Embedded System Design and Modeling

EE382V, Fall 2009

Homework #1

System Design, Methodologies, Languages, SpecC

Assigned: September 10, 2009
Due: September 24, 2009

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 (20 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: Amdahl's Law (20 points)

(a) Derive the equation for the theoretical upper bound on the speedup S that can be obtained in a sequential program when only a fraction P of the program can be accelerated (e.g. through implementation in custom hardware) by a maximum speedup factor of A (a speedup of $A = 2$ means fraction P can be run twice as fast).

(b) What is S for $P = 60\%$ and $A = 10$? What is S for $P = 60\%$ and $A = \infty$ (running the accelerated portion in zero time)? What is your interpretation of the results?

Problem 1.3: Languages (20 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 1.4: SpecC Compiler, Simulator and Tools (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/ee382v_f09/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
```

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

```
mkdir hw1
cd hw1
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 with the various `sir_`*`xxx`* tools:

(a) Compile each example into an SIR file and use `sir_tree` to obtain a graphical representation of the hierarchy of each design.

Problem 1.5: SpecC Language and Modeling (30 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) 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 files.