**ECONO**: Express Coherence Notifications for Efficient Cache Coherency in Many-Core CMPs

José L. Abellán*, Alberto Ros†, Juan Fernández‡ and Manuel E. Acacio†

* Electrical and Computer Engineering Department, Boston University, 02215 Boston-MA, USA
Email: jabellan@bu.edu
†Computer Engineering Department, University of Murcia, 30100 Murcia, Spain
Email: {aros,meacacio}@ditec.um.es
‡Intel Labs Barcelona, 08034 Barcelona, Spain
Email: juan.fernandez@intel.com

Abstract—It is commonly stated that a directory-based coherence protocol is the design of choice to provide maximum performance in coherence maintenance for shared-memory many-core CMPs. Nevertheless, new solutions are emerging to achieve acceptable levels of on-chip area overhead and energy consumption to also meet scalability. In this work, we propose the Express COherence NOtification (ECONO) protocol, a coherence protocol aimed at providing high performance with minimal on-chip area and energy consumption for superior scalability. To maintain coherence, ECONO relies on express coherence notifications which are broadcast atomically over a dedicated lightweight and power-efficient on-chip network leveraging state-of-the-art technology. We implement and evaluate ECONO utilizing full-system simulation, a representative set of benchmarks, and compare it against two contemporary coherence protocols: Hammer and Directory. While ECONO achieves slightly better performance than Directory, our proposal does not need to encode sharer sets like in Hammer, saving significant on-chip area and energy even when considering the extra hardware resources required by ECONO. Projections for a 1024-core CMP reveal that, in comparison to one of the most scalable directory-based protocols to date, ECONO entails more than 2× less on-chip storage overhead while keeping with reasonable power dissipation.

I. INTRODUCTION

Many-core CMP architectures with a network-on-chip (NoC) have emerged as the next-generation of chip multiprocessors or CMPs [1], [2]. To reduce programming complexity, a shared-memory programming model is chosen relying on a hardware coherence protocol to deal with block sharing in a safe way. Nevertheless, as core count increases, developing efficient coherence protocols constitutes a great challenge because the more block sharers the costlier coherence activity will be required. Besides, apart from high performance, an efficient coherence protocol should also deal with other important aspects, including resulting complexity and requirements in terms of on-chip area and energy consumption.

Directory-based, write-invalidate coherence protocols are recognized as the design of choice to provide high performance and scalability in future many-core CMPs [3]. To exemplify the difficult decision-making process to address the previous aspects at once, we choose Hammer [4] and Directory [5] protocols. In ensuring coherence, Hammer relies on sending as many coherence messages as the number of all private caches in the system, meanwhile Directory keeps track of exact coherence information about sharers (through a full bit vector) to send coherence messages just to those private caches with a valid copy of the memory block. Therefore, Directory is more efficient in terms of performance and energy consumption since it only injects the required coherence messages into the CMP’s interconnection network. Besides, Hammer minimizes on-chip area overhead, because it does not devote any extra hardware resources to encode a sharer list for every memory block.

Efficient cache coherency is also related to efficiency of the NoC employed. For instance, the MIT RAW processor’s NoC dissipates 40% of the overall chip power [6] and then, all coherence-related traffic may lead to significant energy consumption. Whereas most prior work examined NoC efficiency at architectural level (e.g., new topologies or improved network routers), other researchers have opted to allocate NoC resources based on characteristics of network traffic. For instance, Volos et al. [7] propose a heterogeneous NoC with two sub networks optimized to efficiently transmit request and response messages. Likewise, as we can see, we have analyzed network traffic to devise a dedicated NoC to improve transmissions of those coherence-related messages with higher impact on the critical path of cache misses (e.g., invalidations of all copies of a shared memory block).

In this work, we propose Express COherence NOtification (ECONO) protocol, a cache coherence protocol tailored to future many-core CMPs. ECONO is basically a directory-based protocol similar to Hammer, but it obtains performance results similar to Directory. Besides, our proposal outperforms both Hammer and Directory in terms of energy efficiency. ECONO ensures coherence by relying on atomic broadcasts that are transmitted over a lightweight dedicated on-chip network leveraging G-Lines technology [8] for maximum performance and scalability. Those atomic broadcasts stem from the most critical coherence-related traffic above explained. Projections for a 1024 core system reveals that our on-chip network entails more than 2× less on-chip storage than one of the most efficient and scalable directory-based coherence to date (SCD [3]), while keeping with reasonable power dissipation (5× power of a single read port in a 512-KB L2 cache bank).

