A module-based Cell Processor Simulator

Felipe Cabarcas[*,1,3], Alejandro Rico[*,1],
David Rodenas[†,2], Xavier Martorell[*,†,2],
Alex Ramirez[*,†,2], Eduard Ayguade[*,†,2]

[*] *Universitat Politecnica de Catalunya, Barcelona, Spain*
HiPEAC European Network of Excellence
[†] *Barcelona Supercomputing Center (BSC-CNS), Jordi Girona 31, Torre Girona building Campus Nord, 08034 Barcelona, Spain*

ABSTRACT

An interesting design alternative to replication-based chip multiprocessors is to create heterogeneous chip multiprocessors composed of several different cores, with one or more of them running the operating system and orchestrating execution, and the others serving as accelerators where parts of the application are off-loaded.

We are developing a simulator for this kind of heterogenous architectures, using the Cell Broadband Engine as a first model and the UNISIM modular infrastructure. Thanks to UNISIM, the modules composing the simulator can be easily changed and replaced by others, allowing us to customize the processor and explore the design space for these emerging architectures.

KEYWORDS: Simulator; Cell processor; heterogeneous multiprocessor; UNISIM.

1 Introduction

Multi-core processors are a design solution adopted by the majority of manufacturers to exploit the amount of resources (transistors) that current technology provides for a single chip. Nowadays, the common designs replicate the same processor to try to benefit the parallelism existent in the applications. An alternative approach consists on building heterogeneous chip multiprocessor, where the different types of cores in the chip are specialized to accelerate some kind of tasks. An example of this trend is the Cell processor [KDH+05], which is one of the first commercial available heterogeneous chip multiprocessors.

We are building a Cell processor simulator with the purpose of having a heterogeneous chip multiprocessor research infrastructure. This approach has the advantage of having a real processor and a complete infrastructure built around it that provides a stable platform from which develop new ideas in research projects.

[1]E-mail: {cabarcas,arico}@ac.upc.edu
[2]E-mail: {david.rodenas,xavier.martorell,alex.ramirez,eduard.ayguade}@bsc.es
[3]Cabarcas is professor at the *Universidad de Antioquia*, Medellín, Colombia.

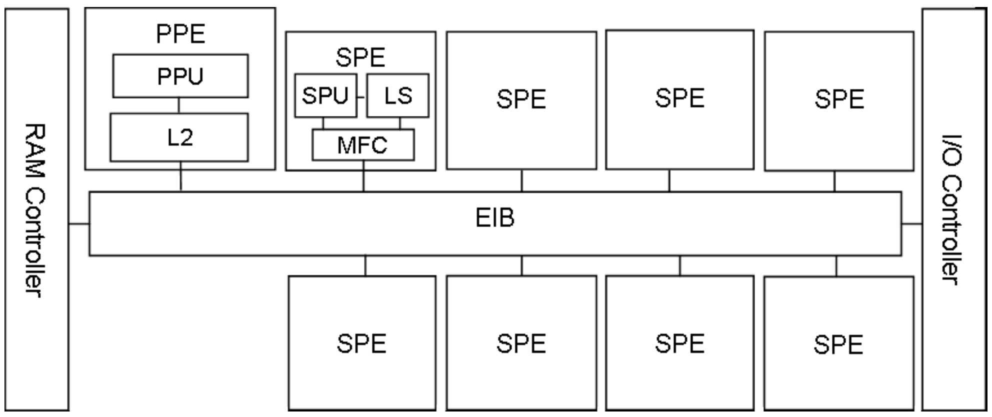

Figure 1: Simplified block diagram of the Cell processor.

The simulator is being written using the UNISIM environment [uni], which allows defining the architecture as independent modules that are easily connected through signals. Thus, modifying the simulator to test new ideas will be as easy as adding, exchanging or redistributing modules and/or connections.

2 The Cell Processor

The Cell processor is a joint initiative of Sony, Toshiba and IBM. The first version of the Cell is the Cell Broadband Engine, which can be seen as a processor specialized in gaming, since it is the main processor of the Sony PlayStation3.

As shown in Figure 1, this processor is composed by a Power-Architecture-compliant *Power Processor Element* (PPE) and eight *Synergistic Processor Elements* (SPE). These cores are connected through an *Element Interconnection Bus* (EIB), which also connects them with the memory and I/O controllers. The main component of the PPE is a dual-threaded, dual-issue, 64-bit *PowerPc Processor Unit* (PPU). Each SPE is composed by three elements: a *Synergistic Processor Unit* (SPU) [FAD+06], a *Local Store* (LS) and a *Memory Flow Controller* (MFC).

The PPU is an in-order processor, fully Power compliant *instruction set architecture* (ISA) and *single-instruction-multiple-data* (SIMD) instructions. This core is the responsible to run the operating system, and acts as the master of the system.

The SPU is also in-order, and has a newly architected ISA, completely SIMD. The data-flow path is 128-bit wide while the register file contains 128 general-purpose registers of 128-bits each one (that can be accessed as 16 bytes, 8 half-words, 4 words, 2 double-words or 1 quad-word).

