Embedded System Design and Modeling

EE382V, Fall 2008

Homework #1

System Design, Models of Computation, Languages, SpecC

Assigned: September 11, 2008
Due: September 25, 2008

Instructions:

- Please submit your solutions via Blackboard. Submissions should include a single PDF with the writeup and an archive for any supplementary files.
- You may discuss the problems with your classmates but 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 1.1: System Design (10 points)

During design space exploration as part of the system design process, the target system architecture and its key architectural parameters are decided on. These design decisions have a major influence on the final design quality metrics such as (i) performance, (ii) power, (iii) cost, and (iv) time-to-market:

(a) Briefly discuss how the following target platform styles rate in relation to each other in terms of the metrics listed above:
  - A pure software solution on a general-purpose processor
  - A general-purpose processor assisted by a custom hardware accelerator/co-processor
  - A general-purpose processor and a specialized processor (DSP or ASIP)

(b) Try to sketch a potential simple strategy for exploring the design space for a given application under a given set of constraints/requirements.

Problem 1.2: Non-determinism (10 points)

(a) What is nondeterminism?

(b) How might nondeterminism arise? (give one example not discussed in class)

(c) What are the advantages and disadvantages of having nondeterminism in a language or model, i.e. in what circumstances might it be positive/desired or negative/undesired?

Problem 1.3: Process and Dataflow Models of Computation (15 points)

In relation to process network and dataflow type Models of Computation (MoCs), we talked about properties such as determinism and deadlocks. For the following questions, either provide a proof (property holds in all cases) or a counter-example:

(a) Does adding an `empty()` method that reports whether a channel has waiting tokens change the determinism of Kahn Process Networks (KPNs)?

(b) To determine the relative rates for a static schedule of Synchronous Dataflow (SDF) actors, one can derive and solve the system of linear *balance equations*. Can a SDF model have satisfiable balance equations (i.e. be consistent) and not run forever? Can a SDF model violate the balance equations (i.e. be inconsistent) and still run forever?

(c) Can a tree-structured (i.e. acyclic) SDF model ever have deadlocks? Can a static, complete and bounded schedule always be found?

Problem 1.4: State-Machine Models of Computation (15 points)

In class, we have discussed the concepts of hierarchy (OR state) and concurrency (AND state) for reducing complexities in a HCFSM (e.g. StateCharts) model. However, both hierarchical and concurrent FSM compositions can be converted into an equivalent plain FSM model:

(a) Show an example of converting a hierarchical FSM into an equivalent plain FSM. Derive an expression for the complexity (number of states and number of transitions) of the equivalent plain FSM as a function of the complexity of the OR-composed FSMs.

(b) Show an example of converting a concurrent FSM into an equivalent plain FSM. Derive an expression for the complexity (number of states and number of transitions) of the equivalent plain FSM as a function of the complexity of the AND-composed FSMs.

Problem 1.5: Languages (10 points)

(a) Why are sequential programming languages (e.g. C/C++/Java) considered to be insufficient for embedded system specification and design?

(b) Why are hardware design languages (e.g. VHDL/Verilog) considered to be insufficient for embedded system specification and design?

Problem 1.6: Models of Computation and Languages (10 points)

In class, we learned about the different Models of Computation (MoCs): KPN, SDF, FSM(D), HCFSM, PSM.

(a) What is the relationship between MoCs and languages?

(b) Can SpecC support all these MoCs? If so, briefly sketch for each MoC that you think can be supported how you would represent a corresponding model in SpecC.

Problem 1.7: SpecC Compiler and Simulator (10 points)

The goal of this problem is to make you familiar with the SpecC environment and basic tools, in particular the SpecC compiler and simulator (`scc`), using some simple examples included with the SpecC tool set.

The SpecC environment is installed on the ECE LRC Linux servers. Instructions for accessing and setting up the tools are posted on the class website:
http://www.ece.utexas.edu/~gerstl/ee382Vf08/docs/SpecC_setup.pdf
In short, once logged in (e.g. remotely via `ssh`), you need to run the provided setup script (depending on your $SHELL):

```
source /home/projects/courses/…/sce-20080601/bin/setup.{c}sh
```

Next, copy the examples found in `$SPECC/examples/simple/` into a working directory for this problem:

```
mkdir hw1_7
cd hw1_7
```

```
cp $SPECC/examples/simple/* .
ls
```

You can then use the provided `Makefile` to compile and simulate all examples:

```
make all
make test
```

Inspect the sources of all examples and the included `Makefile` to understand the use of `scc` for the compilation and simulation process, experiment with the `scc` command-line usage and start modifying the examples to experiment with different features of the SpecC language:

(a) Briefly explain (in 1-2 sentences) the functionality and structure of each example.

(b) Report the output from simulation of each example.

Problem 1.8: SpecC Language and Modeling (20 points)

The directory `$SPECC/examples/parity` contains an example of a parity generator written in SpecC. Copy the example to a local working directory and follow the instructions in the README file to compile and run it:

(a) Draw the PSM diagram (graphical SpecC notation/representation) of the system and briefly describe its functionality. What is the functionality of the behaviors in the files `ones.sc` and `even.sc`? Could the same functionality be modeled as a single behavior and/or why do you think it might be useful or necessary to have it split into two?

(b) Compile and simulate the example for at least three different inputs to verify that it produces the correct results.

(c) Modify the example to separate computation from communication, i.e. replace all shared variable plus event communication between `ones` and `even` to exclusively use (synchronous) message-passing (channel type `c_double_handshake`). Simulate the modified code to verify its correctness. Draw the modified PSM diagram and submit the modified source code.