Interaction of NoC design and Coherence Protocol in 3D-stacked CMPs

Pablo Abad, Pablo Prieto, Lucia G. Menezo, Adrian Colaso, Valentin Puente and Jose Angel Gregorio
University of Cantabria
Santander, Spain
mail: {abadp,prietop,gregoriol,colasoa,vpuente,monaster}@unican.es

***Abstract*—Computer architectures have evolved to structures where communication has become an essential part of the system and most of it currently takes place inside the chip. The number of on-Chip cores and the available off-chip bandwidth is not growing at the same rate. This demands for the inclusion of more sophisticated memory hierarchies inside the chip to deal with off-chip latency and bandwidth problems in order to keep on improving performance. The exhaustion of Moore's law will accelerate the use of 3D-Stacked on-chip memory hierarchies to sustain the required scalability of forthcoming CMPs. For this class of systems' memory hierarchy, coherence protocol and interconnection network are two closely related components, but which are usually designed independently. In this work we will demonstrate that network components can be coupled to coherence protocol in order to extract significant performance benefits. Making use of a well-known snoop coherence protocol, we will present different network optimizations, better able to adapt to the communication requirements of this protocol. Evaluation results show that with minimal hardware changes, for some real applications, full system performance can be improved by up to 48%.**

Keywords—Network on Chip; Routing; Cache Coherence; Chip Multiprocessor

I. INTRODUCTION

Processor and memory speed divergence is a well-known issue to computer architects. Many researchers have come up with solutions able to reduce this growing gap to enable faster computers. However, with the advent of chip multiprocessors, off-chip bandwidth limitations will be the most limiting danger [34]. As on-chip core count increases, the available off-chip bandwidth per processor is reduced, adding high contention delays to the main memory accesses. The work in [34] predicts that in only a few generations, most on-chip area should be exclusively devoted to on-chip caches if we want to keep pace with performance improvements. Traditional memory technologies are becoming an impediment due to their limited density, cost, reliability, etc., which has obliged designers to search for alternative technological and/or architectural solutions for this problem. Vertical stacking of multiple silicon layers (i.e. 3D stacking or 3D IC) or alternative non-volatile memory technologies seem to be two good alternatives able to alleviate off-chip pressure, significantly increasing on-chip available cache capacity.

3D stacking not only provides the chance to increase the number of devices per chip, but also introduces a number of novel and interesting properties. A particular one, affecting the communication substrate, is that the characteristics of physical connectivity vary considerably depending on the dimension. On the one hand, communication latency along the third dimension is orders of magnitude lower than in the other two dimensions. On the other hand, the available connectivity across layers will allow a substantial on-chip bandwidth improvement [26]. This means that density problems of SRAM technology could be relaxed if part of the layers is exclusively devoted to LLC.

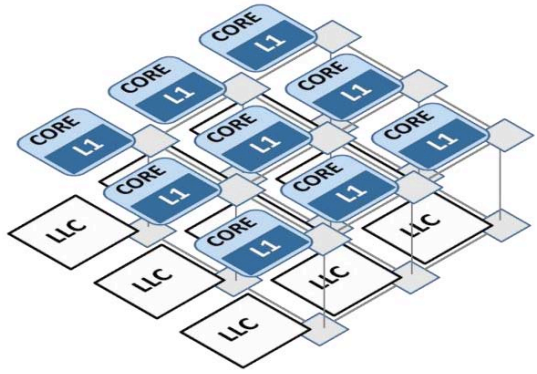

Fig. 1. *Three-dimensional CMP sketch.*

Among all the 3D stacking techniques, Through Silicon Vias (TSVs) allow the integration of multiple layers in the same fabrication process, optimizing vertical communications in aspects such as latency or energy. Thanks to the improvements achieved in TSV fabrication, pillar dimensions have been reduced significantly in recent years [14], increasing the available bandwidth between layers. Density and low latency make this technology one of the most promising solutions for future CMP performance scalability. However, vertical stacking is not exempt from serious problems that currently limit its scalability. With the growing power density derived from smaller transistors, heat dissipation has become a first-order issue for present and future microprocessor generations. 3D stacking further complicates this issue, because the temperature overhead rapidly increases as more CMOS layers are vertically stacked [33]. This complication might not allow the stacking of a large number of layers shortly, making

DOI 10.1109/DSD.2013.16

structures with "short stature" more realistic. A reasonable example of short-to-mid term 3D CMP architectures is sketched in Fig. 1. In a layer, cores are located next to a small one or two-level private memory, while the remaining layer is employed for last-level caching. The Last-Level Cache (LLC) is accessible from the cores through TSV. This organization is known as memory-on-logic.

