A Holistic Approach to Computer System Design Education Based on System Simulation Techniques

Per Stenström and Fredrik Dahlgren

Department of Computer Engineering
Chalmers University of Technology,
SE-412 96 Gothenburg, Sweden
{pers,dahlgren}@ce.chalmers.se

Abstract

In the design of competitive application-specific computer systems, it is important to evaluate design alternatives in which commodity hardware and software components are interwoven with tailor-made components. At Chalmers, we are offering a specialization in computer engineering that emphasizes a holistic approach to the design of application-specific computer systems. Apart from offering disciplinary courses in hardware and software technologies, and design methods, it also offers a project course in which the students apply a holistic design methodology to realistic system design projects. A key component in this project course is the use of system simulation techniques, that have been developed in computer architecture research projects, as an important infrastructure to attack design issues spanning all design levels. This paper provides an overview of the educational goals and the course content of the specialization. It then discusses how system simulation techniques have been used in the project course.

1.0 Introduction

A prevailing methodology in the design of computer systems is to carefully analyze the interaction between the application and a computer-system design alternative with respect to e.g. performance consequences. While this methodology has been applied to the design of instruction-set architectures (ISAs) for several decades, it is applicable to the design of almost any component of contemporary computer systems such as compilers, operating systems, memory and I/O subsystems. This design methodology now forms a common ground for computer architecture courses thanks to the widely adopted text by Hennessy and Patterson [2]. While this textbook mainly focuses on how the methodology is applied to the design of microprocessors, it can be naturally generalized to the design of application-specific computer systems, where commodity components (microprocessors and standard software packages) are interwoven with tailor-made components. Training engineers to make design tradeoffs in a systematic way is an important goal for computer-engineering curricula in Sweden as well as elsewhere.

At Chalmers, a computer-system design specialization within the computer science and engineering curriculum has been designed that has this educational goal in mind. In this paper, we discuss our strategy to reach this goal. In particular, we present our ideas of a project course in which a holistic approach is taken to analyze and compare design alternatives of application-oriented computer systems. In Section 2.0, we discuss the overall educational goals of the specialization in more detail. Section 3.0 then focuses on the content in terms of courses and how they together meet the goal. One such course is a computer-system project course that aims at providing a holistic view of computer-system design and in which all the other disciplinary courses are put into perspective. In Section 4.0 we finally present how we use system simulation techniques in this project course as an important infrastructure to apply a holistic methodology to the design of application-specific computer systems.

2.0 Educational Goals

The main goal of a computer engineering curriculum is to train students to make good design tradeoffs within the constraints of the technologies to meet various requirements set out by end-users in terms of e.g. functionality and performance. Competitiveness is another important attribute that can be translated into cost constraints and conformity to technology trends.

A pedagogical strategy to reach this goal is reflected in Hennessy and Patterson's widely used text [2]. This text teaches us that performance analysis driven by workloads that represent usage is key to evaluating design alternatives in a systematic way across technology boundaries (e.g. compiler, operating system, and hardware technologies). For obvious reasons, the design methodology in the text is applied to high-performance computer design with the goal of training engineers for the core of the computer industry. While this is certainly an important goal in the U.S., a larger number of computer engineers are engaged in application-specific computer system design where high-performance microprocessors and standard software components are interwoven with tailor-made hardware and software components to meet cost/performance or other goals.

Many companies, in Sweden for example, design computer systems dedicated to applications that require high and predictable (real-time) performance and with high availability demands. Telecom is such an application domain where computer servers typically must support a high transaction throughput and for which the processing time of a single transaction must be bounded. Such applications have requirements similar to database and transaction processing applications. It is therefore natural to design these application-specific servers from commodity components by using shared-memory multiprocessors or clusters as important building blocks [8]. However, this example is also illustrative in terms of pin-pointing the particular competencies needed to design industrially important application-specific systems:

- Computer architecture
- Software technologies (operating systems, compilers)
- Hardware technologies (e.g. microelectronics)
- Design methods (real-time and fault-tolerant system design; parallel and distributed hardware and software design)

Computer architecture naturally forms the common ground for establishing the general view of making systematic design tradeoffs across the hardware and software boundary apart from conveying design principles of high-performance single and multiprocessor systems. Apart from this, the engineer should have a deep understanding of the basic properties of the technologies involved (e.g., system software as well as microelectronics) so as to make justified tradeoffs between commodity and tailor-made components to meet the application requirements. Another important ingredient is design methods which should provide the engineer with a framework for interpreting and converting application requirements in terms of performance, dependability, and real-time function into implementable specifications so that design alternatives can be analyzed and compared.

We have designed a specialization in computer systems engineering with this goal in mind, which we discuss next.