The rest of the paper is organized as follows. Section II describes our proposal. A comprehensive evaluation of ECONO is carried out in Section III. Finally, Section IV shows our conclusions and future work.
II. ECONO Cache Coherence Protocol

A. Design

To develop a cost-effective coherence protocol, we take advantage of the minimal area overhead required by the Hammer protocol, that does not store the list of sharers for every memory block. This is the reason why our proposal makes use of broadcasting to accomplish the coherence actions. We decided to convey all broadcast messages through a dedicated on-chip network in order to avoid compromising QoS of applications. Notice that, neither Hammer nor Directory, nor the vast majority of protocol designs operate in this way. Moreover, due to coherence messages are commonly very short (i.e., the combination of the requested block’s address and type of coherence operation could be enough), ECONO’s network features a very lightweight and low-bandwidth infrastructure to minimize its impact on on-chip area as much as possible. Broadcast messages are transmitted in ECONO atomically. In consequence, we obtain simpler protocol specifications which is of paramount importance to reduce design complexity and protocol verification [9]. From now on, these atomic broadcast messages will be referred to as Atomic Coherence Notification (ACN) messages. Finally, for superior efficiency, we implement the ECONO’s network by using G-Lines technology [8] that, as we will see, leads to minimal on-chip energy consumption. This special network will be so-called the G-Line-based express coherence Network (GecNetwork).

B. Baseline Operation

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 II-D). 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 all the information required for the coherence operation: 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 (e.g., the home tile delivers its valid copy of the memory block to the requesting core), 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.Get_w) to the home tile (H). In this case, the proper coherence action would be the invalidation of 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. More precisely, in Hammer the home tile sends as many invalidation messages as the number of L1 caches in the system just to make sure that all L1 sharers (S) invalidate their copies. After an L1 cache (S or I) receives the latter message, it replies with an acknowledgement message towards the requester (R). In the meantime, the requester collects all the acknowledgements and the home’s memory block. Then, upon receiving all messages, the write cache miss is completed and the requester sends an unblock message to the home tile. Note that, like Hammer and Directory, ECONO uses unblock messages in order to handle races in a simple way [10].

Our coherence protocol improves the latter process as illustrated in Fig. 1a. First, instead of sending as many invalidation messages as L1 caches in the system, ECONO makes use of a unique ACN message (see 2.b INV) which is broadcast through a special on-chip network (i.e., 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 with a predictable and constant propagation latency. As a result, after a certain amount of time, the home tile precisely knows that all L1 caches have received the ACN message, thereby its copy of the block and write permission can be delivered to the requester. In the figure, we represent the atomic transmission with three different steps. First, before starting the coherence action, the home tile must acquire the use of the GecNetwork in mutual exclusion (see 2.a) because, as discussed later, this network is shared among all home tiles in the system and is conceptually like a bus. Second, the ACN message is broadcast to invalidate all L1 caches (2.b). And
third, after a certain amount of time (the time required to reach all L1 caches), the home knows that all cached blocks have been removed and it can release the GeNetwork (2.c). The rest of the process would be the same as commented above (3.Data and 4.Unblock).

In Fig. 1b, we show the actions performed in 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). Next, the home would send the block to the requester (4.Data) and so on. It is worth noting that the home tile can distinguish between whether it must expect a Data message from the owner or can send its copy of the memory block (e.g., Fig. 1a), depending on the L2 cache state stored for the particular memory block, as it is done in the Hammer protocol.

