

Use of a Soft-Core Processor in a Hardware/Software Codesign Laboratory

**Roger Chamberlain, John Lockwood, Saurabh Gayen,
Richard Hough, and Phillip Jones**

Roger Chamberlain, John Lockwood, Saurabh Gayen, Richard Hough,
and Phillip Jones, "Use of a Soft-Core Processor in a Hardware/Software
Codesign Laboratory," to appear in *Proc. of Int'l Conf. on Microelectronic
Systems Education*, June 2005.

Dept. of Computer Science and Engineering
Washington University
Campus Box 1045
One Brookings Dr.
St. Louis, MO 63130-4899

Use of a Soft-Core Processor in a Hardware/Software Codesign Laboratory

Roger Chamberlain, John Lockwood, Saurabh Gayen, Richard Hough, and Phillip Jones

Department of Computer Science and Engineering, Washington University in St. Louis
roger@wustl.edu, lockwood@arl.wustl.edu, sg3@cec.wustl.edu, rh3@cec.wustl.edu, phjones@arl.wustl.edu

Abstract

This paper describes our experience to date and current plans for a senior-level microelectronics laboratory course on hardware/software codesign. The course utilizes an open-source, soft-core processor deployed on the FPX platform as an integral component of the students' designs. Students write software to execute on a Leon SPARC-compatible processor and write VHDL to implement hardware-accelerated computational functions in FPGA hardware.

1. Introduction

Practical experience with hardware/software codesign is an important aspect of an undergraduate education in computer engineering. At Washington University in St. Louis, students gain this experience in using an open hardware platform and new tools developed for a senior-level laboratory course.

This paper describes the course and its recent modernization to use new infrastructure developed by the NSF-supported ITR research project entitled Liquid Architecture. The focus of the research project is the investigation of techniques to exploit flexible microarchitectural components at the individual application level [1].

2. Digital Systems Laboratory Course

The senior-level, Digital Systems Laboratory is a required course for computer engineering students, an elective course for computer science students, and an elective course for electrical engineering students.

The pedagogical goals of the course include: providing hands on experience with codesign, developing teamwork skills in the context of an interdisciplinary hardware/software project, and introducing the students to modern computer-aided simulation, synthesis, and debugging tools. While much of the subject matter has been introduced earlier in the curriculum (algorithms, data structures, programming, computer architecture, and digital logic design), the focus of this course is to put the concepts

previously discussed into practical experience. Specifically, students learn how to interface with a System-on-Chip (SoC) bus, how to make effective use of computer-aided design tools, and how to design systems in a way that internal state is observable and the system can be debugged.

Working in small groups of 2 to 3 individuals, the students are expected to design and build successively more complex systems that incorporate both hardware and software design elements. End of semester projects have included audio equalizers, oscilloscopes, and video games.

In previous versions of the class, the laboratory infrastructure included a Motorola 68000 processor and a Xilinx 4000-series FPGA. The students' software was authored in 68K assembly language, and the hardware design was implemented using a schematic design package.

3. Description of New Infrastructure

In the new instantiation of the Digital Systems Laboratory course, the 68K processor is replaced with a soft-core processor. The logic for this processor is synthesized into a Virtex FPGA along with the hardware portion of the students' designs. We use the Leon [2], an open-source core that implements the SPARC V8 instruction set. We use the FPX platform [3] for its dynamically configurable hardware, memory, and network connectivity. The network is used to deliver dynamically reconfigure the FPGA as well as executable binaries for the Leon processor. It is also used to collect output both from the profiling subsystem (described below) and students' designs. An illustration of the system is shown in Figure 1.

As part of their design effort, students author software code in C for execution on the Leon. In addition, students author a hardware module by writing VHDL. The combined circuits are then synthesized and deployed together onto the Virtex FPGA on the FPX platform. The interface between the Leon processor and the students' hardware design is the AMBA bus, a popular on-chip infrastructure used for many SoC designs.

