A NoC-level Support for Broadcast-based Coherence Protocols

Mario Lodde[1] Jose Flich[2] Manuel E. Acacio[3]

Abstract— **Chip Multiprocessor Systems (CMPs) rely on a cache coherency protocol to maintain memory access coherence between cached data and main memory. The Hammer coherency protocol is appealing as it eliminates most of the space overhead when compared to a directory protocol. However, it generates much more traffic, thus stressing the NoC and having worse performance in terms of power consumption. When using a NoC with built-in broadcast support network utilization is lowered but does not solve completely the problem as acknowledgment messages are still sent from each core to the memory access requestor. In this paper we propose a small and simple control network that collects the acknowledgement messages and delivers them with a bounded and fixed latency, thus relieving the NoC from a large amount of messages. Experimental results demonstrate on a 16-tile system with the control network that execution time improves up to 17%, with an average improvement of about 7.5%.**

Keywords— **CMPs, cache hierarchy, NoC**

I. Introduction and Motivation

Many-core chip multiprocessors (CMPs) integrate tens or even hundreds of processor cores onto the same die [1], [2]. In such systems, all the processor cores and other components (memory components, accelerators, memory controllers) are interconnected using a high-speed on-chip network (NoC) [3]. If current trends continue, future CMPs will implement the hardware-managed, implicitly-addressed, coherent caches memory model [4]. With this memory model, all on-chip storage is used for private and shared caches that are kept coherent in hardware using a cache coherence protocol. Communication between threads is achieved through shared memory. On the other hand, future many-core CMPs will probably be designed as arrays of identical or close-to-identical building blocks (tiles) connected over an on-chip switched direct network [2]. These design has been claimed to provide a scalable solution for managing the design complexity, effectively using the resources available in advanced VLSI technologies. Figure 1 illustrates the tile-based CMP that we assume for developing our proposal. More specifically, we show the case of a CMP system with 16 tiles, in which every tile has a core, private L1 caches and a slice of a shared but distributed L2 cache.

Directory-based cache coherence protocols have been typically employed in systems with switched direct networks, as tiled CMPs are. The directory structure is distributed between the L2 cache banks. In this way, each tile keeps the sharing information of

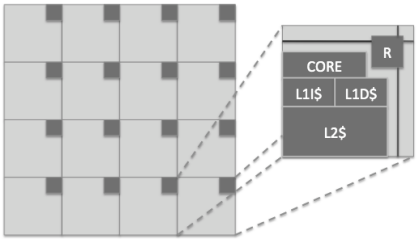

Fig. 1. Tile-based CMP system.

the blocks mapped to its L2 bank. This information includes the state of each cache line and its sharing code, that holds the list of current sharers. Most of the bits of each directory entry are devoted to codify the sharing code, and therefore the total size of the directory structure is mainly determined by it. This extra storage adds requirements of area and energy consumption to the final design and could restrict the scalability of future many-core CMPs

Among the organizations for the sharing code appeared a long time [5], two deserve special attention. On the one hand, a bit-vector (one bit per private cache) could be used to have an exact representation of the sharers in each moment. On the other hand, Dir_0B does not dedicate any bits to the sharing code, requiring a broadcast on all coherence operations and giving rise to broadcast-based directory protocols. The Hammer protocol employed in systems built using AMD Opteron processors [6] is the most representative example of the latter [1].

Figure 2 and 3 show the basic NoC-level actions performed in both protocol implementations when dealing with a L1 write miss. In the broadcast-based protocol, the corresponding L2 bank is notified which triggers a broadcast operation. To complete the operation, all the nodes need to notify the requestor with an ACK message (after invalidating their copy). In a bit-vector directory protocol, however, only the node who owns the block is notified (or the sharers, if they exist).

