Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2545 – 2548

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

# Efficient $Dir_0B$ Cache Coherency for Many-Core CMPs

José L. Abellán[a,*], Alberto Ros[b], Juan Fernández[c], and Manuel E. Acacio[b]

[a] *Electrical and Computer Engineering Department, Boston University, 02215 Boston-MA, USA*
[b] *Computer Engineering Department, University of Murcia, 30100 Murcia, Spain.*
[c] *Intel Labs Barcelona, 08034 Barcelona, Spain.*

**Abstract**

Directory-based cache coherency is commonly accepted as the design of choice to provide high performance and scalability in coherency maintenance for many-core CMPs. However, the on-chip area overhead required to encode sharer sets may compromise their success as core count increases. In this work, we propose the Express COherence NOtification (*ECONO*) protocol, a simple and efficient $Dir_0B$ cache coherence protocol that does not require sharer sets encoding while approaching performance of a conventional directory-based protocol. To accomplish that, *ECONO* relies on express coherence notifications which are broadcast atomically over a dedicated lightweight on-chip network leveraging state-of-the-art technology. Detailed full-system simulation using a representative set of benchmarks corroborates our statement.

*Keywords:* Many-Core CMPs, Cache Coherence, *G-Lines* Technology

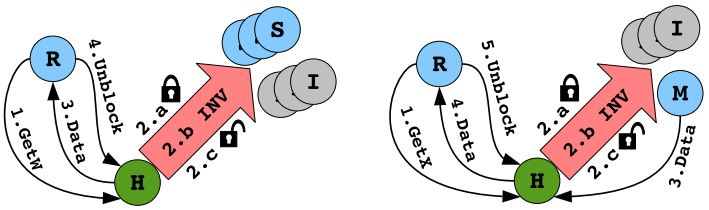
## 1. Introduction

Alternatively to snoopy-based coherence protocols, directory-based protocols constitute the most scalable and high performance choice for coherency maintenance [1] in future many-core CMPs [2]. Nonetheless, from the on-chip area standpoint, the extra hardware resources required by these protocols to encode information about block sharers may jeopardize their success in the context of many-core CMPs. Among all implementations based on directory schemes, we highlight that of the *Hammer* [3] protocol. This protocol achieves the highest on-chip area efficiency because it does not devote any extra hardware resources to encode a sharer list for every memory block. Following the terminology from Agarwal et al. [4], this is a $Dir_0B$ protocol. This comes at the expenses of an inefficient usage of the main CMP's interconnection network because, for ensuring coherence, it requires as many coherence messages (e.g.,invalidation of a cached block) as the number of all private caches in the system, thus making this implementation not really scalable.

In this paper, we propose *Express COherence NOtification* (*ECONO*) protocol, an efficient and simple cache coherence protocol specifically tailored to future many-core CMPs. *ECONO* is basically a directory-based protocol similar to *Hammer*, operating on a request-response operation mode, while approaching performance and scalability of a typical directory scheme with sharer encoding, i.e. a Dim NB protocol according to Agarwal et al. (*Directory* from now on). To accomplish that, our design ensures coherence by relying on *atomic broadcasts* (*ACN* messages) that are transmitted over a lightweight *dedicated on-chip network* (*GecNetworks*) leveraging *G-Lines* technology [5] for maximum performance.

*Corresponding author. Tel.: +1 (617) 697 9272 ; fax: +34 868 884151
*E-mail address:* jabellan@bu.edu.


Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.437




(a) Invalidation of block sharers (b) Data recovery

Fig. 1. Coherence maintenance in *ECONO*. *Messages depicted above fine black arrows travel through the main interconnection network, whereas thick pink arrows illustrate the ACN messages through the GecNetwork.*

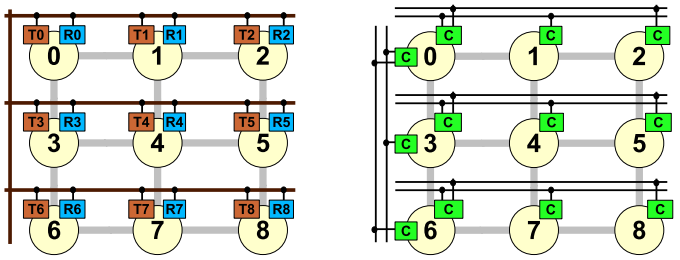
## 2. *ECONO* Cache Coherence Protocol

To illustrate how *ECONO* would operate, we assume a many-core CMP architecture composed of a number of replicated tiles where each tile contains a private L1 cache and a slice of a shared L2 cache (further details in Section 2.1). Moreover, as baseline implementation we consider MESI state machines for the L1 caches, and a very simple *request-response* operation mode in which home tiles do not delegate coherence responses to L1 caches (i.e., no forward messages are employed that would increase complexity at L1 caches and directory controllers). Additionally, every *ACN* message contains the address of the requested block and the type of *ACN* message (e.g., invalidation of block sharers).

Fig. 1 exemplifies how our proposal would operate under two typical scenarios for the maintenance of coherence: invalidation of block sharers, and data recovery from a block owner. Notice that, when coherence activity is not necessary our protocol would behave exactly as the *Hammer* protocol, conveying the required network transactions through the main CMP's network. The first scenario is depicted in Fig. 1a. The figure shows the case of a particular memory block that is being shared among different L1 caches (multiple copies of the block, each one in `S` state), and a requesting core (`R`) that gets a write miss in its L1 cache and sends a write request (`1.GetW`) to the home tile (`H`). In this case, the protocol must invalidate all cached copies before delivering the valid home's copy and write permission to the requester. In *ECONO*, since home tiles do not store a list of sharers for every memory block, a coherence operation similar to what is done in *Hammer* should be performed (see Fig. 1a). However, instead of sending as many invalidation messages as L1 caches in the system, *ECONO* broadcast a unique *ACN* message (see `2.b INV`) through the *GecNetwork*, thereby saving traffic from the main CMP's network and energy. And second, differently to *Hammer*, the requester does not waste time waiting for the acknowledgement messages because our coherence protocol can safely operate without them. It is accomplished by transferring the *ACN* messages atomically. Fig. 1b illustrates the second scenario. Here, there exists only one modified copy of the block in a single L1 cache (i.e., the owner or `M` in the figure) and a requesting core that wants to write or read (see `1.GetX` for the general case) the block. The process would be the same as before but now the owner would invalidate (for a write miss) or downgrade (for a read miss) its copy, and it would send the block to the home tile (`3.Data`).

### *2.1. Physical Implementation*

The *ECONO* protocol is comprised by the *GecNetwork* and the *GLock* (see Fig. 3). Both networks are implemented by assuming the *G-Lines* technology for superior efficiency. In short, every *G-Line* is a long wire that enables extremely fast 1-bit communications across one dimension of the chip. That is the reason why our on-chip networks will be composed of global 1-bit width links as building blocks. First, the *GecNetwork* is used to broadcast the *ACN* messages. This network is made up of 3-bit width horizontal and vertical *G-Lines*, and two types of controllers (Fig. 2a): `Tx`, that a particular home tile uses to transmit an *ACN* message; and `Ry`, that all the remaining tiles utilize (i.e., their corresponding L1 caches) to receive the transmitted *ACN* message. For superior efficiency, we configured two *GecNetworks* for *ECONO*. And second, the *GLock* (see Fig. 2b), that guarantees atomicity for the transmissions of the *ACN* messages. Notice that, multiple home tiles may compete for the *GecNetwork* ownership. We adapt our *GLock* proposal [6] that guarantees atomicity for highly-contended locks. The *GLock* is compounded of 1-bit width *G-Lines* and several controllers (`Cx`).




