EngrECE 298
System-on-Chip Description and Modeling
Spring 2004

Assignment 2

Posted: April 30, 2004 (week 4)
Due: May 7, 2004 (week 5)

Task: Get familiar with Modeling in SpecC

Instructions:

The goal of this second assignment is to make yourself familiar with modeling of SoC designs using the SpecC language. We will use the application introduced in the first assignment, MPEG Audio.

Task 1: Make yourself familiar with the usage of the SpecC compiler, **`scc`**.

For this, log into the server **`alpha.eecs.uci.edu`** using the Secure Shell interface **`ssh.`** The SpecC compiler is available on the server in the directory **`/opt/sce-20030530/`**. To use it, run the provided setup script:

```
source /opt/sce-20030530/bin/setup.csh
```

Next, copy the examples found in **`/opt/sce-20030530/examples/simple/`** into a working directory. Then, use **`scc`** to compile and run those examples. Of course, you may also want to take a look at the sources and the **`Makefile`** provided in the example directory.

Task 2: Make yourself familiar with the SoC Environment, **`sce`**.

For this, read the SCE tutorial **`sce-tutorial.pdf`** found in the directory **`/opt/sce-20030530/doc/SCE_Tutorial/`** on the server. Follow the initial steps described in the tutorial document and learn how to use SCE. Note that the initial steps of the document are sufficient for now. You don't need to go beyond the capture and simulation of the provided specification model of the vocoder application.

For the next task, it will also be helpful if you look at the source code of the vocoder specification model so that you understand how the test bench with the design under test is modeled. Just click on the **`Main`** behavior and open the source code editor.

Task 3: Create an initial Specification Model for MPEG-Audio.

For this task, you will need to follow the steps and guidelines discussed in the fourth lecture. Your specification model should contain a test bench at the top level of the hierarchy, consisting of a **`Stimulator`**, **`Decoder`**, and **`Monitor`** behavior. The stimulator behavior should read a compressed audio file and feed it into the decoder behavior. The monitor behavior should receive the decoded audio stream and write it into an output file. Then, a script (or Makefile) can be used to compare the generated file against a known good file.

As a guideline, your specification model should look at the top level just like the specification model of the vocoder application used in the tutorial.

--
Rainer Doemer (ET 444C, x4-9007, doemer@uci.edu)