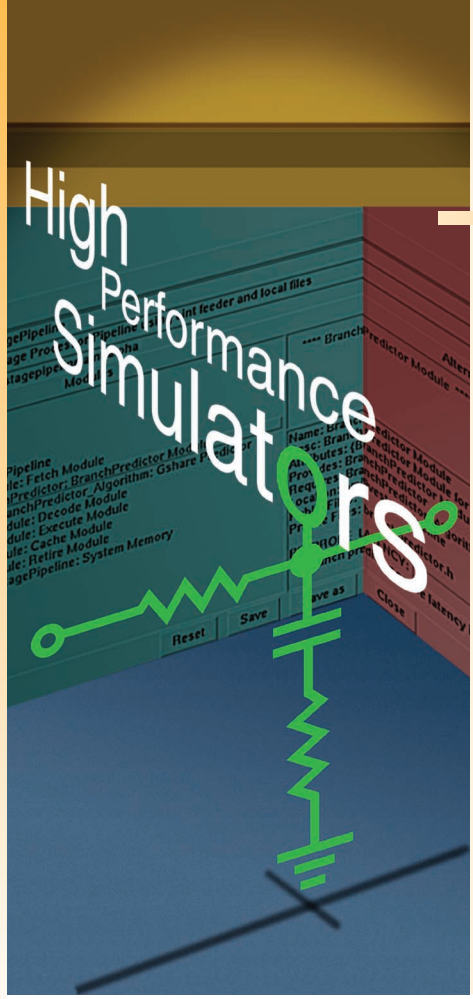

GUEST EDITORS' INTRODUCTION

Performance Simulation Tools

Understanding the performance of microprocessors, multiprocessors, and distributed computers requires studying them in isolation as well as observing their interaction with the entire system architecture.

Shubhendu S. Mukherjee
Intel

Sarita V. Adve
University of Illinois at Urbana-Champaign

Todd Austin
University of Michigan

Joel Emer
Intel

Peter S. Magnusson
Virtutech Inc.

High-performance computing has grown largely in scale with Moore's law. As hundreds of millions—and soon to be billions—of transistors crowd onto processor chips, they support computing devices of extreme complexity. Predicting the performance of these machines often requires using sophisticated software programs to model them. Performance simulators—software programs typically written in a high-level language such as C or C++—enable exploration of design alternatives for future high-performance computers.

Since the early 1980s, the design of high-performance computers has been largely data-driven. For example, analyses of instruction usage revealed that real machines do not use all instructions with equal frequency. Designers used this observation to optimize the implementation of these machines.

Direct measurement, however, is a post-design step and does not always help optimize machines under design. As an alternative, designers adopted analytical models to predict performance. Such models are successful in many cases, particularly in culling the design space in preliminary explorations. However, analytic models have been less successful in assessing detailed design tradeoffs. Because these tradeoffs are crucial in today's highly competitive high-performance computing market, designers have reverted to simulation models to predict machine performance.

Performance modeling and analysis are now integral to the design flow of modern computing systems, especially for high-performance microprocessors. As Figure 1 shows, designers begin by developing performance models of the target architecture, followed by actual logic design—also called Register Transfer Language, or RTL. Circuit designers convert the logic specification into circuits, and layout engineers eventually position the circuits on the processor floor plan.

For very complex designs, such a design process can take as long as seven years. Of course, the process involves close interactions between the steps. The performance model, in particular, is refined as logic, and circuit designers feed back better timing estimates for different hardware components of the target architecture. Thus, the performance model's fidelity to the target architecture is often key to the success of the architecture itself.

Additionally, the performance model must run at least faster than RTL; otherwise, developers could obtain performance estimates from the logic blocks themselves. Our experience shows that performance models usually run several orders of magnitude faster than RTL.