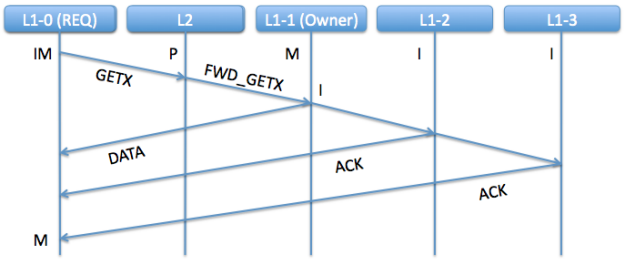

Fig. 2. L1 cache write miss with hammer protocol.

Summarizing, bit-vector directory protocols do not scale since the size of the directory grows linearly with the number of cores. However, these protocols

[1]Grupo de Arquitecturas Paralelas, Universitat Politècnica de València , e-mail: mlodde@gap.upv.es
[2]Grupo de Arquitecturas Paralelas, Universitat Politècnica de València , e-mail: jflich@disca.upv.es
[3]Universidad de Murcia, e-mail: meacacio@ditec.um.es

[1]We use the terms "broadcast-based directory protocol" and "Hammer protocol" interchangeably along this work.

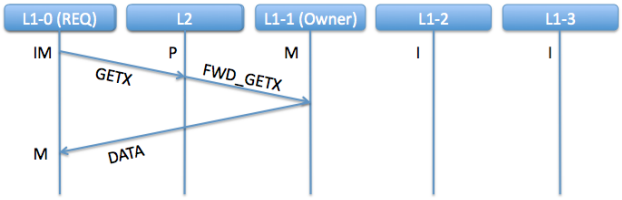

Fig. 3. L1 cache write miss with directory protocol.

generate the minimum amount of network traffic. On the other hand, broadcast-based protocols completely remove the sharing code resulting in a completely scalable directory structure, but the number of messages on coherence events increases linearly with the number of cores, which limits its applicability to systems with a small number of cores.

In an isolated design environment, where the NoC and the coherence protocol are designed separately, many optimization opportunities are lost. If, however, the NoC is co-designed with upper layers, the overall system can be optimized. In particular, the NoC can be designed to efficiently deal with the operations of a broadcast-based directory protocol. This is the approach followed in this work.

Figure 4 shows a classification of the messages that travel on the NoC of our simulated system when running nine different SPLASH-2 applications under the Hammer coherency protocol (simulation details are provided in the evaluation section). Network messages are divided in four categories: *Requests* includes all the requests sent by L1 to L2 caches; *Responses* includes all data messages sent to the requestor and writeback acknowledgements sent by the L2 back to the L1s; *Coherence reqs* includes the coherence requests (invalidations and cache-to-cache transfer commands) sent from the home L2 bank to all the L1s; and *Coherence res* includes all the acknowledgements from the L1 caches back to the L1 requestor. As it can be derived from the figure, 60% on average of the total traffic is due to coherence requests and their associated acknowledgements. It is also important to note that this amount reaches almost 90% for some applications (Barnes, FFT and Water-nsquared).

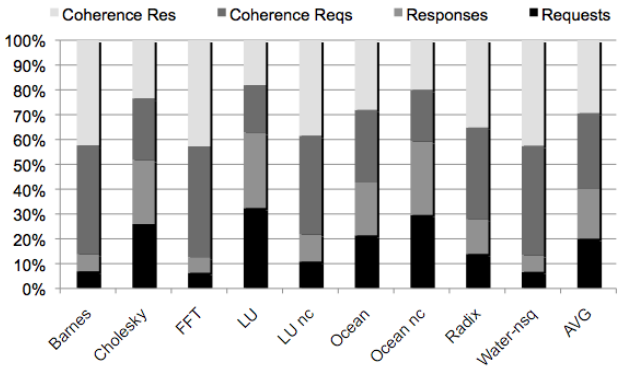

Fig. 4. Breakdown of NoC messages when using the Hammer protocol for different SPLASH-2 applications.

