Efficient and Scalable Barrier Synchronization for Many-Core CMPs

José L. Abellán
Dept. de Ingeniería y Tecnología de Computadores
Facultad de Informática - Universidad de Murcia
30100 Murcia, Spain
jl.abellan@ditec.um.es

Juan Fernández
Dept. de Ingeniería y Tecnología de Computadores
Facultad de Informática - Universidad de Murcia
30100 Murcia, Spain
juanf@ditec.um.es

Manuel E. Acacio
Dept. de Ingeniería y Tecnología de Computadores
Facultad de Informática - Universidad de Murcia
30100 Murcia, Spain
meacacio@ditec.um.es

ABSTRACT

We present in this work a novel hardware-based barrier mechanism for synchronization on many-core CMPs. In particular, we leverage global interconnection lines (*G-lines*) and *S-CSMA* technique, which have been used to overcome some limitations of a flow control mechanism (EVC) in the context of Networks-on-Chip, to develop a simple *G-line*-based network that operates independently of the main data network in order to carry out barrier synchronizations. Next, we evaluate our approach by running several applications on top of the Sim-PowerCMP performance simulator. Our method only takes 4 cycles to carry out the synchronization once all cores or threads have arrived at the barrier. Hence, we obtain much better performance results than software-based barrier implementations in terms of scalability and efficiency.

Categories and Subject Descriptors

B.0 [**Hardware**]: General; C.1.4 [**Processor Architectures**]: Parallel Architectures

General Terms

Design, Performance

1. INTRODUCTION

Nowadays, Chip-multiprocessors (or CMPs) constitute the best way to take advantage of the increasing number of transistors available in a single die, since they provide higher-performance and lower-power than complex uniprocessor systems exploiting thread-level parallelism (TLP). In fact, following the well-known Moore's Law, it is clear that more and more cores will be integrated in future CMP layouts even reaching hundreds of them all integrated in the same chip. CMPs of this kind are commonly referred to as many-core CMPs.

To extract high performance from such architectures, applications must be divided into multiple threads which need to communicate and synchronize among them. Currently, most of CMP designs implement a shared-memory programming model: communication is directly supported by hardware (conventional memory access instructions: loads and stores); and synchronization is typically supported by a combination of atomic instructions (*test&set* or *LL/SC*) to implement higher-level mechanisms (locks/unlocks or barriers). These software-based implementations typically imply busy-waiting on shared variables in memory and introduce significant performance bottlenecks which limits scalability [3]. Thus, a lot of hardware-based approaches have arisen to overcome such limitations in the context of multiprocessors.

CF'10, May 17–19, 2010, Bertinoro, Italy.
ACM 978-1-4503-0044-5/10/05.

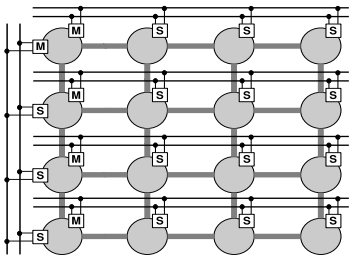


Figure 1: *G-lines* architecture for a 16-core CMP and 2D-mesh network.

Our proposal is a hardware-based approach but in the context of many-core CMPs. This is based on *global interconnection lines* (*G-lines* from now on) and a technique referred to as *smart carrier sense multiple access* (*S-CSMA*) [5]. In short, the *G-lines* provide a one cycle broadcast across the chip between one transmitter and one receiver on a per-bit granularity. The *S-CSMA* technique enables one receiver to sense the number of transmitters utilizing the line at any given clock cycle. However, this new architecture has been used in the context of networks-on-chip (NoC) [5] to enhance a flow control mechanism (EVC) in terms of latency and power consumption.

2. *G-LINE*-BASED BARRIER

Our approach is graphically outlined in Figure 1 for a 4x4 mesh network. As we can see, the *G-line*-based network interconnects all cores through two sort of controllers, namely Master (M) and Slave (S). Each controller is attached to two *G-lines*: one of them is used to transmit signals and the other one for receiving signals. Moreover, the Master controller is responsible for carrying out the count of signals across its *G-line*. It performs the accounting by means of a device which implements the *S-CSMA* technique.

Next, we explain the process of barrier synchronization for any given many-core CMP with a 2D-mesh network (see Figure 2). Without loss of generality, we assume that all

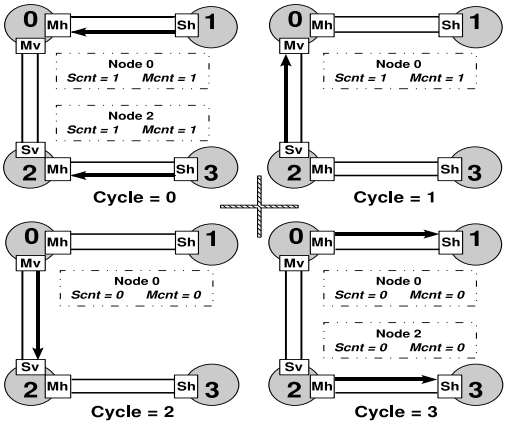


Figure 2: Steps for our barrier synchronization.