C. Extensions to the Baseline ECONO Protocol

As explained above, the baseline ECONO implementation makes use of a request-response operation mode that involves up to four hops in the critical path of a cache miss (see Fig. 1b). From now on, this implementation will be called 4-hop ECONO. Hammer and Directory optimize this situation by reducing the number of hops to three using forward messages transmitted from the home tiles to the owner caches. Fig. 1c) illustrates this optimization when considering our ECONO protocol. As we can see, through a forward ACN message (2.b FWD), the home tile would delegate the coherence action to the particular block owner (see M) which will ultimately send its copy of the block to the requesting core (3.Data), thereby saving one hop in the critical path of the cache miss. Hence, a shorter critical path involves a slightly higher protocol complexity and a larger ACN message that must include the requester’s ID needed by the block owner. This implementation will be referred to as 3-hop ECONO.

Figure 2a illustrates the format of an ACN message. As we can observe, it is made up of two different fields: the head, that is used to identify the type of action to be applied (e.g., invalidation or forward); and the payload, that stores the information required to identify the cached blocks that will be affected by the coherence action (e.g., the block address). Modifications to the ACN format define another group of extensions for the two implementations considered for ECONO. On the one hand, regarding the baseline 4-hop ECONO, according to Section II-B, the head field needs to cover two coherence actions: invalidation and downgrade. In particular, we have also considered two subtypes for the invalidation case depending on whether the action involves a single owner (M in Fig. 1b), or multiple sharers (S in Fig. 1a). Note that, like in Directory, the home tile readily would identify both cases by means of different states at the L2 cache. Therefore, to cover all possible cases, this field contains 2 bits. Furthermore, the payload field contains the block address of the ACN operation. This configuration is called BlockAddress and is outlined at the top of Figure 2b. While this is the most precise way to perform a coherence action, since the block address unequivocally identifies every copy in the L1 caches, we could consider a smaller number of bits in order to reduce propagation latency of the ACN messages as we will show in Section II-D. For example, every ACN message could contain a subset of the entire block address that would identify the cache entry where the block would be (i.e., the index). Nonetheless, the problem with this configuration, called Index and shown in the center of Figure 2b, would be when the L1 caches do not follow a direct-mapped scheme. For instance, in a N-way set-associative scheme, up to N cached blocks could be invalidated which may entail extra L1 cache misses that would degrade performance. To alleviate such an issue, we propose two different optimizations.

The first optimization will be referred to as Index+ExtraBits and is depicted at the bottom of Figure 2b. Here, apart from including the index field, we could incorporate some extra bits taken from the block address to check against the same bits stored in the tag address of every L1 cache entry. So, the more extra bits included the lower probability of invalidating wrong cached blocks. As a second optimization, we propose to apply two types of filters at the L1 caches aimed at reducing the number of cached blocks affected by the imprecision that the ACN messages entail. The state filter, that prevents from invalidating cached blocks with irrelevant stable states: in downgrade or invalidation actions on owners’ caches, stable states different to M or E; and for invalidations of multiple sharers, stable states other than S. Moreover, we propose the sharing filter, that would exclude all private blocks in the particular L1 cache entry from the coherence operations, due to coherence is only maintained for shared blocks. To this end, we assume an OS-based classification similar to [11].

On the other hand, as to the 3-hop ECONO, the head field needs to cover three coherence actions. First, invalidation, which is used to invalidate all shared copies in the system

1As explained in Section II-C, Fig. 1c illustrates an optimized version of the second scenario.
2This will be necessary to support the state filter explained below.
(e.g., for a write cache miss in Fig. 1a). Second, forward-downgrade, which is used to downgrade the owner’s copy and forward the data to the requesting core (for a read cache miss in Fig. 1b). And third, forward-invalidate, where the owner forwards and invalidates its copy (for a write cache miss in Fig. 1b). Consequently, in the 3-hop ECONO, the head also requires 2 bits. Finally, for the payload field, we will only consider an ACN message in which the block address is used along with the requester’s ID to identify the destination of the forward message. This configuration will be referred to as ID+BlockAddress and is shown in Figure 2c. The number of bits required by the ID will depend on \( \log_2(\text{Tiles}) \) in the system (e.g., 4 bits for the 16-tile configuration assumed in this work). Besides, in this ECONO implementation, we will not make use of shorter and imprecise ACN messages. Imprecise forwards would be very difficult to manage because there may be forwards to wrong owners, forwards to the correct owner on wrong blocks, or even data responses to wrong requesters from correct or wrong owners.