One well-known strategy to reduce the traffic generated by coherence requests is to use a NoC with multicast or broadcast support. With this kind of support, on every coherence event the home directory would inject a single multicast message instead of multiple serialized messages (one per destination). Obviously, this would save a very significant fraction of the coherence requests. However, broadcast-based directory protocols cannot take full advantage of a NoC with multicast or broadcast support due to the fact that the acknowledgment messages (*Coherence res*) are still there and would penalize performance in a noticeable way (note from Figure 4 that these messages represent more than 40% of the total traffic in several cases). In this way, dealing with these acknowledgement messages in the Hammer protocol is still an open issue. In this work we tackle such challenge and take an heterogeneous design approach of the NoC in order to speedup the Hammer protocol. In particular, we design a fast and simple gather control network (GCN) associated with the NoC that collects all the acknowledgement messages of the L1 caches. The main NoC is still used to transport requests, responses (including messages with data) and coherence requests. However, all the acknowledgement messaging associated with coherence requests is separated from the NoC and sent through the gather network. With this design, the hammer protocol performance is boosted and pairs and even improves the performance of a typical directory-based protocol [2]. In particular, execution time of applications is reduced on average by 7.5% whereas messages in the network are reduced by 60%.

The rest of the paper is organized as follows. In Section II we describe the proposed NoC with the gather control network in place. Then, in Section III we provide performance and NoC results for different coherence protocols. Then, Section IV describes the related work, and finally, Section V concludes the paper.

II. Heterogenenous NoC design with Gather Control Network

A. Overall NoC Design

The complete NoC switch design is shown in Figure 5. The network is made of 4-stage pipelined switches with the typical stages: IB (input buffer), RT (routing), VA/SA (virtual channel and switch allocation) and XT (crossbar transversal). Each switch is connected to its neighbors through each dimension and direction and to the local nodes. Up to three nodes are located on each tile (the L1 cache, the L2 bank, and possibly a memory controller).

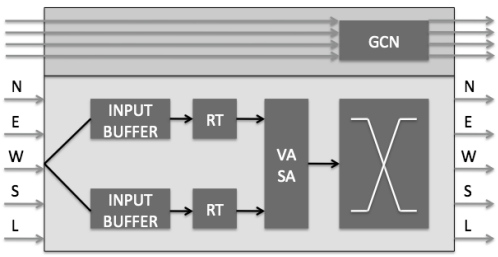

Fig. 5. Switch design.

[2]In [7] we have also implemented a similar approach for directory-based protocols.

Each input port has five virtual channels. Each virtual channel has its own input buffer, which can store up to four flits. A Stop&Go flow control regulates the advance of flits between adjacent switches.

B. Gather Control Network

An additional control logic and wiring is added to to this typical NoC switch desig, building the gather control network (GCN). A GCN consists of 16 single-wire subnetworks (one per node) made of AND gates, where 15 nodes are located at the leaves of the tree and the other node is at the root of the tree. The GCN serves as an ACK of all the nodes.

Figure 6 shows the number of wires of the gather control signals between switches. Each switch handles both input and output control signals through all its ports. For a $N \times N$ mesh NoC, the number of outgoing control signals through all the output ports is $N - 1$, in our case 15. Each control signal is a different one-bit subnetwork addressed to a different destination ($N - 1$ destinations).

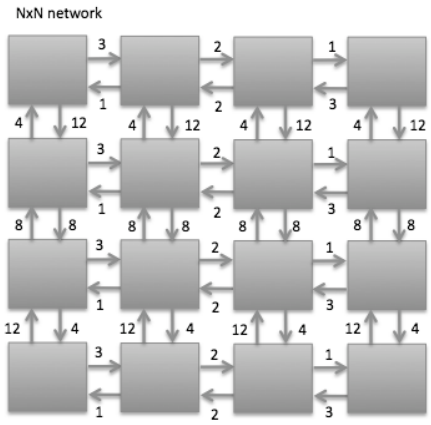

Fig. 6. Control signals distribution for the gather network. YX layout.

Figure 7 shows the layout for the subnetwork dedicated to node 1. ACK messages will be collected through Y columns and then collected through the first row. At each switch control signals are grouped with AND gates. In particular, at each switch the incoming control signals and the control signal from the local cores are ANDed. Since there is at least one core connected to each switch, an AND gate is needed at each switch for the control network addressed to node 1, except for the last row of switches.

