Architectural Simulation in the Kilo-core Era

R. Giorgi
University of Siena
Department of Information Engineering
giorgi@dii.unisi.it

A. Scionti
University of Siena
Department of Information Engineering
scionti@dii.unisi.it

A. Portero
University of Siena
Department of Information Engineering
portero@dii.unisi.it

P. Faraboschi
HP Labs - Exascale Computing Lab
paolo.faraboschi@hp.com

ABSTRACT

The continuos improvements offered by the silicon technology make possible the integration of always increasing number of cores on a single chip. Following this trend, it is expected to approach microprocessor architectures composed of thousands of cores (i.e., *kilo*-cores architectures) in the next future. In this context simulation tools represent a crucial factor for designing architecture at such core scale. This paper proposes a framework based on the COTSon simulator [2], and able of scaling towards heterogeous kilo-cores architectures. Compared with current state-of the-art architectural simulators, the proposed framework provides full-system simulation, a well balanced trade-off between simulation speed and accuracy, and the support for power consumption estimation. Experimental results confirm the ability of the framework to scale up at least to 2000 cores[1].

Categories and Subject Descriptors

B.2.2 [**Arithmetic and Logic Structures**]: Performance Analysis and Design Aids—*Simulation, Verification, Worst-case analysis*; B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids—*Simulation, Verification, Worst-case analysis*

General Terms

Measurement, Performance

Keywords

Many-core system simulation

1. INTRODUCTION

Looking at the Moore's Law, the continuos improvements offered by silicon technology make possible placing always increasing number of transistors on a single chip. Following this trend, it is expected that future high performance microprocessor systems will be composed of thousand of cores (i.e., *kilo*-cores architectures) connected each other through a high bandwidth network-on-chip. Matching the performance request with the power consumption requirement will bring to a massive adoption of heterogeneous architectures. More than in the past, the adoption of architectural simulators has become essential for assuring the correctness of the design. Architectural simulators historically suffered from low simulation speed and accuracy, imposing serious limitations on the ability of predicting correct behaviors of the designed architecture, especially in the many-core era. With the aim of providing a tool characterized by a high simulation speed and accuracy for an heterogeneous kilo-core architecture, this paper proposes a framework based on the COTSon infrastructure. Compared with current state-of-the-art simulation platforms, the proposed one offers a complete environment for a full-system simulation and for the power consumption estimation. In order to guarantuee fast simulations, the framework implements a *functional-directed* approach, where functional emulation is alternated to a complete timing-based simulation. The result is the ability of supporting the full stack of applications, middlewares and OSs. The adoption of the open source Qemu [3] based functional emulator opens the door to the support of several core architectures. Finally, the integration of the proposed framework with the McPAT tool [4], provides the ability of predicting the power consumption for a given design.

[1]This work has been supported by the IST-FET project TERAFLUX, funded under the EU's 7th Framework Programme.

2. PREVIOUS WORK

Recent works proposed different approaches aiming at the simulation of kilo-core systems. In [5] the authors present HORNET, an architectural simulator based on a highly configurable network-on-chip engine. It may use processor traces, execute a MIPS-based timing simulation or execute native code using the Pin instrumentation tool. Although it targets kilo-core systems, it exploits a network oriented design, with limited support for accurate heterogeneous core timing simulations and the inability of running full-system simulations. Similarly, in [8] the authors describe Multi2Sim, a framework for multi-threaded multi-core processor simulations. It is composed of a functional emulation and a timing-simulation layers. The main limitations are given by the only support to the MIPS architecture and the inability of supporting full-system simulation. MARSS-x86 [7]

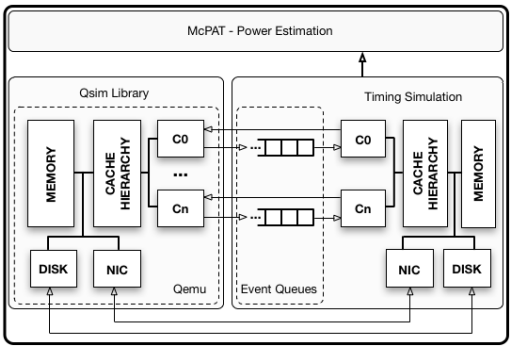

Figure 1: The proposed kilo-core simulation framework

is a fast cycle-accurate full-system simulator for heterogeneous architectures. It combines the functional emulation provided by Qemu with the timing simulation of PTLsim [9]. While the Qemu emulation makes possible to target several core architectures, PTLsim only offers accurate timing models for the x86-64, requiring additional code to target different systems. Furthermore, PTLsim has not been designed to work on a kilo-core scale. Furthermore, neither Multi2Sim nor MARSS-x86 are able to estimate power consumption. MANIFOLD [1] is a simulator designed with a similar approach to our proposed framework. It decouples functional and timing simulations. Functional simulation is based the Qsim library, while the timing model is based on a dedicated module. It supports also the integration of thermal and power modeling, however it appears to a premature stage of development. Finally, GRAPHITE [6] is a multicore simulator designed to provide high level of simulation performance by distributing the simulation workload. Although its ability of scaling up to kilo-core architectures keeping a fast simulation process, it does not provide an accurate simulation model, only providing accurate estimation methods.

