Multicore Education Through Simulation

Ozcan Ozturk
Computer Engineering Department
Bilkent University
Ankara, Turkey
ozturk@cs.bilkent.edu.tr

Abstract

This paper presents the experiences using a commercial full system simulation platform – Simics – in a graduate Chip Multiprocessors class. The Simics platform enables students and researchers to do research on computer architecture, operating systems, and hardware/software co-simulation. It provides the ability to simulate machines that are not physically available. This platform has been used in Chip Multiprocessors course to help graduate and undergraduate students in related areas. This course deals with both hardware and software issues in Chip Multiprocessors, and concludes with a team project at the end of the semester. The simulation-based approach was successful when student feedback and final projects are considered.

1. Introduction

As commonly accepted, current performance trajectory to double the chip performance every 24 to 36 months, can be achieved by the integration of multiple processors on a chip rather than through increases in the clock rate of single processors due to the power limitations present in processor design. Mutlicore architectures have already made their way in the industry [2], [3], [6], [4], [5], [7], with more aggressive configurations being prototyped such as the Intel's 80 core TeraFlop [1]. Since future technologies offer the promise of being able to integrate billions of transistors on a chip, the prospects of having hundreds of processors on a single chip along with an underlying memory hierarchy and an interconnection system is entirely feasible.

One of the most important benefits of a multicore architecture over a traditional single processor design is the power consumption reduction via reduced clock frequency. Other benefits of multicore architectures include: (i) scalability provided through many dimensions of parallelism such as thread-level parallelism, loop-level parallelism, and instruction-level parallelism (ii) simpler verification which in turn reduces the time-to-market and lowers the chip costs (iii) better utilization of the available silicon area since the cores share common logic (iv) faster and cheaper on-chip communication.

Despite the many advantages of multicore architectures over uniprocessor architectures, one of the key questions raised by many researchers is the effectiveness of multicore architectures [8], [9]. One aspect of this problem is due to the infancy of the software solutions targeting such architectures. Current programs, compilers, and software architecture techniques in general, rely on the fact that there is only one core running on the background [10]. Hence, it becomes very difficult to effectively use the underlying processing power. There are some initial attempts to target this problem, however these techniques are still in their infancy [10].

Clearly, teaching multicore architectures to today's computer engineers is a desirable goal in every curriculum. In general, computer engineering curriculum is designed to provide a balanced education in the design and analysis of both computer software and computer hardware. According to American Society of Engineering Education: "Computer engineers are solidly grounded in the theories and principles of computing, mathematics and engineering, and apply these theoretical principles to design hardware, software, networks, and computerized equipment and instruments to solve technical problems in diverse application domains." However, in practice, this is not always the case. Hence, many computer engineering schools concentrating on the software topics lack hardware knowledge.

On the other hand, current language extensions or class libraries are not sufficient to utilize the available processing power. Software designers need to understand the nature of multicore architectures and learn how to enforce sequencing of code on multiple cores. Therefore, it is critical to include multicore education in both undergraduate and graduate computer engineering curriculum.

In order to equip students with a better understanding of the multicore architecture and its programming, Chip Multiprocessors course can be offered with parallel programming concepts on these architectures. This will enable students to learn the state-of-the-art multicore architectures, while giving them the opportunity to write parallel programs on these architectures. Using Simics-like simulator in teaching such a course is both possible and practical since it is very expensive to obtain various multicore architecture systems and experiment on them. This is especially valuable for universities with limited financial resources.

In this paper, we present how Simics platform can be used in teaching different multicore architecture concepts. Specifically, Simics toolkit has been used in Chip Multiprocessors course to help both graduate and senior undergraduate students understand the hardware/software issues related to Chip Multiprocessors. The remainder of this paper is structured as follows. Section 2 gives details about the Simics platform and how it is used in teaching Chip Multiprocessors course. Course overview including the student background, objectives, teaching methodology, course plan, and assessment are given in Section 3. The paper is concluded in Section 4.

2. The Simics Platform

Simics [11] is a full-system simulator designed by Virtutech to strike a balance between accuracy and speed in simulation. Simics provides a range of accuracy/speed options. Specifically, simulation platform provides functional accuracy and sufficient abstraction while achieving tolerable performance levels. It is fast enough to run realistic benchmarks on a wide set of unmodified operating systems including Solaris, Linux, and Windows XP. Moreover, systems ranging from a basic embedded system to a complex multiprocessor environment can be modeled in Simics.

Computer architecture related courses can benefit from such a platform since it is flexible to support wide range of microprocessor types, instruction sets, operating systems, and memory hierarchies. In addition to this, through the Academic Licensing Program, it is offered to universities at no cost. Simics can be used in many different types of courses, from low-level computer architecture to parallel programming on different systems. At the computer architecture level, Simics can be used to teach effects of system parameters on the performance and energy. System properties can easily be modified through a configuration file, thereby opening up possibilities to experiment with machines of different configuration. Another important use is to teach students assembly language programming. Programs can be executed instruction by instruction providing the contents of registers and memory changes.

It is also possible to simulate a multicore environment using Simics, even if the host is a single processor architecture. This provides greater flexibility in testing arbitrary configurations including multiple processors, multiple nodes, and multiple device configurations. A sample configuration for two processors and a shared memory is given in Figure 1.

```
OBJECT cpu0 TYPE x86-hammer
{
    freq_mhz: 3500
    physical_memory: phys_mem0
}

OBJECT cpu1 TYPE x86-hammer
{
    freq_mhz: 3500
    physical_memory: phys_mem0
}

OBJECT phys_mem0 TYPE memory-space
{
    map: ((0xa0000, vga0, 1, 0,0x20000),
      (0x100000, mem0, 0, 0x100000,
      0xff00000),
    ...
}
```

Figure 1. An example configuration for two processors with a shared memory.