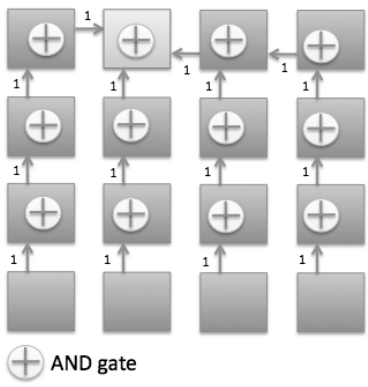

Fig. 7. YX layout for gather control network for end node 1.

Each GCN logic at each switch is connected to its neighbors control logic blocks with dedicated wires. Figure 8 shows the logic at each switch for the gather control network. The goal of the logic block is simply to AND the corresponding input signals and to distribute the results through the corresponding output ports, depending on the location of the switch in the mesh topology and the selected layout.

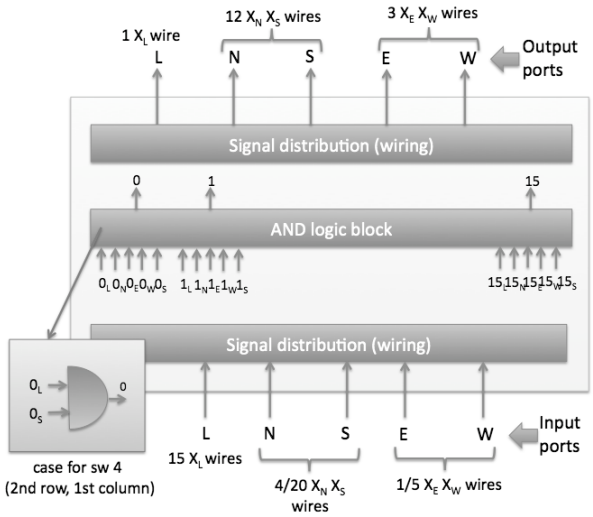

Fig. 8. General GCN block at switches.

The logic receives as input 15 control signals from the local core, each addressed to a different destination node. X_L means a control signal coming from the local port and addressed to destination X. Thus, we have from 0_L up to 15_L signals (excluding the one for the local core). In addition, we have up to 20 control signals coming from either north (N) or south (S) input ports and up to 5 control signals from either east (E) or west (W) input ports. These signals are distributed (based on the location of the switch in the mesh) and assigned to the corresponding inputs of the AND gate array. The outputs of the AND gates are then distributed over the output ports, again depending on the location of the switch. Notice that 15 output control signals are generated, one per destination in the system (except for the local node). Figure 8 shows the case for the AND gate generated for node 0 at switch 5 (2nd row and 1st column of the mesh). This output signal will be forwarded through the north (N) output port. The logic at each switch is minimum, consisting of 16 AND gates, most of which with just two entries. The signal distribution blocks are simply a rearrangement of the input and output control signals to the appropriate inputs and outputs of the AND gates.

To better balance the wires, we can use a mixed approach where wires for half the nodes are mapped YX and wires for the other half of nodes are mapped XY. The latency through the GCN does not change as the path follows the same manhattan distance. Figure 9 shows the case where nodes with the underlined ID number follow the YX mapping and the other follow XY mapping. This way we achieve a perfect distribution of wires, where each bidirectional port handles 10 wires for a 4×4 mesh network.

As the system size increases, the number of wires increases, but not the logic complexity at the

switches. For a $N \times N$ network, this mapping strategy requires $(N^2 + N)/2$ wires per direction and dimension.

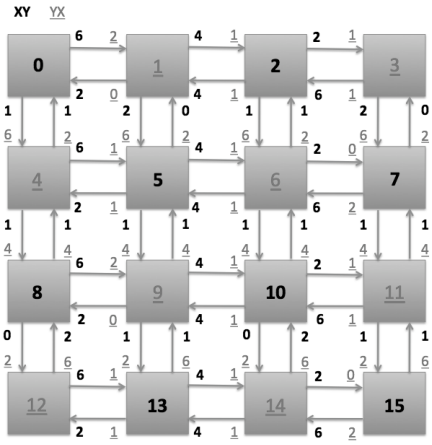

Fig. 9. Control wire distribution for mixed XY/YX mapping.

