Homework 3
System Design Flow and System-Level Design Tools

Assigned:	**Fri**	**10/25/19**
Due:	**Tue**	**11/05/19, noon**

Instructions:

- Please submit your solutions via Canvas. Submissions should include a single PDF with the writeup and an tar.gz archive for any supplementary files.
- In contrast to the lab assignments, the homework assignments have to be completed individually.
- You may discuss the problems and the concepts with your classmates. This fosters group learning and improves the class' progress as a whole. However, make sure to submit your own independent and individual solutions.
- Some questions might not have a clearly correct or wrong answer. In such cases, grading is based on your arguments and reasoning for arriving at a solution.

.

Problem 3.1: Goals of System Level Design Languages (10 points)

In class, we discussed six goals of an SLDL. Please select two out of these six which are in your consideration the most important for system level design. Name each goal, and describe its importance and influence in system level design. You may use an example each to illustrate your point.

Problem 3.2: Specification Model (15 points)

Writing a good specification model is essential to a successful design process. We discussed in class how to write a specification model as input to a top down system level design flow. Name 3 requirements of a specification model, explain each requirement and outline what influence it has to the design process.

Problem 3.3: System Model (15 points)

Assume you are tasked with writing the drivers on top of a Linux operating system for a custom hardware device for video compression. The interface to the hardware accelerator and its interface is already specified (e.g. Memory Mapped Registers, DMA channels, functionality). The application is also available in C. How would you model the system to allow driver development while the integrated chip is not available? Describe how you at which abstraction level to capture the drivers and the accelerator. Describe your reasoning (advantages / disadvantages) for your selected abstraction level.

Problem 3.4: Simulation Performance (10 points)

Two virtual platforms are captured in an SLDL (e.g. SystemC, SpecC). Both models are identical except that they differ in their timing annotations. Platform ***A*** terminates at 125.048.350 time units with simulating 5.123 waitfor statements. Platform ***B*** terminates at 645.053.000 time units with simulating 1.736 waitfor statements. For which simulation does the designer have to wait less (shorter real-time)? State the underlying reasons for the performance difference.

Problem 3.5: System-On-Chip Environment (SCE) (20 points)

The goal of this problem is to make you familiar with the System-On-Chip Environment (SCE) by going through the tutorial that demonstrates SCE on the GSM Vocoder design example introduced in class. The tutorial instructions are available as part of the SCE installation (see below) and online at:

http://www.ece.neu.edu/~specc/sce/sce-tutorial/html/

http://www.ece.neu.edu/~specc/sce/sce-tutorial/sce-tutorial.pdf

SCE is installed next to the SpecC tools on the COE Linux servers. Instructions for accessing and setting up SCE and the tutorial are posted at the SpecC Wiki page:

https://github.com/neu-ece-esl/specc/wiki/sceIntro

Again, once logged in (e.g. VLAB, or remotely via `ssh -X` and make sure to have an X11 server running locally), you need to run the provided setup script (depending on your $SHELL):

```
source /ECEnet/Apps1/sce/latest/bin/setup.{c}sh
```

Next, setup a local working directory for the tutorial demo, launch the SCE GUI and follow the steps of the tutorial:

```
mkdir demo
cd demo
setup_demo.{c}sh
sce
```

Read and go through the tutorial up to and including Section 3 (you are free to venture into HW and SW synthesis steps but those have not been tested and are not required at this point; use at your own risk but if you do, we would be happy to hear about any successes/failures). Make sure to simulate all the generated models and generate both a TLM and a PAM in the final Communication Synthesis step.

Report on model complexities (File→Statistics), the time it takes for simulating the design and encoding delays (maximum/worst-case delay over all frames) after each design step. The encoding delays are a product of the simulation, hence encoding delay is in virtual time. The encoding delays are printed as part of the simulation.

Hint: To have real-time time of a simulation run reported, go to Project→Settings→Simulator and prepend `/usr/bin/time` in front of the Simulation Command.

Problem 3.6: Interpretation of Results (30 points)

The previous question did guide you through a refinement process of the GSM Vocoder. Your model underwent a stepwise refinement and you have obtained detailed results for each refinement step.

(a) SCE provides statistics about model the complexities (File→Statistics). It reports the number of behaviors and classifies them as either 'extern', 'leaf', 'sequential', 'concurrent', 'fsm', or 'other'. Explain this classification (what is the meaning of each). For each category, briefly outline the properties (or potentials) a behavior of such a category has. Describe the importance of this classification with respect to the synthesis flow.

(b) In Problem 2.3, you captured metrics (number of lines, simulation time and simulated time) for each model in the refinement process. Graph the result for each metric individually. The x-axis enumerates the models. The y-axis shows the actual metric. For each graph, explain the development of the actual metric over refinement stages (explain the trend).

(c) The graphs for simulation time (how long does it take to execute the model) and simulated time (what does the model report as a time within the simulation) show a trade-off. This trade-off is called the TLM trade-off. Describe the TLM-trade-off with a few sentences.