Embedded System Design and Modeling

EE382V, Fall 2010

Homework #1

System Design, Methodologies, Languages, SpecC

Assigned: September 2, 2010
Due: September 16, 2010

Instructions:

- Please submit your solutions via Blackboard. Submissions should include a single PDF with the writeup and a single Zip or Tar 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 general, grading is based on your arguments and reasoning for arriving at a solution.

Problem 1.1: System Design

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: Languages

(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 1.3: Language Concepts

We have discussed the importance of separation of concerns.

(a) What is separation of concerns?

(b) Name two concerns that should be separated in a language as they cover orthogonal aspects?

(c) Show a short code excerpt that demonstrates a non separated code and code that separates these concerns (in C/C++/SpecC/SystemC or similar).

Problem 1.4: Specification

Writing a good specification model is essential to a successful design process. One important characteristic of a specification model is to be *free of implementation detail*. Please briefly outline why this is important with respect to the top down design flow.

Problem 1.5: Design Methodology

(a) Compare and contrast a top-down, bottom-up and meet-in-the-middle design methodology.

(b) Why do we perform computation design before communication design in the SpecC/SCE methodology?

Problem 1.6: SpecC Language and Modeling

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/ee382v_f10/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/gerstl/sce-20080601/bin/setup.{c}sh
```

The SpecC installation includes a comprehensive set of examples showing the features and use of the language. Examples are found in `$SPECC/examples/simple/`. You can copy them into a working directory:

```
mkdir hw1
cd hw1
cp $SPECC/examples/simple/* .
ls
```

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

```
make all
make test
```

It is recommended to inspect the sources of all examples and the included `Makefile` to understand the use of `scc` for the compilation and simulation process, and to experiment with the `scc` command-line usage and with the various `sir_`*`xxx`* tools.

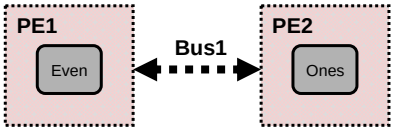
The directory `$SPECC/examples/parity` contains an example of a parity generator written in SpecC. For this assignment, copy the example to a local working directory and follow the instructions in the README file to compile and run it:

(a) Draw the SpecC diagram (graphical 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) 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, i.e. using the `c_double_handshake` channel from

the standard SpecC channel library. Simulate the modified code to verify its correctness. Draw the modified SpecC diagram and include a listing of the modified source files.

(c) Assume a partitioning where the `Even` behavior is mapped to *PE1*, the `Ones` behavior is mapped to *PE2*, and everything is statically scheduled:



Manually refine the specification model from (b) into a computation model where the `Parity` design reflects this partitioning. Insert execution delays of 30/50 time units per word in `Even`/`Ones`. Modify the testbench (`IO`) to print the input-to-output latency of the parity encoder. Document the transformation steps you applied and include listings of your modified source code.