CellSim: A Cell Processor Simulation Infrastructure

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

[*] *Dept. Arquitectura de Computadors, Universitat Politècnica de Catalunya, Barcelona, Spain*
[†] *Barcelona Supercomputing Center – Centro Nacional de Supercomputación, Barcelona, Spain*

ABSTRACT

CellSim is a modular simulator for heterogeneous multiprocessors based on the UNISIM infrastructure. It can be configured to model IBM's Cell Processor. The simulator has been validated against the real machine in order to be used as a valid research tool.

The modularity of CellSim allows doing research on heterogeneous multiprocessors. CellSim can be parametrized to measure the impact of different features on the Cell. Moreover, different architectures can be modeled by assembling specific modules.

KEYWORDS: Heterogeneous chip multiprocessors; simulation; Cell Processor

1 Introduction

There is an agreement between the industry and the research community about the importance of Heterogeneous Chip Multiprocessors (HCMPs) in the future. On the one hand the computing industry led by IBM, Toshiba and Sony has developed the Cell processor [KDH+05], Intel has recently announced the Larrabee project [int07], and the embedded industry has been developing heterogeneous multiprocessors for several years, like the TI's OMAP, or the Nexperia by NXP. On the other hand, the research community [ABC+06, Pat] has shown that the future multiprocessors should be heterogeneous.

However, the success of HCMPs depends on the ability of software designers and computer architects to develop strategies to extract parallelism from applications and to use them to effectively drive the available hardware [ABC+06]. Hence, it will be required that simulators not only model application traces, or statistics, but also the dynamic behavior of programs. Researchers, then, can find bottlenecks and strategies to exploit more parallelism.

[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.

This poster presents a validated Cell Broadband Engine simulator (CellSim). CellSim is a modular simulator that not only can simulate the Cell Processor but also a wide range heterogeneous multiprocessor architectures. Section 2 contains the description of CellSim, and 3 presents the validation process we have carried out.

2 CellSim Structure

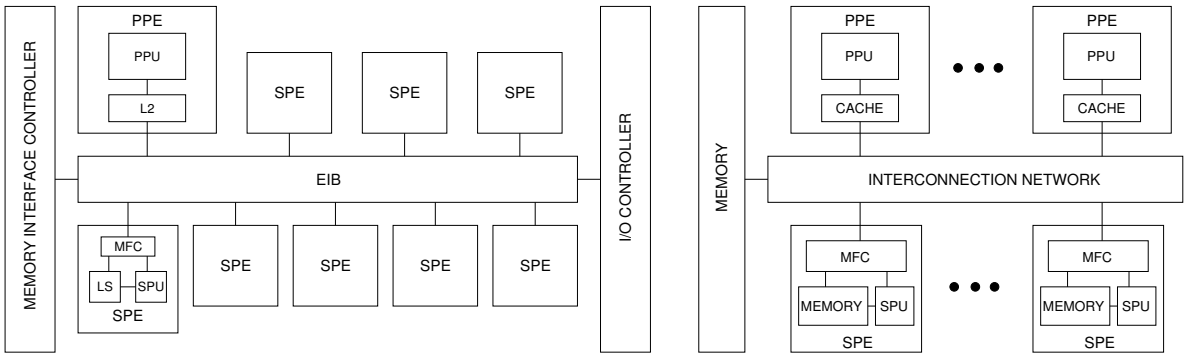
CellSim is a modular simulator for heterogeneous multiprocessors. Its modularity is based on the fact that it is composed by modules, which are connected among them through ports. We have used UNISIM [uni] as the supporting infrastructure for module description and connection handling.

We have implemented the modules corresponding to the current version of the Cell Broadband Engine Architecture, the Cell Processor. Using these modules we have configured CellSim in order to be able to validate it against the real processor.

As can be seen in Figure 1(a), the Cell processor is a multiprocessor composed of one PowerPC Processing Unit (PPU), with its cache (L2), and eight synergistic processing elements (SPE) composed of a Local Store (LS) a Synergistic Processing Unit (SPU) and a Memory Flow Controller (MFC).

The modules of the simulator, as it is shown in Figure 1(b), represent each hardware component of the Cell Processor in Figure 1(a). The modules that compose the simulator are the following: PPU, Cache, SPU, MFC, Memory and Interconnection Network. It is possible to connect one PPE, 8 SPEs, and a Memory module to an Interconnection Network to obtain the same configuration with respect to the Cell Processor. However, it is also possible to connect more PPEs and SPEs to create different architectures.

In order to create a common interface for the simulator, we have developed the *MemoryAccess* class, which is the information packet that is transmitted between modules. A MemoryAccess object includes the type of access (*Load* or *Store*), the target and source addresses, the number of bytes and, if it is a store, the data. This common interface combined with the fact that each module has a physical page range assigned, provides the necessary flexibility to have a modular simulator.



(a) Cell (b) CellSim

Figure 1: Cell and CellSim block diagram.

The Memory module is used both as main memory and as LSs. It does not implement any specific internal distribution of memory banks, so sequential accesses can be overlapped.

Moreover, the accesses latency and the number of ports can be configured. Due to this, the Memory module can also be used, at the SPE, to behave as the LS, which have several ports connected to both the SPU and the MFC.

The PPE is formed by two modules: PPU and Cache. The PPU is a PowerPC 405 32-bit architecture compliant instruction set simulator (ISS). It can execute a parametrizable number of instructions and instruction fetch per simulator cycle. Memory requests go through the Cache connection. Since CellSim is not a full system simulator, the PPU module is responsible for emulating the operating system services: loading and execution of elf files, system calls, start and stop SPEs, mapping logical addresses onto physical ones, etc.