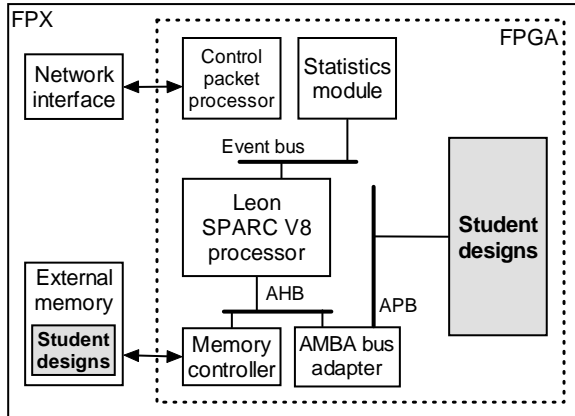


Figure 1. System architecture.

The course leverages the use of existing software and hardware tools. The software tool set includes the `gcc` compiler suite and the GRMON instruction-level simulator and debugger. GRMON is provided with the Leon system and its debugger is based on `gdb`. The hardware tool set includes Modelsim, Synplicity, the Xilinx place-and-route tools, and the Identify tool. Identify provides the students with an on-chip logic analyzer that can access internal hardware signals.

A statistics module and associated event bus were implemented to enable detailed performance monitoring of multiple aspects of the system without perturbing the performance of the system. Currently, the statistics module monitors cache hit/miss rates and provides cycle-accurate profiling of time spent in user-specified software routines. The ability to measure performance without perturbing the system is achieved by using independent hardware to both detect events and store intermediate results. Students can specify what statistics are to be collected and over what specific portions software by entering parameters to a web interface. The web interface software then interacts with the statistics module by sending control packets to and from the FPX hardware.

4. Experience to Date

The revised course was first offered during the Fall 2004 semester. A number of positive results were noted. First, the transition from programming in assembly language to C brought the students into alignment with current practice in the field. Second, the use of the soft-core processor enabled entire applications (both hardware and software) to be simulated using ModelSim prior to being executed on the FPX platform.

While there were clear advantages to the new course, there were still issues to resolve at the end of the semester. The entire processor core was included

as source code in the students' hardware design project. No incremental design optimizations were utilized. To build circuits, students ran tools to resynthesize and place-and-route all of the logic. Even using fast machines (Athlons and Pentium 4s), the CAD tools still took over a half-hour to run.

5. Current Plans

To address the long CAD run times, we are developing an incremental design flow where the presynthesized and routed Leon processor core will interface to dynamically-generated, student-defined logic across pre-defined signals of the AMBA bus mapped to a fixed location on the FPGA. To make the infrastructure easy to use for students, tools have been developed that integrate eGroupWare with on-line, in-circuit testing of the FPX hardware [4].

6. Conclusions

We have developed a course where students both develop hardware and software to implement complete System-on-Chip projects. The projects the last two semesters include hashing for biosequence search applications and network packet encoding/decoding. Students use modern computer-aided design tools to compile, simulate, synthesize, place, route, and debug their codesigned systems. They compile programs for the Leon processor with `gcc` and debug their software with GRMON. The combined system is deployed within FPGA logic that runs on the FPX platform.

7. Acknowledgements

This work has been supported by NSF under grant 0313203. The authors would like to acknowledge the efforts of the entire Liquid Architecture group.

8. References

- [1] P. Jones, S. Padmanabhan, D. Schuehler, S. Friedman, P. Krishnamurthy, H. Zhang, R. Chamberlain, R. Cytron, J. Fritts, and J. Lockwood, "Extracting and Improving Microarchitecture Performance on Reconfigurable Architectures," *Proc. Workshop on Compilers and Tools for Constrained Embedded Systems* (at CASES), September 2004.
- [2] Free Hardware and Software Resources for System on Chip. <http://www.leox.org>
- [3] J. Lockwood; "Platform and Methodology for Teaching Design of Hardware Modules in Internet Routers and Firewalls," *Proc. Microelectronics Systems Education*, June 2001, pp. 56-57.
- [4] J. Mitchell and J. Lockwood, "Tools for On-Line, In-Circuit Testing of Internet Content Processing Hardware," *Proc. Microelectronics Systems Education*, June 2005.