D. Physical Implementation

To develop our proposal, we choose a conventional tiled CMP architecture composed of a number of replicated tiles interconnected by means of a tightly integrated point-to-point 2D-mesh network (see Fig. 3 for a 3×3-tile layout). Each tile is comprised of a CPU core, two levels of an inclusive hierarchy of caches (private L1s and a logically-shared physically-distributed last-level L2 cache or L2 bank) with their respective L1 and L2 controllers, and a network interface or router that connects all tiles to the main network. Moreover, we use the less significant bits of the block address to determine the home tile for every memory block.

The architecture of ECONO is made up of two dedicated on-chip networks: the GecNetwork, to transmit the ACN messages; and the GLock, to guarantee atomicity for the transmissions. Both networks are implemented by assuming the G-Lines technology for superior efficiency. This technology was conceived to mitigate the well-known performance limitations in terms of latency, area overhead and power consumption of conventional RC-wires as feature sizes shrink [12]. 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.

1) GecNetwork: To broadcast the ACN messages, this dedicated on-chip network outlined in Fig. 4a is made up of horizontal and vertical G-Lines, and two types of controllers: 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. In this way, Tx controllers are co-located with the L2 controllers, whereas Ry controllers are co-located with the L1 controllers (see Fig. 3). The transmission of the ACN messages follows an efficient pipelined three-phase scheme. First, a particular Tx controller writes a message into its horizontal line. Then, the horizontal line in which the Tx is attached to broadcast the message to all the Ry controllers attached to the same line. Second, a vertical line is devoted to broadcast this message to the remaining horizontal lines. And third, these horizontal lines broadcast the message to the rest of Ry controllers. To shorten propagation latency of the ACN messages, we could employ lines of \( n \)-G-Lines to transmit \( n \) bits in parallel in every of the three phases.

2) GLock: To perform the atomic transmissions of the ACN messages, there may be multiple home tiles competing for the GecNetwork ownership. Therefore, it would be necessary to use a fair and efficient mechanism to resolve these possible scenarios. Due to its similarity to the problem of efficiently managing highly-contended locks in parallel applications, we adapt the GLock proposal in [13] to manage contention at L2-bank level (i.e., at every home tile). As outlined in Fig. 4b, our GLock comprises a set of manager and local controllers (called C for simplicity) which are interconnected by means of horizontal and vertical G-Lines. In short, the local controllers are in charge of requesting, receiving and releasing the GecNetwork ownership, meanwhile manager controllers arbitrate for fairly granting the ownership. Since the GLock is utilized by home banks, the C controllers are co-located with the L2 controllers (see Fig. 3).

III. Evaluation

A. Methodology

As testbed, we use full-system simulation by means of Virtutech Simics [14] (running Solaris 10) extended with Wisconsin GEMS toolset [15]. The latter provides detailed simulation of multiprocessor systems along with precise modeling of timing and energy consumption for the interconnection network, through GARNET [16] and ORION 2.0 [17] components, respectively. Moreover, energy consumed by the cores in the simulated 16-core CMP has been modeled with McPAT [18]. Table I summarizes the values of the main configurable parameters assumed in this work.
As benchmarks, we use nine multi-threaded applications: five from the SPLASH-2 benchmark suite [19], one from the PARSEC benchmark suite [20], and three additional scientific applications. Table II shows them and their respective problem sizes. As we will discuss, these applications were chosen because they exhibit different communication patterns ranging from small coherence activity like in Barnes to high activity like in Swaptions. All experimental results reported in this work are for the parallel phase of the benchmarks under study.

To quantify the performance benefits derived from our proposal, we compare the best-performing configurations for 4-hop and 3-hop ECONO against the two contemporary coherence protocols considered in this work: Hammer and Directory. To determine the optimal settings, we explored the efficiency of all the extensions discussed in Section II-C. From this study, omitted in this work for the sake of brevity, we conclude the following:

1) The GecNetwork must be comprised of three G-Lines resulting in a 3-bit width network.
2) Two GecNetworks for ECONO is enough to provide marginal contention.
3) The 4-hop ECONO with a Index+5 payload presents the best choice because this imprecise configuration has marginal extra L1 cache misses, while taking benefit from shorter propagation latencies (i.e., ACN messages with less bits to transfer).
4) The 3-hop ECONO protocol with the ID+BlockAddress payload constitutes the best option because imprecise ACN messages lead to high complexity in the protocol as we will explain.
5) We must implement both the state and sharing filters (further details in Section II-C).