C. Switch Design: Support for the gather control network

In this section we provide an analysis of the overhead of the GCN. The basic switch described in Ssection II-A has been implemented using the 45nm technology open source Nangate [8] library with Synopsys DC. Cadence Encounter has been used to perform the Place&Route. Link width and flit size are set to 8 bytes. Both 4×4 and 8×8 2D meshes have been implemented. Two scenarios are analyzed, a conventional 2D mesh without introducing the gather network circuit, and the same 2D mesh when the gather network circuit is added.

Table I shows the critical path of a conventional 2D mesh network when two types of switches are used (with and without registers at output ports). Two link lengths have been analyzed: 1.2mm and 2.4mm. As it can be seen, a registered switch has a lower critical path than the non registered switch. By now on, the registered switch will be used to analyze the impact when introducing the GCN.

TABLE I
CONVENTIONAL 2D MESH CRITICAL PATH.

Critical path (ns)	N.R. Switch		R. Switch	
link length (mm)	1.2	2.4	1.2	2.4
2D mesh	1.86	2.17	1.35	1.75

Table II shows the critical path of the gather network circuit when analyzed in isolation, which is fixed by the dedicated logic that connects the two most physically separate nodes in a chip. The latency of this circuit depends on the mesh radix, since the gather network critical path crosses the whole network. Then, as the number of nodes increases and the link length does too, the critical path of the gather network circuit increases.

For a 4×4 network with a link length of 1.2mm, the gather network critical path is smaller than the critical path of a conventional network, and hence, the gather network is able to work at the same operating frequency than the conventional 2D mesh. That is, in that case, introducing the dedicated circuit does not affect the rest of the network. In contrast, if the link length is increased, the gather network has a higher critical path than the conventional 2D mesh. That implies two possibilities. First, the whole network increases its clock cycle to that fixed by the gather network circuit. Second, the 2D mesh operates at the same clock cycle while the gather network circuit requires 2 clock cycles to operate. When designing an 8×8 network, it can be seen that the gather network circuit does not scale as well as the point-to-point communication protocol of the NoC. However, it can be noticed that in the worst case the GCN is still able to cross the network in 4.32 ns (3 clock cycles). As we will see in the performance evaluation section, much higher delays in the gather network will not induce any penalty to performance.

TABLE II
GATHER CONTROL NETWORK CRITICAL PATH.

Critical path (ns)	4x4 Network		8x8 Network	
link length (mm)	1.2	2.4	1.2	2.4
GCN	1.23	2.20	2.65	4.32
2D mesh	1.35	1.75	1.35	1.75

Table III shows the area of a switch and the area occupied by the gather network circuitry in a single switch. In a non registered switch, the area overhead introduced by the gather network circuit is just an 1.54%. In a registered switch this overhead increases up to 2.81%.

TABLE III
SWITCH AND GATHER CONTROL NETWORK CIRCUIT AREA.

Area (mm^2)	N.R. Output	R. Output
Switch	0.132	0.165
GCN	$0.20 * 10^{-2}$	$0.45 * 10^{-2}$

III. EVALUATION

In this section we provide an evaluation of the resulting coherence protocol co-designed with the NoC. In particular, we compare the performance with different applications when using the Hammer protocol with neither broadcast nor gather support at NoC level (HAMMER), the Hammer protocol with NoC built-in broadcast support (HAMMER_BC), the Hammer protocol with both broadcast and gather control network support (HAMMER_BC_GCN), and a MOESI directory-based protocol (DIRECTORY).

We implemented the NoC and all the coherency protocols with an in-house memory/network simulator which is cycle-accurate and flit-level accurate. The memory/network module has been integrated into the Graphite simulator [9] which allows us to run applications and capture memory accesses.