(a) *GecNetwork* (b) *GLock*

Fig. 2. Dedicated networks required for a 9-tile CMP. *Every circle represents a tile, thick gray lines constitute the main 2D-mesh interconnect, and finer lines are for the ECONO's networks with their respective controllers depicted as boxes.*

Table 1. CMP baseline configuration.

| | |
|---|---|
| 16 CPU Cores | 3 GHz, in-order, single-issue |
| Cache Hierarchy | Line: 64 Bytes; L1 I/D-Cache: 32 KB, 4-way, 2 cycles; L2 Bank: 512 KB, 8-way, 12+4 cycles; Memory: 250 cycles |
| Network | 2D-mesh; Bandwidth: 48 GB/s; Flit: 16 bytes; Link bandwidth: 1 flit/cycle; Data/Control size: 72/8 bytes |

Table 2. Benchmarks and input sizes.

| **Benchmarks and Input Size** |
|---|
| SPLASH-2: Barnes (8192 bodies, 4 steps); FFT (64K doubles); Ocean (258×258); Radix (1048576); Raytrace (Teapot) |
| PARSEC: Swaptions (simmedium) |
| Others: EM3D (38400, 2, 15%, 50 steps); Tomcatv (256 points, 5 steps); Unstructured (Mesh.2K, 5 steps) |

## 3. Evaluation

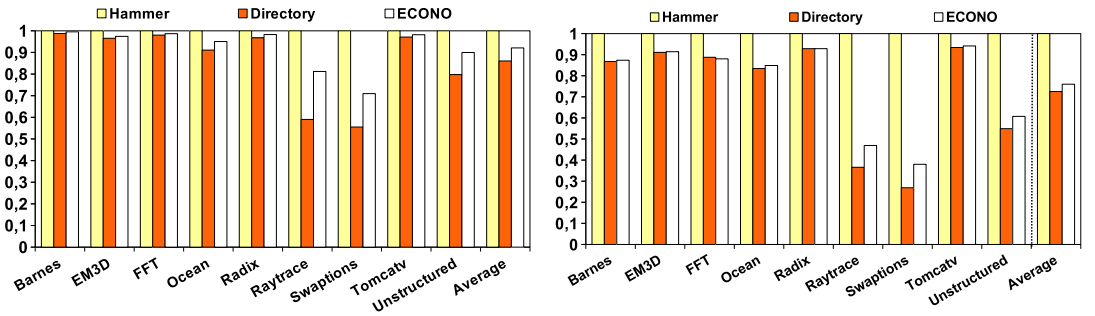
### *3.1. Methodology*

As testbed, we use full-system simulation by means of Virtutech Simics [7] (running Solaris 10) extended with Wisconsin GEMS toolset [8]. Table 1 summarizes the values of the main configurable parameters assumed in this work. As benchmarks, we use nine multi-threaded applications from both the SPLASH-2 [9] and PARSEC benchmark suites [10], and three additional scientific applications. Table 2 shows them and their respective problem sizes. These applications were chosen because they exhibit different communication patterns. All experimental results are for the parallel phase of the applications. Besides, we compare our *ECONO* protocol against the $\text{Dir}_0\text{B}$ *Hammer* protocol and *Directory* (we assume a full-map bit vector for encoding of sharers).

### *3.2. Performance Results*

Fig. 3a shows the normalized execution times with respect to those obtained when *Hammer* is considered. As expected, *Directory* achieves important reductions in execution time (14% on average) because of using precise coherence information which entails less coherence messages to be transmitted (e.g., Invalidations) and to wait for (i.e., Acknowledgements). As we can observe, *ECONO* implementation, by relying on a atomic broadcasts through a very fast *G-Line*-based on-chip network, achieves a considerable improvement over *Hammer* protocol (10% on average). Nevertheless, relying on a simple request-response operation mode, *ECONO* requires one more hop when data recovery is necessary (no forwards are allowed), and then, it cannot outperform *Directory*, that manages this situation much more efficiently than *Hammer* does. First, similar to *ECONO*, *Directory* would require a single coherence message sent to the particular owner that eventually would be in charge of sending the data to the requester. Second, similar to *ECONO*, the requester would not spend any time waiting for acknowledgement messages. Nonetheless, we approach performance of *Directory* (penalization of 4% on average).