To ensure a fair comparison in terms of performance, all protocols share a common specification that consists of write-invalidate policy, MESI state machines for the first-level caches, and inclusive cache hierarchies with write-back caches and an invalidation-on-eviction policy for L2 replacements. In consequence, the Hammer implementation evaluated in this work is an optimized version of [4] because of having precise knowledge about the memory blocks that are cached. Note that, otherwise, broadcasts would always be sent to both on-chip caches and memory in order to obtain the requested data. This would generate extra traffic into the interconnect when data is off-chip, and important performance degradation waiting for off-chip responses when no needed. Moreover, Directory has been implemented by using upgrade requests. This entails an optimization in terms of network traffic because an upgrade is used to request write permission for a cached block with only read permission. If the requester still remains as a sharer at home’s coherence information, an acknowledgement message from the latter would be sent instead of the home’s copy of the block, thereby saving network traffic (i.e., 8 vs. 72 bytes according to Table I). Note that, the rest of protocols evaluated in this work do not store the list of sharers so that this optimization cannot be applied.

B. Performance Results

Before starting with the evaluation part, we depict in Fig. 5 a characterization of the different coherence actions performed by the Directory protocol in order to understand where the improvements come from. We choose Directory because this protocol distinguishes a more diverse set of coherence actions to perform, what provides more information for the performance comparison.

As we can observe in the figure, each bar is broken down into four categories depending on the percentage of coherence actions devoted to: forward messages, that comprehend all coherence messages transmitted from the home tiles to the owners in order to recover the single valid copy of the requested block (Fwd in the figure); upgrade actions, that result from granting write permission to a requesting core that has a read-only copy of a cached block (i.e., it implies invalidation of sharers and an acknowledgement sent to the requester, or Inv&Ack); coherence activity devoted to grant write permission to a requester that does not have a block that is shared at the home tile (i.e., it involves invalidation of sharers and delivery of the home’s block copy to the requester, or Inv&Data); and finally, those actions that only require

<table>
<thead>
<tr>
<th>TABLE I: CMP baseline configuration.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiles</td>
<td>16</td>
</tr>
<tr>
<td>CPU Core</td>
<td>3 GHz, inorder, single-issue</td>
</tr>
<tr>
<td>Cache Hierarchy</td>
<td>Cache line size: 64 Bytes</td>
</tr>
<tr>
<td></td>
<td>L1 I/D-Cache: 32 KB, 4-way, 2 cycles</td>
</tr>
<tr>
<td></td>
<td>L2 Cache Bank: 512 KB, 8-way, 12×4 cycles</td>
</tr>
<tr>
<td>Memory access time</td>
<td>250 cycles</td>
</tr>
<tr>
<td>Network configuration</td>
<td>Topology: 2D-mesh</td>
</tr>
<tr>
<td></td>
<td>Routing: X-Y Dimension Ordered</td>
</tr>
<tr>
<td></td>
<td>Packet size: 72 Bytes (Data); 8 Bytes (Control)</td>
</tr>
<tr>
<td></td>
<td>Bandwidth: 48 GB/s</td>
</tr>
<tr>
<td></td>
<td>Flit size: 16 Bytes</td>
</tr>
<tr>
<td></td>
<td>Link bandwidth: 1 flit/cycle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II: Benchmarks and input sizes.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmarks</td>
<td>Input Size</td>
</tr>
<tr>
<td>Barnes</td>
<td>8192 bodies, 4 steps</td>
</tr>
<tr>
<td>FFT</td>
<td>64K complex doubles</td>
</tr>
<tr>
<td>Ocean</td>
<td>258 x 258 ocean</td>
</tr>
<tr>
<td>Radix</td>
<td>1048576 radix</td>
</tr>
<tr>
<td>Raytrace-opt</td>
<td>Teapot</td>
</tr>
<tr>
<td>EMD3</td>
<td>38400, degree 2, 15%, 50 steps</td>
</tr>
<tr>
<td>Tomcatv</td>
<td>256 points, 5 time steps</td>
</tr>
<tr>
<td>Unstructured</td>
<td>Mesh.2K, 5 time steps</td>
</tr>
<tr>
<td>Swaptions</td>
<td>simmedium</td>
</tr>
</tbody>
</table>
sending the requested data to the requesting core (i.e., Data).
Note that, we do not expect to find significant performance
differences among the four protocols for benchmarks that
are dominated by coherence actions belonging to the Data
category (i.e., all but Raytrace, Swaptions and Unstructured),
because in such benchmarks all the protocols would behave in
the same way.