We ran several applications of SPLASH-2 benchmark suite on the 16-core CMP system considered so far. Broadcast is achieved by following the XY routing. By default, the control gather network is modeled with 2 cycle delays. Each tile has a 128KB private L1 cache (64KB for instructions and 64KB for data, tag and cache latencies are set to 1 and 2

cycles) and a 512KB L2 bank (tag and cache latencies are set to 2 and 4 cycles); the total size of the L2 cache is thus 8MB.

Figure 10 shows the normalized execution time for different SPLASH-2 applications when using the different protocols and NoC implementations. As a first result, we can see how directory protocol achieves much lower execution time when compared to a Hammer protocol with no NoC support. In some applications, execution time can be reduced by 17%. On average, execution time is reduced by 6%.

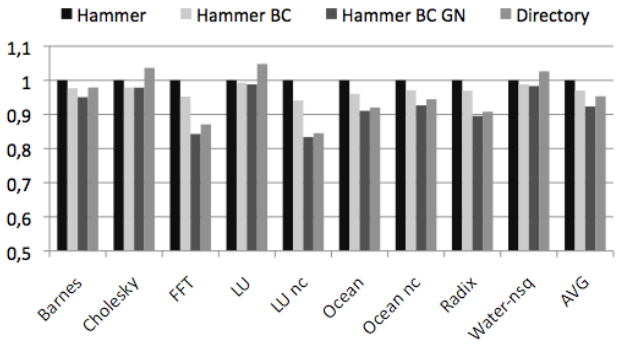

Fig. 10. Normalized execution time for different coherence protocols and different applications.

Now, as we provide NoC support for the Hammer protocol we can see how execution time reduces significantly. As a first measure, adding a broadcast facility reduces execution time by %3 on average (%7 in some applications). Furthermore, execution time is reduced again when the GCN is in place. On average, execution time is now 8% lower than a naive Hammer protocol. Indeed, execution time is even lower that when using directories. This is due to the speedup in the notification of the ACK messages that are needed also in directory-based protocols. This benefit is much higher than the overhead in broadcasting all the nodes in the system.

Figure 11 shows the normalized number of messages sent over the network for the different protocol-network support combinations. As can be seen, the directory-based approach achieves a significant reduction in messages overhead. Indeed, only 40% of messages are needed (when compared with HAMMER). Notice now how the different NoC additions (broadcast and GCN) help in reducing significantly the number of messages in the network. Indeed, when combined, the Hammer protocol has similar traffic overhead than DIRECTORY. Therefore, the Hammer protocol is able to cope with directory-based protocols without the overhead of managing directory structures, and with no significant messaging overhead over the network.

Figure 12 shows the performance achieved with HAMMER_BC_GCN when the GCN has different delays. We used a gather control network ranging from a 2-cycle delay up to a 128-cycle delay. Notice that we showed realistic delays of up to 4 cycles can be achieved even for 8×8 configurations. As we can observe, the results are quite insensitive to low and medium GCN delays. Indeed, performance is not significantly affected even for a 64-cycle GCN. Only, on average, execution time is increased by 1%.

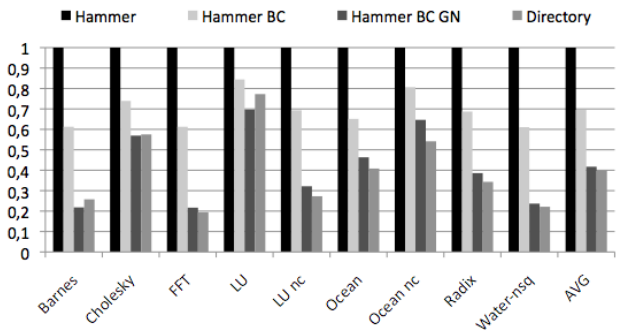

Fig. 11. Normalized number of injected messages for different coherence protocols and different applications. GCN signals are not included.

Indeed, for a 8 cycle delay (which gives a large slack in the design), performance is unaltered. Only when using large (and unrealistic) 128-cycle delays the performance is impacted and the benefits of the GCN are cancelled (performance is similar to the HAMMER_BCAST approach).