Fig. 3b shows the total network traffic normalized with respect to *Hammer*. In particular, each bar plots the number of bytes transmitted through the interconnection network (the total number of bytes transmitted by all the switches of the interconnect). As expected, *Directory* outperforms *Hammer* (27% on average) because of using precise information of cached blocks that removes all unnecessary coherence messages from the interconnect. Regarding the *ECONO* protocol, since coherence actions are transmitted through the *GecNetwork*, we achieve




(a) *Normalized Execution Times* (b) *Normalized Network Traffic*

Fig. 3. Performance comparison of the different coherence protocols for a 16-core CMP. Results are normalized with respect to *Hammer*.

significant reductions in network traffic in comparison to *Hammer* (25% on average). Nonetheless, we can observe that it does not outperform *Directory*. The reason is that requiring indirection to the home tiles for data recovery (see Fig. 1b) implies more hops and then, more messages are injected into the main interconnect to convey data blocks to the requesting cores. However, network traffic is increased by only 3% on average.

## 4. Conclusions and Future Work

In this paper we propose *ECONO*, an efficient $Dir_0B$ coherence protocol for many-core CMPs. To keep coherence, our proposal relies on express coherence notifications (*ACN* messages) which are broadcast atomically over a dedicated lightweight on-chip network leveraging *G-Lines* technology. To quantify the performance benefits of our proposal, we compare *ECONO* against a commercial implementation of a $Dir_0B$ protocol (*Hammer*), and a conventional $Dir_nNB$ protocol (*Directory*). Execution-driven simulations of a 16-tile CMP reveal that *ECONO* significantly outperforms *Hammer* in both execution time and network traffic and obtains similar average performance results than *Directory*, although avoiding codification of sharer sets, thereby saving also on-chip area. As future work, we will improve efficiency of *ECONO* by considering forward messages to owner caches like in *Hammer* and *Directory* protocols. Moreover, we will study scalability, area and energy consumption of *ECONO*.

## Acknowledgements

This work was supported by the Spanish MINECO under grant TIN2012-38341-C04-03. This work was done while José L. Abellán was a PhD student in the Computer Engineering Department at the University of Murcia (Spain).

## References

[1] D. Sanchez and C. Kozyrakis., SCD: A Scalable Coherence Directory with Flexible Sharer Set Encoding, in: Proc. IEEE International Symposium on High-Performance Computer Architecture, 2012.
[2] J. P. Shin et al., A 40nm 16-core 128-thread CMT SPARC SoC Processor, in: Proc. the International Solid-State Circuits Conference Digest of Technical Papers, 2010.
[3] A. Ahmed et al., AMD Opteron Shared Memory MP Systems, in: Proc. HotChips Symposium, 2002.
[4] A. Agarwal et al., An Evaluation of Directory Schemes for Cache Coherence, in: Proceedings of the $15^{th}$ International Symposium on Computer Architecture, 1988.
[5] T. Krishna et al., Express Virtual Channels with Capacitively Driven Global Links, IEEE Micro 29(4) (2009) 48–61.
[6] J. L. Abellán et al., GLocks: Efficient Support for Highly-Contended Locks in Many-Core CMPs, in: Proc. IEEE International Parallel & Distributed Processing Symposium, 2011.
[7] P. Magnusson et al., Simics: A Full System Simulation Platform, IEEE Computer 35(2) (2002) 50–58.
[8] M. M. Martin et al., Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) toolset, Computer Architecture News 33(4) (2005) 92–99.
[9] S. C. Woo et al., The SPLASH-2 Programs: Characterization and Methodological Considerations, in: Proc. IEEE International Symposium on Computer Architecture, 1995.
[10] C. Bienia et al., The PARSEC benchmark suite: Characterization and Architectural Implications, in: Proc. IEEE International Conference on Parallel Architectures and Compilation Techniques, 2008.