1) Execution Time: Fig. 6 shows the normalized execution
times with respect to those obtained when Hammer is con-
sidered. 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). More specifically, the magnitude of these
savings depends on the number of coherence operations opti-
mized by Directory (i.e., those that fall into Fwd, Inv&Ack
and Inv&Data categories in Fig. 5). This is the reason why
negligible performance improvements are obtained in all but
Raytrace, Swaptions and Unstructured benchmarks. Besides,
the highest improvements are achieved in Swaptions because in
this benchmark roughly 50% of the coherence actions involve
a single point-to-point forward message (see Fwd category
in Fig. 5). Conversely, in Hammer, a number of messages
equal to the number of L1 caches in the system is required.
This increases execution time mainly because of the higher
contention at the main interconnect, and the more time spent
at requesting cores waiting for all the acknowledgements.

Regarding our ECONO implementations, by relying on a
single ACN message transmitted in broadcast over a dedicated
and very fast G-Line-based on-chip network, both implemen-
tations outperform the Hammer protocol. Nevertheless, the
4-hop ECONO requires one more hop when data recovery is
necessary as we discussed in Figure 1b, and then, it cannot
outperform Directory, that manages this situation much more
efficiently than Hammer does as follows. 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. In particular, Raytrace and
Swaptions report the higher performance gap between the
4-hop ECONO and Directory because, as shown in Fig. 5,
these benchmarks have a fraction of approximately 30% and
50% devoted to forward messages, respectively (see the Fwd
category). Finally, when analyzing the execution times reported
by 3-hop ECONO, using less number of hops leads to slightly
better results than Directory (2% on average). Note that, this
improvement is expected to be higher for larger many-core
CMPs because ECONO always transmits a single ACN mes-
sage through its extremely fast GecNetwork, whereas Directory
would require more and more coherence messages transmitted
through the ever-larger packet-switched CMP’s interconnect.

2) Network Traffic: Fig. 7 shows the total network traffic
depending on the implementations discussed above, normal-
ized with respect to Hammer. In particular, each bar plots the
number of bytes transmitted through the interconnection net-
work (the total number of bytes transmitted by all the switches
of the interconnect). As expected, Directory outperforms Ham-
mer (27% on average) because of using precise information
of cached blocks that removes all unnecessary coherence
messages from the interconnect. According to our character-
ization in Fig. 5, these improvements come as consequence
of the aggregate fraction of Fwd, Inv&Ack and Inv&Data
categories, since for such coherence actions Directory always
performs better than Hammer: Fwd and Inv&Data would
require sending as many point-to-point transmission as the
number of L1 caches in the system; and Inv&Ack would be
implemented as Inv&Data since an efficient implementation
of upgrade operations requires keeping track of the sharer list
for every memory block, so that a home tile always knows if
the requester still remains as a sharer in order to successfully
reply with the Ack. In this way, neither Hammer nor ECONO
implement upgrade messages.

Regarding the ECONO implementations, recall that our
protocol removes an important amount of coherence-related
traffic from the main CMP’s network (i.e., the home tiles
transmit ACN messages over the GecNetwork rather than
invalidation or forward messages over the main interconnect,
and no acknowledgements are necessary), so that it must lead
to significant reductions in network traffic. Considering the
4-hop ECONO implementation, we can observe that it does
not outperform Directory (degradation of 3% on average).
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. Note that, according to
the simulation parameters shown in Table I, control messages
are one-9th of the size of data messages, and then injecting
more data messages to reach the requester’s cache nullifies the
benefits of removing all coherence requests and acknowledge-
ments for almost all the benchmarks.
TABLE III: Projection of peak power dissipated by ECONO protocol.

