`** will not work correctly.

Step 3: Capture the partial source code provided by Doulos

Review the slides 21 through 36 and capture the relevant source code shown on the slides into the corresponding source code files. You may want to use the cut-and-paste functionalities of your web browser and text editor for this task. Next, apply any needed adjustments manually.

Step 4: Adjust and complete the stimulus and monitor modules

The source code for a simple stimulus module is shown on slide 36. Capture this source code into the corresponding **`stim.cpp`** and **`stim.h`** files. Then adjust and extend the listed test vectors so that the module sends out new test values which let the connected multiplier module compute the following 7 multiplications as test cases for this assignment: 1*6, 2*6, ..., 7*6.

Next, design and specify the monitor module. Note that there is no source code provided for it in the slides.

The desired monitor module should look very similar to the stimulus module. Adjust the ports so that they match the structural model from slide 25. Next, make the monitor sensitive to the falling edge of the clock signal by changing **`Clk.pos()`** into **`Clk.neg()`**. This ensures that the monitor module executes at the “right” time, namely *after* the values provided by the stimulus get processed by the multiplier, but *before* the next set of test vectors is fed in.

The main job of the monitor is to display and check the computed values from the multiplier. For each set of test vectors, print the current values of the input and output signals to the screen and then add a corresponding SystemC assertion that validates the values for correctness.

Step 5: Simulate the model with the Accellera SystemC library

Inspect the provided **`Makefile`** with your text editor and ensure that you are using the correct file names and matching include-directives. Then compile and simulate your model.

```
make all
make test
```

The compilation should run smoothly without any errors or warnings, and the simulation should show the 7 test vectors and correctly computed products.

Finally, add a brief text file **`README.txt`** in which you describe the model and any issues you encountered while building and simulating it. We may take this into account when grading.

Step 6: Create a package for submission

In order to submit (or distribute your model), create a compressed **`tar`** archive of your source files, as follows:

```
gtar cvzf hw3.tar.gz README.txt Makefile *.cpp *.h
```

3. Submission:

For this assignment, turn in the following deliverables:

```
hw3.tar.gz
```

To submit these files, change into the parent directory of your **`hw3`** directory and run the **`~ecps203/bin/turnin.sh`** script. Again, this command will locate the current assignment deliverable and allow you to submit it, just as before.

Remember that you can use this turnin-script to submit your work at any time before the deadline, *but not after!* Since you can submit as many times as you want (newer submissions will overwrite older ones), it is highly recommended to submit early and even incomplete work, in order to avoid missing the hard deadline.

Late submissions will not be considered!

To double-check that your submitted files have been received, you can run the **`~ecps203/bin/listfiles.py`** script.

For any technical questions, please use the course message board.

--
Rainer Doemer (EH3217, x4-9007, doemer@uci.edu)