3.0 Computer-System Engineering Specialization

The courses in the specialization structured into categories are laid out in the diagram in Figure 1. Because the design methodology in [2] forms the basis for the specialization, a course in Computer Architecture based on this text is mandatory. In addition to this basic course, more advanced courses on computer architecture are offered such as on parallel computer architecture. Then the students have an option to pick courses fairly freely according to their

interests. In this way, it is possible for the students to get a more "soft" or a more "hard" profile. In order to get a certain depth, however, some courses that they select must be graduate courses. There are of course implicit dependencies among courses defined by precedences in terms of prerequisites. Also, what is not shown is a seminar course called "industrial computer system engineering" aiming at conveying experiences from real design projects by inviting lecturers from other institutions and companies.

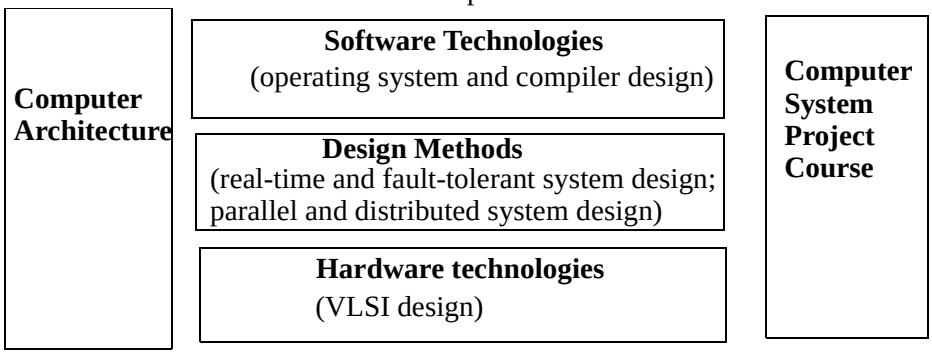

Figure 1: Course content by category in the specialization.

Most notably, however, there is a project course (computer system project course) that aims at taking a holistic approach to put all the pieces together. We next discuss the approach we have taken to provide this holistic approach by using a design evaluation testbed based on system simulation techniques.

4.0 Computer System Project Course

The problem with a disciplinary approach to a curriculum is that the students lose the "big picture". We have addressed this problem by offering a course on computer system projects that aims at taking a holistic approach to put all the pieces together. Our key strategy is to use a complete system simulation platform as an infrastructural component to carry out design projects. We first describe the basic simulation platform in Section 4.1 and then we briefly discuss how we have used it in our project class in Section 4.2.

4.1 System Simulation Techniques

Complete system simulation is a concept that refers to modeling an entire computer system consisting of interacting hardware and software components using simulation techniques. Thus, a detailed model from a functional and timing point of view is designed that can be used to study the interaction between the application and the computer system, from a functional and performance point of view.

Practically useful simulation infrastructures for complete system simulation have evolved from concrete needs in computer architecture research. For example, instruction-level simulation approaches, such as Tango [1], were motivated from the needs in e.g. multiprocessor architecture research. Typically, applications with virtually no operating system interaction are executed on top of such simulators that can drive timing models of e.g. memory systems to evaluate and compare design alternatives. Several simplifications were made to gain simulation efficiency such as abstracting away the pipeline organization and the impact of operating system activities. This was a reasonable design tradeoff given the concrete research issues addressed in the research projects.

Recently, we have seen practical simulation platforms that can also model the performance impact of virtually all the components found in an application-specific computer system. At the microarchitectural level, researchers have designed detailed simulation models that make it possible to evaluate the performance of superscalar-processor design alternatives [6]. At the system level, it is also possible to model the interaction between the operating system and the

hardware platform. Examples of practical systems are the SimOS [5] and the SimICS/Sun4m [3] simulation platforms. On these platforms, it is possible to run codes with a high efficiency.

The SimICS platform, that we use, is a complete system simulation platform. The core of SimICS is a highly efficient instruction-set simulator that currently models SPARC V8. In order to gain speed, the simulator assumes that all the instructions take the same time to execute. The instruction-set simulator is augmented with features that support symbolic debugging and profiling. With a slowdown of less than two orders of magnitude, it keeps track of the content of TLBs and caches. It thus allows simple cache measurements to be taken such as cache/TLB miss measurements. Of course, the slowdown depends to a great extent on the amount of detail in the timing model.

An overview of the SimICS/Sun4m platform is shown in Figure 2. From a functional point of view, the platform implements a system model sufficient to port a complete operating system. In essence, this functional system model mimics the functionality of the Sun4m architecture and all operating systems that have been ported to this model can run correctly with no modifications. Currently we use Linux 2.0.30. Thus, all Unix-compliant binaries can run unmodified. In addition, it is of course also possible to develop applications, then compile them, and then profile them so as to pin-point performance bottlenecks.

Apart from the functional model provided by the system architecture, it is possible to develop detailed timing models of the computer system. Such models are plugged into the SimICS environment. It is also possible to use the simulation platform to develop detailed timing models of multiprocessor systems. We use the very same environment as a key resource in our research projects that focus on design principles for shared-memory multiprocessors and their interaction with scientific, technical, and database applications [4].