<table>
<thead>
<tr>
<th>Cores</th>
<th>Transmitters</th>
<th>Receivers</th>
<th>Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>350</td>
<td>1,082</td>
<td>0.642</td>
</tr>
<tr>
<td>256</td>
<td>696</td>
<td>2,160</td>
<td>1.281</td>
</tr>
<tr>
<td>512</td>
<td>1,386</td>
<td>4,326</td>
<td>696</td>
</tr>
<tr>
<td>1024</td>
<td>2,762</td>
<td>8,654</td>
<td>4,326</td>
</tr>
</tbody>
</table>

Fig. 8: Normalized Energy Consumption w.r.t. Hammer.

As to the 3-hop ECONO protocol important reductions are reported, due to the fact that forwarding operations are enabled. As we can observe in Fig. 7, this implementation saves network traffic by 3% compared to Directory. In particular, applications such as Swap or Raytrace that present frequent invalidation of sharers and forward operations (see Inv&Ack, Inv&Data, and Fwd categories in Fig. 5), report the highest improvements. Moreover, Unstructured reports similar results to Directory because, in this benchmark almost 40% of the coherence activity observed in Directory is optimized by using upgrade petitions that replace data responses with the shorter acknowledgement messages (see Inv&Ack category in Fig. 5).

3) Energy Consumption: We quantify the peak power dissipated by the G-Line-based ECONO architecture. To this end, we employ the same power dissipation parameters for a 65-nm CMOS process described in [8]: 0.6 mW per T controller; and 0.4 mW per R controller. Moreover, according to [8] no static power is dissipated by the G-Line circuitry. We obtained that the total peak power dissipated by ECONO using two GecNetworks is equal to 87.6 mW. More precisely, 31.8 mW for a single 3-G-Line GecNetwork and 12 mW for its GLine. Utilizing CACTI [21], the magnitude of 87.6 mW is approximately one-4th of the power dissipated per read port in the L1 caches simulated (see Table I). Moreover, Table III illustrates how the 3-G-Lines 2-GecNetworks ECONO would scale as the core count grows from 128 to 1024 cores. Since every G-Line can support up to seven controllers as much [8] (i.e., 7×7-core CMPs using a 2D-mesh topology), for larger CMPs we connect groups of 7×7 cores in a hierarchical scheme. The total amount of transmitters and receivers is also shown in every case with the total power dissipated by them. As we can observe, for a 1024-core system, 5,118 W are dissipated. By using CACTI [21], we obtain that this magnitude is roughly 5× the power dissipated by a read port in an L2 cache (see Table I) that is not very much taking into account this large number of cores.

We conduct an energy comparison for the four protocols and the 16-core CMP system. For that, both ORION 2.0 and McPAT tools have been configured to provide energy results for a 65-nm CMOS process to be consistent with the G-Line technology. Fig. 8 illustrates the normalized energy consumption normalized with respect to Hammer protocol considering both static and dynamic energy. Each bar is broken down into three categories depending on the percentage of energy devoted to: our ECONO infrastructure, taking into account the power dissipation results exposed above; the main interconnection network (Network); and all tiles in the simulated system (Tiles). As we can observe, reductions in the energy consumed are directly related to the reductions in execution time and network traffic already exposed for Directory and both ECONO implementations (see Tiles and Network categories respectively). For instance, the Network category is considerably reduced in case of Raytrace, Swaps and Unstructured due to the important reductions in network traffic reported in Fig. 7. Likewise, Tiles category is also improved due to the reductions in execution time plotted in Fig. 6. Moreover, due to the negligible extra power dissipation required by the ECONO infrastructure for a 16-core CMP, the ECONO category represents a marginal fraction of the energy consumed for the full system. Consequently, 4-hop ECONO is more energy efficient than Hammer (18% on average), and that 3-hop ECONO is the most energy efficient design (3% better than Directory on average).