3. Course Overview

Chip Multiprocessors course was first offered in Spring 2008 with an initial enrollment of 8 students including graduate and senior undergraduate students, all computer engineering majors. This course was the first advanced computer architecture course to be offered in the department. As such, initial enrollment was not so high. However, the number of students doubled in Spring 2009, making the course among the most popular graduate courses in the computer engineering department. Students in the class had no exposure to advanced computer architecture topics except what has been covered in a sophomore computer organization class.

This course is designed to provide a deep understanding of the multicore architectures. Although computer engineering curriculum is designed to provide a balanced education in both software and hardware, in practice, many students lack major hardware knowledge. This course aims to fill this gap for average students, while providing deeper knowledge to the more interested ones. To achieve this goal, first part of the course covers multicore evolution starting from Moore's law with examples from state-of-the-art multicore architectures. Next, issues such as communication, memory hierarchy, cache coherency, data distribution, task assignment, and operating system involvement is covered. In the third part of the course, programming on such architectures is covered using OpenMP. In the last part, advanced topics such as heterogeneous multicore architectures, chip multithreading, etc. is discussed.

Student learning and instructional effectiveness is measured by student assessment. Assessment instruments used in Chip Multiprocessors course are exams, homeworks, and a semester-long project. While all of these instruments comprise the assessment, project is the most important parameter among these as it provides hands-on experience.

Most of the student projects tried to modify the architectural properties of a given multicore architecture and measured the effects of various parameters. For example, a team of students considered different cache hierarchies. Specifically, they performed a tradeoff analysis between shared versus private caches at levels L1 and L2 on several benchmarks. One of the systems tested is shown in Figure 1. Similarly, another group implemented a new memory coherency protocol, whereas another one studied message passing efficiency. These projects were all tested using real-life workloads and benchmarks such as SPEC.

Another group of projects tried to parallelize real-life applications on multicore architectures using Simics. For example, one group of students parallelized various matrix multiplication algorithms using OpenMP. After the parallelization step, programs are executed on different multicore configurations to measure the scaling of these algorithms with OpenMP. Furthermore, Simics is used by students to compare a hybrid OpenMP-MPI approach with pure-MPI and pure-OpenMP scenarios. For the specific application in question, students found out that hybrid OpenMP-MPI performs better compared to both pure-MPI and pure-OpenMP schemes.

4. Conclusion

This paper presents multicore education using a simulator. Instead of buying a big and expensive server to let students try and run parallel programs with real workloads, a full system simulation platform can be used. In particular, Simics has been used to implement different multicore architectures and writing parallel programs on these architectures. This simulation framework provides great advantages for teaching multicore architectures and parallel programming when the hardware is not available. Overall, simulation-based approach was successful when student feedback and final projects are considered.

5. Acknowledgment

This work is supported in part by EC FP7-PEOPLE Marie-Curie IRG Grant 239446.

References

[1] http://www.intel.com/idf/.

[2] Intel quad-core Xeon. http://www.intel.com/quad-core/ ?cid=cim:ggl—xeon _us_clovertown—k7449—s

[3] J. Kahle et al. Introduction to the Cell Multiprocessor. *IBM Journal of Research and Development*, 49(4-5), 2005.

[4] P. Kongetira et al. Niagara: A 32-Way Multithreaded SPARC Processor. *IEEE MICRO Magazine*, Apr. 2005.

[5] R. McGowen. Adaptive designs for power and thermal optimization. In *Proc. the 2005 IEEE/ACM International Conference on Computer-Aided Design,* San Jose, CA, 2005.

[6] Pham, D., Asano, S., Bolliger, M., Day, M., Hofstee, H., Johns, C., Kahle, J., Kameyama, A., Keaty, J., Masubuchi, Y., Riley, M., Shippy, D., Stasiak, D., Suzuoki, M., Wang, M., Warnock, J., Weitzel, S., Wendel, D., Yamazaki, T., Yazawa, K.: The design and implementation of a first-generation cell processor. Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International (Feb. 2005) 184–592 Vol. 1

[7] M. B. Taylor et al. The RAW microprocessor: A computational fabric for software circuits and general purpose programs. *IEEE Micro*, 22(2), 2002.

[8] Kumar, R., Tullsen, D.M., Jouppi, N.P., Ranganathan, P.: Heterogeneous chip multiprocessors. Computer **38**(11) (2005) 32–38

[9] Kumar, R., Tullsen, D.M., Ranganathan, P., Jouppi, N.P., Farkas, K.I.: Single-isa heterogeneous multicore architectures for multithreaded workload performance. In: ISCA '04: Proceedings of the 31st annual international symposium on Computer architecture. (2004) 64

[10] Corezilla: build and tame the multicore beast. (2007) Session Chair-Markus Levy and Session Moderator-Lauren Sarno and Session Panelist-Gordon Cameron and Session Panelist-Wen-mei W. Hwu and Session Panelist-James R. Larus and Session Panelist-Christopher K. Lennard and Session Panelist-Craig Lund and Session Panelist-James Reinders and Session Panelist-Takashi Yoshimori.

[11] Virtutech Simics. http://www.virtutech.com/