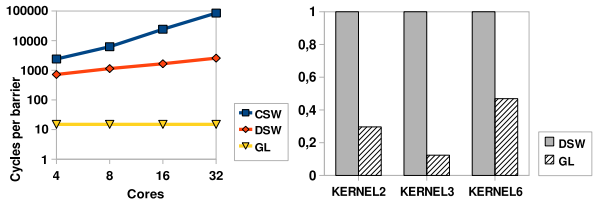
cores execute the same barrier at the same time, and we describe the explanation on a 2x2 mesh layout.

The process is as follows. At cycle 0, the horizontal Slaves (Sh) signal, through its corresponding *G-line*, the arrival of cores 1 and 3 at the barrier and wait until their horizontal Masters (Mh) command to resume execution, by monitoring the other *G-lines*. Then, the horizontal Masters count the number of received signals and update their counters *Scnt*=1 (because there is just one Slave for each), besides they also set the counter *Mcnt* to 1 due to the arrival of cores 0 and 2 at the barrier. At cycle 1, upon each horizontal Master updates both counters to its maximum values, its corresponding vertical Slave (Sv) repeats the process. In particular, the vertical Slave writes into its *G-line*, and the vertical Master (Mv) updates the counter *Scnt*=1, but also sets the counter *Mcnt* to 1 because their corresponding horizontal Master (in core 0) have updated its *Mcnt*=1. At cycle 2 the release stage starts in order to resume execution by using the unused *G-lines*. To do so, the vertical Master signals its vertical Slave and resets all counters. Finally at cycle 3, the horizontal Masters do the same on their corresponding horizontal *G-lines* to wake up the horizontal Slaves.

3. EVALUATION

As testbed, we have extended the Sim-PowerCMP performance simulator [2] to support our hardware-based barrier. As benchmarks, we consider a synthetic benchmark and various kernels from Livermore Loops [1] following the same recommendations given in [4]. The synthetic benchmark consists of a loop of four consecutive barriers with no work or delays between them executed 100,000 times. Regarding the latter, we focus on the following kernels: Kernel 2, which is an excerpt from an incomplete Cholesky conjugate gradient code; Kernel 3 is a simple inner product; and finally, Kernel 6 is a general linear recurrence equation. Furthermore, we compare our hardware-based barrier with two software-based implementations [3]: a centralized sense-reversal barrier based on locks (or CSW); and a binary combining-tree or distributed barrier (DSW).

From the results presented in Figure 3, we can derive the following appreciations. On the one hand, Figure 3(a) shows the very efficient DSW barrier as opposed to the CSW version. And second, it is clear that our mechanism highly outperforms the others in both efficiency and scalability. On the other hand, Figure 3(b) shows the average normalized times for all kernels executed over a 32-core CMP layout (1,000



(a) Synthetic benchmark (b) Livermore loops

Figure 3: Average times for different barrier mechanisms.

iterations and 1,024 elements). As we can see, comparing our approach and the best based-software barrier implementation (DSW), our hardware barrier mechanism reports much better performance than the software barrier. In more depth, we obtained the percentage of barrier time (through the statistics reported by Sim-PowerCMP) for each kernel executed with the DSW version: 81% (K2), 89% (K3) and 61% (K6). As a result, the improvements of our approach are clearly justified.

4. CONCLUSIONS AND FUTURE WORK

In this work, we have presented a novel hardware-based barrier mechanism for many-core CMPs based on a *G-line*-based network along with the use of the *S-CSMA* technique. We have measured the benefits of our approach by means of several applications running on top of Sim-PowerCMP. Our mechanism meets scalability, efficiency and hardware simplicity. As future work, we will measure the efficiency of our method in terms of power consumption and quantify the reduction in terms of network traffic. Moreover, we will multiplex in space (virtual hierarchies) and time the use of our hardware-based barrier in many-core CMPs.

5. ACKNOWLEDGMENTS

This work was supported by the Spanish MEC and MICINN, as well as European Comission FEDER funds, under Grants "CSD2006-00046" and "TIN2009-14475-C04". José L. Abellán is supported by fellowship 12461/FPI/09 from Comunidad Autónoma de la Región de Murcia (Fundación Séneca, Agencia Regional de Ciencia y Tecnología).

6. REFERENCES

[1] http://www.netlib.org/benchmark/livermorec.

[2] A. F. et al. Sim-powercmp: A detailed simulator for energy consumption analysis in future embedded cmp architectures. In *AINA Workshops (1)*, pages 752–757, May 2007.

[3] J. M. M.-C. et al. Synchronization without contention. In *ASPLOS*, pages 269–278, April 1991.

[4] J. S. et al. Exploiting fine-grained data parallelism with chip multiprocessors and fast barriers. In *MICRO*, pages 235–246, December 2006.

[5] T. K. et al. Noc with near-ideal express virtual channels using global-line communication. In *Hot Interconnects*, pages 11–20, August 2008.