4) On-Chip Area: In this section, we compare on-chip area overhead required by ECONO against Directory. For that, we analyze different types of encoding representations ranging from the full-map employed in Directory to those optimized representations of Hierarchical [22] and SCD [3]. The former allows an exact and area-efficient representation of sharer sets where a hierarchy of two levels of directories are used: each first-level directory encodes the sharers of a subset of caches, and the second level tracks directories of the previous one. SCD constitutes one of the most efficient directories to date. It encodes exact sharer sets by using variable-size representations: lines with one or few sharers use a single directory tag, while widely shared lines use additional tags. To increase performance it also leverages novel highly-associative caches as in Cuckoo Directory [23].

Table IV shows the storage overhead depending on the three directory organizations considered (Full-map Directory, Hierarchical and SCD) from 128 to 1024 cores. Area overhead is given as a percentage of the total area required by aggregating all L2 caches in every case (we use the L2 cache size shown in Table I). We base the percentage on the study carried out in [3]. Last column accounts for our ECONO protocol. In this case, the percentages stem from the ECONO on-chip area overhead considering the 3-G-Lines 2-GecNetworks configuration. To obtain such values, we follow the specifications given in [8]: 7-mm G-Lines and a differential pair width of 5.6 µm. This leads to 0.0392 mm² per G-Line including its controllers, actually 0.0292 mm² if we share the ground shield every two pairs. The table illustrates that our ECONO infrastructure considerably reduces the on-chip area overhead of a Full-map Directory (10×-27×), and save more than 2× area with respect to the most efficient scalable SCD protocol. From this study, we can also affirm that ECONO is the most scalable design in terms of on-chip area overhead.
TABLE IV: Storage requirements for different Directory protocols against the 3-G-Lines 2-GecNetworks ECONO protocol.

<table>
<thead>
<tr>
<th>Cores</th>
<th>Full-map</th>
<th>Hierarchical</th>
<th>SCD</th>
<th>ECONO</th>
<th>Full-map vs ECONO</th>
<th>SCD vs ECONO</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>34.18%</td>
<td>23.09%</td>
<td>10.94%</td>
<td>2.44%</td>
<td>10.25×</td>
<td>4.48×</td>
</tr>
<tr>
<td>256</td>
<td>59.18%</td>
<td>24.22%</td>
<td>12.50%</td>
<td>2.91%</td>
<td>17.16×</td>
<td>4.29×</td>
</tr>
<tr>
<td>512</td>
<td>109.18%</td>
<td>26.95%</td>
<td>13.87%</td>
<td>5.23%</td>
<td>19.11×</td>
<td>2.65×</td>
</tr>
<tr>
<td>1024</td>
<td>209.18%</td>
<td>30.86%</td>
<td>15.82%</td>
<td>7.22%</td>
<td>27.71×</td>
<td>2.19×</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS AND FUTURE WORK

In this paper we propose ECONO, an efficient and scalable 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. We study a baseline implementation for our coherence protocol and propose two different extensions. First, 4-hop ECONO with imprecise coherence information to shorten the size of ACN messages, hence operation latency. And second, we also consider a 3-hop optimization to reduce the number of hops and save network traffic. Detailed execution-driven simulations of a 16-tile CMP suggest that the best option to implement ECONO consists of the 3-G-Lines 2-GecNetworks fabric. Moreover, when considering imprecise coherence information, Index+5, that comprises both the index field and five extra bits taken from the block address, is the preferred choice. To quantify the performance benefits of our proposal, we compare the best-performing configurations for 4-hop and 3-hop ECONO against two contemporary coherence protocols, Hammer and Directory. Our study reveals that our proposed protocol improves considerably Hammer in terms of execution time and network traffic, outperforms significantly Directory in on-chip area, and constitutes the most energy efficient design. Projections for a 1024-core system reveals its promising scalability since ECONO requires 2× less space overhead in comparison to the most efficient SCD protocol to date, while keeping with reasonable power dissipation.

As future work, we plan to minimize impact on energy consumption for those ACN messages transmitted when there is a single block owner or a reduced number of block’s sharers. For that, we would incorporate broadcast filtering at GecNetwork level to reach only the required L1 caches. Moreover, we will investigate the possibility to include the identity of the next block owner into the ACN messages. In this way, every atomic broadcast would notify all L1 caches regarding the block owners for their cached blocks, thereby a subsequent L1 cache miss could be resolved directly by the particular cache owner avoiding instruction to its home tile.

ACKNOWLEDGMENT

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