When this kind of architecture is used for general-purpose processors, the commonly accepted consensus is that shared memory is the most productive programming paradigm [7]. In order to avoid the LLC access becoming a system bottleneck, NUCA [16] structures are the most suitable organization. NUCA organization is combined with scalable point-to-point interconnection networks with short ultra-wide links, providing a large bandwidth within the chip [8]. To maintain system correctness, coherence invariants for multiple copies of data blocks must be guaranteed. This can be done by using protocols originally conceived for off-chip systems, such as directory-based ones [13], but their utilization increases data access latency due to the burden of multiple indirections across the chip. The portion of the chip reachable per clock cycle is shrinking as the technology advances, making indirections more expensive. Therefore, taking advantage of bandwidth availability to avoid adding extra delay makes sense. Snoop-based protocols running on top of scalable interconnection networks provide the best design choice for CMP systems if the number of cores is not very high. In fact, currently, most commercial aggressive CMPs, such as [21][32], use this approach.

Hardware-based cache coherence and more particularly broadcast-based protocols impose an important set of characteristics and limitations on network traffic. These special features include both correctness (end-to-end deadlock or in-order delivery) and performance ones (broadcast communications). In this work we analyze how to maximize the use of the protocol features from the network perspective, in order to improve system performance. More specifically, assuming an architecture as the shown in Figure 1, we will demonstrate that very simple behavioral modifications of the interconnection network, based on the specific needs of the coherence protocol, can improve system performance at an almost negligible cost. Gains have been determined for a representative (covering different network-utilization scenarios) set of benchmarks and evaluated in a full system simulator that accurately models software and hardware components in the CMP. Even with such localized and simple architectural modifications, our proposal is able to improve system performance by up to 50% in some applications.

The rest of the paper is organized as follows. Section II provides a brief summary of prior work in the same area. Section III describes in detail the protocol employed in this work, and how its special features interact with the communication substrate. Section IV presents the set of network optimizations proposed in order to take advantage of coherence protocol. In Section V, an exhaustive evaluation of our proposals is carried out, through a full-system evaluation infrastructure. Finally, Section VI states a few conclusions and future work.

II. Related Work

Tight protocol-network interaction is a well-known issue in general-purpose computing. Over recent years, researchers have explored different ways to efficiently combine communication and coherence invariants. Merging both components, designing networks to support protocol requirements or adapting protocols for the network substrate are the main ways to extract maximum performance from memory hierarchy. The technical literature is profuse [19], and the aim of this section is only to provide a brief summary of a few relevant approaches.

Among the works designed to optimize performance by integrating the coherence protocol's behavior in the interconnection network, a good example is that of Eisley et al. [11]. This approach implements part of the coherence protocol on each router in the network. This allows actions to be added during message transit toward the destination that optimize the protocol and thus improve performance. However, these solutions create strong relationships between coherence protocol and interconnection network. Changes in one of them affect the behavior of the other, complicating even more the already intricate process of designing and verifying a coherence protocol.

In contrast to the previous example, most of the works aim for network modification to support protocol features. There is a broad range of aspects where network tuning can provide significant benefits. Concerning correctness, the interconnection network can release protocols from correctness issues such as snoop ordering [4] or reduce the hardware overhead of end-to-end deadlock support [2]. In contrast, performance-oriented solutions can provide efficient support for collective communications [18][25][22] or reduce the penalty of high-latency communications through highly efficient pipelines [23] or through micro-architectures able to deal with high congestion levels [15].

Finally, some works have already started moving one step further, working to understand coherence protocol and system organization interaction. This way, better component placement [3] or heterogeneous router configurations [37] provide relevant performance benefits at minimal cost. This paper shares the same motivation, but extending it to a technological environment such as 3D stacking. We will exploit the 3D properties in order to accelerate coherent protocol responsiveness through interconnection network design.

III. Protocol-Network Interaction

In this section, we will target our design constraints. Like in many commercial systems [6][17][21] we will focus our attention on a broadcast-based coherence protocol. Initially, we will briefly describe the protocol. Later, we will describe how some protocol features, in a system like the one described in the previous section (Fig. 1) can exploit special features of network traffic.

A. Cache Coherence Technique. TokenB

There are two major groups of techniques to maintain consistency invariant in multiprocessor systems based on

interconnection networks other than a single bus. On the one hand, there are techniques based on *directory*, namely the existence of a centralized structure (usually distributed in slices) which contains the information needed to locate any datum that is within the chip. The main advantage of this technique is the low bandwidth required and the two main disadvantages are the increased latency due to the indirection to this structure and the space required to store the directory itself. On the other hand, there are broadcast-based coherence protocols, in which the discovery of shared data is performed by requesting it from all elements of the system that can store a copy of it. Thus, cache-to-cache transfers are performed very efficiently at the expense of increasing bandwidth requirements and the storage overhead of keeping shared information is avoided. In on-chip environments, coherence protocols should use bandwidth availability to avoid indirections, making broadcast-based techniques preferable [4][6][28].