The memory-mapped LS provides 256 Kbytes for both instructions and data. Each SPU has only direct access to its own LS, if it wants to communicate to another element outside the SPE, it has to perform a *direct memory access* (DMA) transaction through its MFC.

The MFC is fundamentally a DMA controller. Each MFC contains memory mapped control registers that allow the PPE, and other SPEs, to control the state of the SPU. Through these registers it is also possible to send word-long messages (mailbox), configure DMA operations, etc. [cbe].

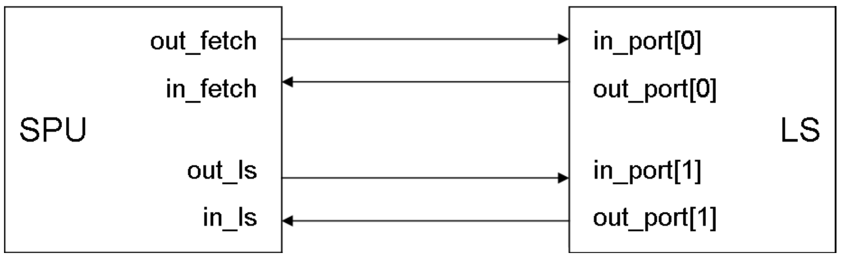

Figure 2: Communication interface between the SPU and LS modules.

3 UNISIM

UNISIM is a hardware simulator environment. It provides a common set of rules for module development. UNISIM allows defining a hardware architecture through a set of modules and their conections using signals. This way it is posible to encapsulate a hardware component into a reusable module.

The reusability of the modules is based on a well defined communication protocol. The module connection signals are composed of: *data*, *accept*, and *enable*. A communication transaction begins by sending the data, the receiving module can accept it or not, and the sender can enable or disable the transaction. Thus, a centralized control is, in this infraestructure, distributed among all connection signals.

4 The simulator

We are developing a functional execution-driven Cell processor simulator. We decided the simulator to be functional because its short term goal is to perform research at architectural level. If micro-architectural level detail is needed, it would simply require a new detailed module to be developed, and replacing our functional module with it. The rest of the simulator would remain unchanged.

In a first stage of design, we defined the structure of the simulator as a set of UNISIM modules corresponding directly to the modules that compose the Cell, Figure 1. Thanks to the modularity that UNISIM provides, the work was divided among two teams: one working on the design and implementation of the PPE and the memory hierarchy, and the other on the SPE. The teams meet regularly to show the progress, discuss the problems and agree on the communication protocol.

The modules are being developed making them easily configurable and interchangeable. Every module can be instantiated with concrete features and the interfaces between them are well-defined to have a pattern of the behavior that a substitute module must have.

As an example, in Figure 2 we show the communication interface between the SPU and the LS modules. The *out_fetch* port is connected to the first *in_port* of the LS to request the next instructions to execute. The SPU receives the instructions through the *in_fetch* port. The memory accesses resulting from a load or a store are computed through the *out_ls* and the *in_ls* ports of the SPU. The LS is multiported and the number of ports can be defined when the module is instantiated.

5 Current Status

As of June 2006, the PPE and the SPE work separately and they will be easily joined thanks to the unique interface between modules that has been defined. The PPE and the SPE are finished at 40%. The main memory is totally finished, and the bus and the rest of levels of the memory hierarchy are under development.

The PPE module supports 59 instructions and 14 system calls. It is able to execute the "Hello World!" program. The SPE has 27 instructions implemented of the 186 total, and executes memory instructions and several arithmetic and control flow operations.

Some profiling results have shown that the simulator is executing UNISIM procedures for synchronization tasks about the 90% of the time, and the 10% is performing the actual emulation. The PPE and the SPE execute, separately, about 30.000 instructions per second. This seems to be very slow but we hope that it will be scalable and will not slow down as the simulator grows.

6 Acknowledgements

We want to thank Sylvain Girbal, Olivier Temam and Daniel García-Pérez of the UNISIM developing team for being at our disposal whenever we have needed.

Supported, in part, by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D058240CO; by the DURSI of the Generalitat de Catalunya and the European Social Fund, scholarship No. FI-2006-01133; and by the EU-funded Integrated Project SARC Scalable Computer Architecture Contract Number 027648.

References

[cbe] Cell broadband engine architecture, version 1.0. http://www-128.ibm.com /developerworks/power/cell/.

[FAD+06] Brian Flachs, S. Asano, S. Dhong, P. Hofstee, G. Gervais, R. Kim, T. Le, P. Liu, J. Leenstra, J. Liberty, B. Michael, H.-J. Oh, S. Mueller, O. Takahashi, A. Hatakeyama, Y. Watanabe, N. Yano, D. Brokenshire, M. Peyravian, V. To, and E. Iwata. The microarchitecture of the synergistic processor for a cell processor. *IEEE Journal of Solid-State Circuits*, 41(1), 2006.

[KDH+05] J. A. Kahle, M. N. Day, H. P. Hosftee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the cell multiprocessor. *IBM Journal of Research and Development*, 49(4/5), 2005.

[uni] The unisim site. http://www.unisim.org.