The increased complexity of target architectures and applications has made performance modeling a daunting task. Microarchitecture pipelines have extended from five to 20 stages to exploit increasing levels of parallelism. This trend will continue as Web server and database markets require both fine-

0018-9162/02/$17.00 © 2002 IEEE

grained multithreading embedded in aggressive pipelines and coarse-grained multiprocessing spanning multiple processors. Such multithreading and multiprocessing architectures are even moving on chip, exemplified by Intel's Hyperthreaded architectures, IBM's Power4, and Hewlett-Packard's Mako processor.

Understanding the performance of distributed computing, network server, and parallel applications on this new generation of processors requires studying how they interact with the system architecture and the operating system. Unfortunately, performance models that can evaluate system architectures and operating systems have evolved into extremely complex and gigantic software projects.

IN THIS ISSUE

This special issue presents four performance simulators—Rsim, Simics, SimpleScalar, and Asim—that address different aspects of the complexities encountered in performance simulation.

In "Rsim: Simulating Shared-Memory Multiprocessors with ILP Processors," Christopher J. Hughes and coauthors review the development process for Rsim, an architecture simulator widely used in academic research related to multiprocessors. It provides detailed models for shared-memory multiprocessors based on processors that support dynamic scheduling. The experience with Rsim's detailed processor modeling demonstrates that simple models of an older generation of sequential processors cannot approximate the more complicated dynamically scheduled processors.

The Simics simulation platform is based on the idea that reliable performance estimates require full system simulation. Simics runs unmodified firmware, operating system kernels, and device drivers. In "Simics: A Full System Simulation Platform," Peter S. Magnusson and colleagues describe how this system simulates a network of multiple, heterogeneous computers that designers can use to duplicate real-world scenarios. Simics also can export the models for these functions to other tools. Originally an academic research project, Simics is today a commercial product, available from Virtutech.

"SimpleScalar: An Infrastructure for Computer System Modeling," by Todd Austin, Eric Larson, and Dan Ernst describes how researchers can reuse this uniprocessor performance simulator's tools to quickly obtain meaningful results from complex architectures. Subsequent to its development, other researchers incorporated models of multithreaded and multiprocessing architectures into SimpleScalar.

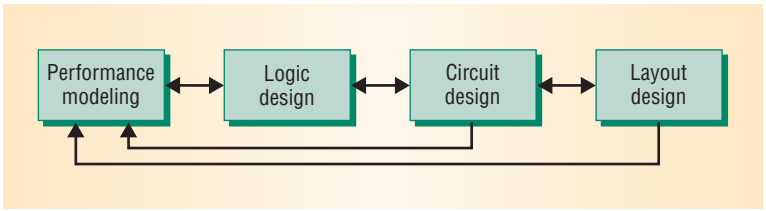

Figure 1. Typical flow in a microprocessor design process. Interaction between the process steps refines the performance model throughout the process.

Finally, Asim extends SimpleScalar's reuse philosophy to finer-grained modular components within the simulator itself. "Asim: A Performance Model Framework" by Joel Emer and his colleagues explains how Asim provides a simulation infrastructure with a library of modules that model different hardware components, such as caches and branch predictors. With this library, designers and researchers can easily reuse, extend, and modify architectural components to quickly build complex performance models. Currently, Asim is a proprietary tool within Compaq and Intel.

Performance simulation is a research topic of long-standing importance that comprises a huge body of literature. Comprehensive coverage is impractical in a single special issue, but these four articles and tools demonstrate the overall state of the art in performance simulation and also offer a glimpse of the problems and challenges that lie ahead. We hope you enjoy them. ■

Shubhendu S. Mukherjee *is a senior hardware engineer in VSSAD at Intel. Contact him at shubu.mukherjee@intel.com.*

Sarita V. Adve *is an associate professor in the Computer Science Department at the University of Illinois at Urbana-Champaign. She is a member of the ACM and the IEEE. Contact her at sadve@cs.uiuc.edu.*

Todd Austin *is an assistant professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. Contact him at taustin@eecs.umich.edu.*

Joel Emer *is an Intel Fellow in VSSAD. Contact him at Joel.emer@intel.com.*

Peter S. Magnusson *is CEO of Virtutech. Contact him at psm@virtutech.com.*