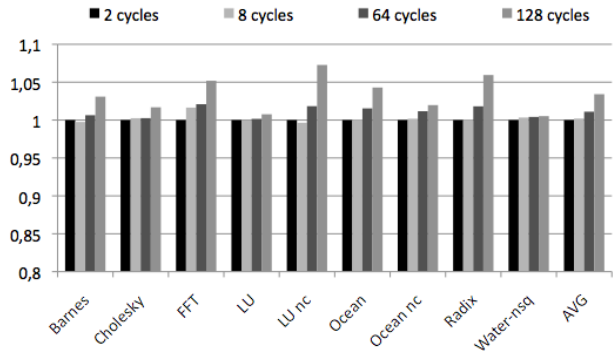

Fig. 12. Execution time with different gather network delays.

IV. Related Work

Cache coherence protocols have traditionally maintained a firm abstraction of the interconnection network fabric as a communication medium. More recently, however, some proposals exploring on-chip network optimizations for cache coherence protocols have appeared.

Cheng *et al.* [10] leveraged the heterogeneous interconnects available in the upper metal layers of a CMP, mapping different protocol messages onto wires of different widths and thicknesses. Flores *et al.* [11] propose to combine a protocol-level technique with the use of a simpler heterogeneous interconnect. Eisley *et al.* [12] propose an implementation of the coherence protocol within the network based on embedding directories in each switch node. In [13], it is presented a priority-based NoC, which differentiates between short control signals and long data messages to achieve a significant reduction in cache access delay. Walter *et al.* [14] explore the benefits of adding a low-latency, customized shared bus as an integral part of the NoC. More recently, Vantrease *et al.* [15] advocate nanophotonic support for building high-performance atomic cache coherence protocols.

The AMD's Coherent HyperTransport (TH) [6] implements the Hammer broadcast-based protocol enabling the construction of small-scale multiprocessors. Subsequently, the HyperTransport Assist [16], developed by AMD for the 12-core Magny Cours,

added a directory cache to reduce the frequency of broadcasts. Also, the Intel's QuickPaht Interconnect (QPI) implements two different protocol modes [17]. In one of them coherence transactions are broadcasted and every core must respond to the home with a *snoop response* that indicates the state of the block at that core. JETTY [18] and Blue Gene/P [19] are two proposals to filter the broadcast requests that would miss at destination nodes in order to reduce energy consumption due to cache look-ups. Filtering has also been proposed at source nodes [20], [21] to save energy and bandwidth.

Proposals for efficient multicast support in on-chip networks have also appeared [22]. Additionally, it has been evaluated the case of using this kind of support in combination with a cache coherence protocol implementing imprecise directories, demonstrating that multicast support is not enough to completely remove the performance degradation that the inexact sharing codes introduce [23].

V. Conclusion

The Hammer protocol scales to a large number of cores as it does not require large memory structures to keep the sharer's list typically found in directory-based protocols. However, the network traffic requirements of Hammer impacts the performance significantly. We have redesigned a standard mesh-based NoC and extended it to support a dedicated and set-aside gather control network, which is made of a set of AND gates along a tree wiring enabling the fast notification of ACK messages.

The wiring requirements for the gather control network keep low up to 256-core systems, and with a low delay of few network cycles. Area impact is lower than 3% of the switch area.

With the GCN support and built-in broadcast, the Hammer protocol is able to cope with the performance of directory based approaches, even achieving better performance. Network traffic is reduced significantly and put on par to directory-based protocols. Therefore, the Hammer protocol behaves as the directory-based protocols without the expensive control structures required, thus enabling Hammer to effectively scale.

Acknowledgements

This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grant TIN2009-14475-C04. It was partly supported by the project NaNoC (project label 248972) and by the VIRTICAL project (grant agreement 288574) which are funded by the European Commission within the Research Programme FP7.

References

[1] J. Rattner, "Single-chip cloud computer: An experimental many-core processor from intel labs, available at download.intel.com/pressroom/pdf/rockcreek /sccannouncement," .

[2] "Tile-gx processors family, available at http://www.tilera.com/products/tile-gx.php," .