In summary, the SimICS/Sun4m platform can be used as a general tool to evaluate the performance of design alternatives spanning the application code, compiler, operating system, and hardware platform.

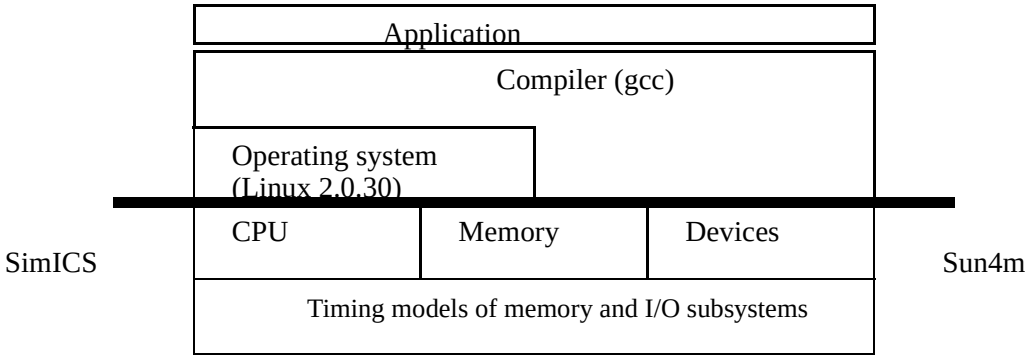

Figure 2: The overall organization of the SimICS/Sun4m simulation platform.

4.2 Example Project

With the tool outlined in the previous section as an important infrastructure, it is possible to design models of application-specific computer systems. These models can mimic commodity components (e.g. microprocessors taking the form of instruction-set simulators) as well as tailor-made components (e.g. special functions in the OS, application code, and even hardware components etc.). Thus, it is possible to make design tradeoffs based on solid evaluations of performance consequences of design alternatives across all the levels in a computer system. It is also possible to analyze the impact of technology trends on the performance of an application-specific computer system. To do this, one can parameterize the timing models to study the impact of future technologies on the performance of the system.

We are currently offering a course using this tool for the first time. The design project this year has the following goal. Make the system design of a multiprocessor-based system using SPARC processors and a Linux operating system so that a parallel application (example applications from SPLASH-2) can be run within given performance constraints. To carry out the task, the participating students design a multiprocessor simulation model and carefully evaluate the performance consequences of design alternatives including algorithmic changes, operating system changes, and multiprocessor architectural changes depending on their background.

The course is given for the first time this year and the experiences so far are promising. Based on prerequisites from a course in parallel computer architecture, the students have studied memory system optimization techniques such as cache coherence protocol enhancements [7] and selected promising candidates based on application/architecture interactions.

5.0 Conclusions

We have discussed how a holistic approach can be taken to train computer engineers in application-specific computer system design. A key technology, that has significantly strengthened this educational approach, has been the availability of highly efficient complete system simulation environments, such as SimICS/Sun4m. This has opened up new opportunities to emphasize the design methodology that has played a key role in the major developments of high-performance computer systems.

References

[1] H. Davis, S. Goldschmidt, and J. Hennessy: "Multiprocessor Simulation and Tracing Using Tango," in *Proc. of 1991 Int. Conf. on Parallel Processing*, pp. 99-107 (Part II), 1991.

[2] J. L. Hennessy and D. A. Patterson: *Computer Architecture: A Quantitative Approach*, 2nd Edition, Morgan Kaufmann Publishers Inc., 1995.

[3] P Magnusson, F Dahlgren, H. Grahn, M. Karlsson, F. Larsson, A. Moestedt, J. Nilsson, P Stenström, and B. Werner: “SimICS/Sun4m: A Virtual Workstation. In *Proc. of USENIX98*, June 1998.

[4] J. Nilsson, F. Dahlgren, M. Karlsson, P. Magnusson, P. Stenström: "Computer System Evaluation with Commercial Workloads," in *Proc. of IASTED Conference on Modeling and Simulation*, pp. 293-297, May 1998.

[5] M. Rosenblum, S. Herrod, E. Witchell, and A. Gupta: "Complete Computer System Simulation: The SimOS Approach," in *IEEE Parallel and Distributed Technology*, pp. 34-43, 1995.

[6] S. Vijay, R. Parthasarathy, and S. Adve "RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors," in *TCCA Newsletter*, pp. 32-38, Sept. 1997.

[7] P. Stenström, M. Brorsson, F. Dahlgren, H. Grahn, and M. Dubois: “Boosting Performance of Shared-Memory Multiprocessors,” in *IEEE Computer*, pp. 63-70, July 1997.

[8] P Stenström, E. Hagersten, D. Lilja, M. Martonosi, and M. Venugopal: “Trends in Shared-Memory Multiprocessing,” in *IEEE Computer* , Vol. 30, No. 12, pp. 44-50, December 1997.