3. COTSON-BASED FRAMEWORK

Figure 1 sketches the architecture of the proposed simulation framework. It is composed of three main blocks connected each other. The left side of the figure shows the functional emulation block, formed by the open source Qemu full-system emulator. We opted for this open source functional layer, since it offers the largest architectural support (e.g., x86-64, Microblaze, MIPS, PowerPC, etc.), allowing us to simulate heterogeneous architectures. The right side of the figure shows the timing simulation block. For each component (i.e., cores, caches, memory, disks and network interfaces) of the target architecture, there is a specific timing model. Periodically, the functional emulation is paused in order to perform the timing simulation. The timing simulator block collects a set of events (e.g., instruction counts, memory acceses, etc.) from the functional emulator, and uses its accurate internal models to adjust the global simulation time and the speed of the functional emulation. For correctly managing events produced by a many-core target architecture, a set of event queues are implemented. The support for a kilo-core system is given by interconnecting a large number of *nodes*. Each node is compose of a certain number of cores with their cache hierarchies, memory blocks, network interfaces and disks. The nodes are connected each other through their network interfaces. Properly setting the timing model of the network interfaces it is possible to correctly simulate the presence of a network-on-chip. With the aim of decoupling functional emulation from timing simulation, we designed a communication interface, that simplifies exchanging events and feedbacks information. On the functional emulator side we exploit *Qsim*, a library that simplifies the extraction of events from the Qemu emulator. As previously mentioned, one of the main issues when design an architectural simulator concerns the trade-off between simulation speed and accuracy. Since timing simulation is a time consuming process that causes the main slowdown, we choose to limit the timing simulation only to those application phases of interest. This goal is accomplished by implementing a sampling mechanism: the functional emulation is monitored, and whenever one of the phases is reached the timing simulation is enabled. Finally, with the aim of providing power consumption estimation for the targeted system, we directly interfaced our framework with the McPAT tool. The power consumption estimation can be performed both at the end and during the simulation process, by querying and processing simulation events collected in a local database.

4. EXPERIMENTAL RESULTS

In order to show the capability of the proposed framework to efficiently simulate target systems on the scale of 1000-cores, we implemented a parallelized matrix multiplication algorithm. As an initial experiment, we correctly run the benchmark on a system composed of 32 nodes, each of them composed of 32 cores (i.e., for the experiments we used standard x86-64 cores). We run the matrix multiplication using two matrices of 1024 elements. Further experiments demonstrate that the system can easily target the matrix multiplication algorithm on a system composed of up to 2000-cores.

5. CONCLUSIONS

This work presents a simulation framework targeting future heterogeneous kilo-core architectures. The framework decouples the functional emulation from the accurate timing simulation. Enabling timing simulation only for the application phases of interest, allows exposing a well balanced trade-off between simulation speed and accuracy. Thanks to the large number of supported architectures by the open source emulator, and to a core clustering approach, it easily supports the simulation of heterogeneous systems. Experimental results confirmed the ability of simulating future kilo-core systems.

6. REFERENCES

[1] www.ece.gatech.edu/research/labs/casl/multicore.html.
[2] E. Argollo and et al. Cotson: Infrastructure for full system simulation. *SIGOPS Oper. Syst. Rev.*, 43(1):52–61, January 2009.
[3] F. Bellard. Qemu, a fast and portable dynamic translator. In *Proceedings of the 2005 USENIX Annual Technical Conference*, 2005.
[4] S. Li and et al. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual*

International Symposium on Microarchitecture, pages 469–480. IEEE/ACM, December 2009.

[5] M. Lis and et al. Scalable, accurate multicore simulation in the 1000-core era. In *ISPASS*, pages 175–185, April 2011.

[6] J. E. Miller and et al. Graphite: A distributed parallel simulator for multicores. In *HPCA*, 2010.

[7] A. Patel and et al. Marss-x86: A qemu-based micro-architectural and systems simulator for x86 multicore processors. In *1st International Qemu Users' Forum*, pages 29–30, 2011.

[8] R. Ubal and et al. Multi2sim: A simulation framework to evaluate multicore-multithreaded processors. In *SBAC-PAD*, pages 62–68, 2007.

[9] M. T. Yourst. Ptlsim: A cycle accurate full system x86-64 microarchitectural simulator. In *ISPASS*, pages 23–34, April 2007.