[3] L. Benini and G. De Micheli, *Networks on chips: technology and tools*, Academic Press, 2006.

[4] J.Leverich, H.Arakida, A.Solomatnikov, A.Firoozshahian, M.Horowitz, and C.Kozyakisy, "Comparing memory systems for chip multiprocessors," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 358–368, 2007.

[5] D.J. Sorin, M.D. Hill, and D.A. Wood, *A Primer on Memory Consistency and Cache Coherence*, Morgan & Claypool Publishers, 2011.

[6] P. Conway and B. Hughes, "The amd opteron northbridge architecture," *IEEE Micro*, vol. 27, no. 2, pp. 10–21, March 2007.

[7] M.Lodde, T.Roca, and J.Flich, "Heterogeneous network design for effective support of invalidation-based coherency protocols," in *Proc. of the 2012 Interconnection Network Architecture: On-Chip, Multi-Chip Workshop*, January 2012.

[8] "The nangate open cell library, 45nm freepdk, available online at https://www.si2.org/openeda.si2.org/projects/nangatelib/," .

[9] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "Graphite: A distributed parallel simulator for multicores," in *The 16th IEEE Intl. Symp. on High-Performance Computer Architecture*, January 2010.

[10] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. Carter, "Interconnect-aware coherence protocols for chip multiprocessors," in *In Proc. of the 33rd Intl. Symp. on Computer Architecture*, June 2006, pp. 339–351.

[11] A. Flores, J.L. Aragon, and M.E. Acacio, "Heterogeneous interconnects for energy-efficient message management in cmps," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 16–28, January 2010.

[12] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," in *In Proc. of the 39th IEEE/ACM Intl. Symp. on Microarchitecture*, December 2006, pp. 321–332.

[13] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, "The power of priority: Noc based distributed cache coherency," in *In Proc. of the 1st Intl. Symp. on Networks-on-Chip*, 2007, pp. 117–126.

[14] I. Walter, I. Cidon, and A. Kolodny, "Benoc: A bus-enhanced network on-chip for a power efficient cmp," *IEEE Computer Architecture Letters*, vol. 7, no. 2, pp. 61–64, July-Dec .2008.

[15] D. Vantrease, M. Lipasti, and N. Binkert, "Atomic coherence: Leveraging nanophotonics to build race-free cache coherence protocols," in *In Proc. of the 17th Intl. Symp. on High Performance Computer Architecture*, February 2011, pp. 132–143.

[16] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes, "Cache hierarchy and memory subsystem of the amd opteron processor," *IEEE Micro*, vol. 30, no. 2, pp. 16–29, March/April 2010.

[17] R.A. Maddox, G. Singh, and R.J. Safranek, "Weaving high performance multiprocessor fabric: Architecture insights into the intel quickpath interconnect," *Intel Press*, 2009.

[18] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary, "Jetty: Filtering snoops for reduced energy consumption in smp servers," in *Proc. of Intl. Symp. on High Performance Computer Architecture*, January 2001.

[19] V. Salapura, M. Blumrich, and A. Gara, "Design and implementation of the blue gene/p snoop filter," in *Proc. of Intl. Symp. on High Performance Computer Architecture*, February 2007.

[20] J. F. Cantin, M. H. Lipasti, and J. E. Smith, "Improving multiprocessor performance with coarse-grain coherence tracking," in *Proc. of Intl. Symp. on Computer Architecture*, June 2005.

[21] A. Moshovos, "Regionscout: Exploiting coarse grain sharing in snoop-based coherence," in *Proc. of Intl. Symp. on Computer Architecture*, June 2005.

[22] S. Rodrigo, J. Duato J. Flich, and M. Hummel, "Efficient unicast and multicast support for cmps," in *In Proc. of the 41st IEEE/ACM Intl. Symp. on Microarchitecture*, December 2008, pp. 364 – 375.

[23] A. Ros and M.E. Acacio, "Evaluation of low-overhead organizations for the directory in future many-core cmps," in *Proc. of the 4th Workshop on Highly Parallel Processing on a Chip*, September 2010, pp. 87 – 97.