The SPE is composed of three modules: SPU, Memory and MFC. The SPU is an ISS, that executes parametrizable number of instructions per simulator cycle. It has two connections to the Local Store (memory) module, one for instruction fetch and the other for Loads and Stores. Channel requests are serviced through one of the connections with the MFC, while the second connection is used to report the state of the SPU and to start and stop it when requested by the PPU. The Local Store services the SPU and the MFC load and store requests. The direct memory access (DMA) command related services are handled by the MFC, which also contains the memory mapped registers of the SPEs.

The EIB of the Cell processor is a very efficient interconnection network, however it is not scalable. Therefore we have decided to implement the Interconnection Network (IN) module with a K-Bus topology. Although it is significantly different than the EIB, we have demonstrated that it is flexible enough to be configured and simulate a performance close to the Cell Processor (Section 3).

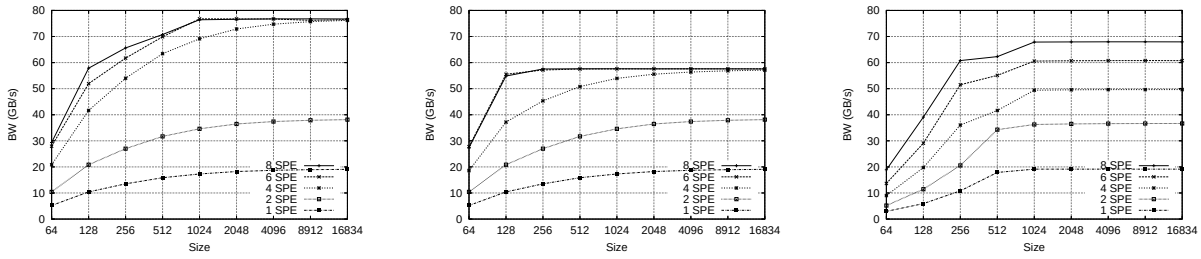
3 CellSim Validation

The timing validation has been performed in two phases. In the first phase we validated the SPU timing and, we did an overall validation in the second. For both phases, we used the timebase registers in the PPE and in the SPEs (*Decrementers*).

Since the PPU and SPU performances can be tuned using parameters (Issue bandwidth), the overall validation concentrates on the MFC and IN timings. For this task we chose the programs presented in Daniel Jimenez' EIB bandwidth study [DXA07]. These programs stress the EIB by performing continuous DMA transfers to analyse the bandwidth the Cell's interconnection network provides. One of these experiments is the SPE-SPE cycle program which sends data among LSs of several SPEs distributed in a cycle (from SPE i to SPE $(i+1)$ mod $total\_spes$).

To carry out the overall validation, we have executed Daniel Jimenez' programs in a Cell Processor at 2.4 GHz and in CellSim using the following parameters: one PPE that executes 1 instruction per cycle using a Cache with 512 lines of size 128 bits and 8 ways. Eight SPEs executing 2 instructions per cycle, each of them using a 256-KByte LS with a latency of 6 cycles and 3 ports. The MFCs were configured with a command queue size of 16 and delay 30 cycles. Finally, the EIB was configured with 4 (and 3) buses, each having a bandwidth of 8 bytes per bus cycle and handling up to 16 outstanding transfers per node.

Figure 2 present the bandwidth for the SPE-SPE cycle experiment. The Cell Processor (Figure 2(c)) assigns any physical SPE to a logical SPE, so it does not guarantee that SPE i is physically next to SPE $i+1$. This fact generates a lot of contention in the EIB for the cycle program, which results in a bandwidth lower than CellSim with 4 buses (Figure 2(a)), where



(a) CellSim SPE-SPE cycle 4 buses (b) CellSim SPE-SPE cycle 3 buses (c) Cell SPE-SPE cycle

Figure 2: Interconnection network bandwidth results of Daniel Jimenez' programs. SPE-SPE programs transfer data between SPEs. SPE-SPE cycle distributes SPEs in a cycle (SPE $i \rightarrow$ SPE $i + 1$).

the physical SPE layout does not have any effects. In order to obtain a bandwidth closer to the real machine in this experiment, we adjust the number of buses in the IN to 3 so as to decrease CellSim's bandwidth (Figure 2(b)).

Acknowledgements

This work has been supported by the European Commission in the context of the SARC integrated project (EU contract 27648-FP6), and the Spanish Ministry of Education under contract CICYT TIN2004-07739-C02-01. It has been also supported by the Programme AlBan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D058240CO; and by the Spanish Ministry of Education, scholarship No. AP2005-4245.

References

[ABC+06] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 18 2006.

[DXA07] Daniel Jimenez-Gonzalez, Xavier Martorell, and Alex Ramirez. Performance Analysis of Cell Broadband Engine for High Memory Bandwidth Applications. In *ISPASS 2007: Proc. of the IEEE International Symposium on Performance Analysis of Systems and Software*, April 2007.

[int07] Innovation, processing performance and cooperation key to moving forward faster. INTEL DEVELOPER FORUM, April 17 2007.

[KDH+05] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the Cell multiprocessor. *IBM Journal of Research and Development*, 49(4-5):589–604, July 2005.

[Pat] Yale N. Patt. Guest Speaker at the IEEE Computer Society 60th Anniversary Reception in San Juan, Puerto Rico, 14 June.

[uni] https://unisim.org/site/. UNISIM.