A technique belonging to the latter group is called *tokenB* [28]. This technique associates a fixed number of tokens with each block. In order to write a block, a processor must acquire all the tokens. To read a block, only a single token is needed. In this way, the coherence invariant is directly enforced by counting and exchanging tokens. In *TokenB*, coherence requests are broadcast directly from the requesting processor to all other coherency controllers in the CMP. Only caches sharing the block should respond with an acknowledgement message. The main advantage of this technique is that it separates correctness from performance. Precisely this separation helps to improve performance through better use of the links in the underlying interconnection network, without compromising the correctness of the protocol.

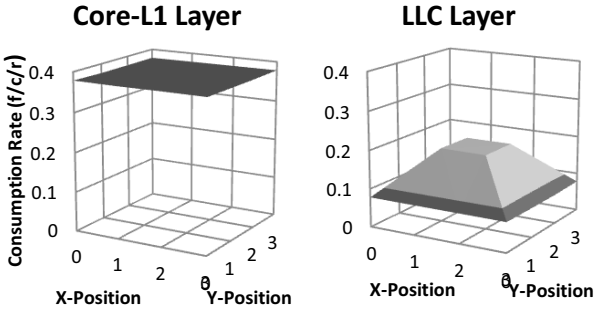

Fig. 2. *Network consumption rate in each CMP Layer (flits/cycle/router).*

B. Unbalanced traffic distribution

Combining the broadcast-based protocol, TokenB, with a three-dimensional CMP organization similar to the one proposed in Fig. 1, every broadcast request with *Num_procs (Num_procs-1 cores and one LLC bank)* destinations has an unbalanced distribution. Every private L1 receives a request, as well as the bank at LLC where data should be depending on its address. For this reason, most of the destinations (*Num_procs-1*) are within the core layer, while only one destination is in a different layer. This fact imposes most of the pressure on XY links in the core layer, keeping other network links unoccupied most of the time. To prove this, we have performed a simple experiment measuring destination distribution for every application from the employed benchmarks depicted in Section V. All the applications show a similar behavior, providing here only the results of one of them for the sake of brevity. The results in Fig. 2 represent the injection/consumption rate of every router at each layer for the *apache* workload. As can be seen, the consumption rate in the core layer is more than twice as large as the one in the LLC layer. This direct effect of protocol-organization interaction will obviously cause unbalanced link utilization and therefore an increase in the network congestion. Early appearance of congestion has a negative impact on network performance, causing network latency to grow significantly in adverse scenarios. This, ultimately, will increase average access time for memory operations, which will degrade system performance.

C. End-to-end deadlock

In cache coherent systems, split-transactions are necessary in order to achieve minimal performance requirements. In a memory transaction, a chain of messages, each one with a different purpose or nature, could be involved. For example, in a simple request-reply protocol, such as the one employed in Dash [24], two classes of messages can be identified: request commands and data responses. In more advanced protocols, such as Quick Path Interconnect [17], there are six types. This characteristic has to be considered during the network design process in order to guarantee system correctness.

In cache coherent CMP (cc-CMP) systems, the routers are connected to coherence or memory controllers. Those coherence controllers have a limited capacity to store pending memory transactions. In order to process some transactions, the controller needs to generate additional network messages (a reply generated for a request message is the basic example). If this "reply" buffering is exhausted, there is no way to drain additional "request" messages from the network, and a cyclic dependency involving two different controllers could be generated, permanently blocking message advance. This anomaly is known as message-dependent deadlock [35] and it has been widely studied in cache coherent systems since the initial cc-NUMA prototypes [24].

The most common and cost effective solution to this problem is to use virtually separated networks for each class of messages [17]. Thus, request and reply messages do not share network resources and the cyclic dependency generated at controllers is broken. This solution is used extensively not only in cc-CMP but also in application-specific systems where application traffic is reactive, including peer-to-peer streaming or slave locking [31][36]. Dividing network resources among message types means that network traffic seen by different resources could have distinct characteristics. For example, the traffic from the virtual network of L1-miss requests never moves from LLC Layer to Core-L1 layer, because the message source always belongs to the Core-L1 layer.

IV. NoC Support for Coherence Protocols in 3D-stacked CMPs

On-chip cache levels distributed among different 3D layers combined with broadcast-based coherence protocols lead to a scenario where network traffic presents specific features. As we have described in the previous section, different congestion

levels in each layer and additional correctness requirements due to the reactive traffic nature determine traffic shape. In this section we will analyze in more detail certain traffic aspects derived from these features and explain how to combine them with network design decisions in order to improve system performance.

A. *Message Class-Aware Routing in a 3D-stacked CMP*

In those networks where deterministic routing is employed, the common assumption is that every message in the network progresses following the same policy. However, as end-to-end deadlock avoidance precludes the utilization of different virtual channels (reserved for different message types), each one could employ a different routing policy without causing routing-induced deadlocks. The only condition to ensure correctness is that each individual routing protocol must be deadlock free. In those cases where a uniform distribution of network traffic is common, the utilization of different routing protocols might not mean substantial performance differences. However, in our case, where traffic density is highly dependent on Z position, the combination of different policies can lead to significant performance benefits.

Due to the implementation cost constraints, the most extended routing protocol in the NOC environment is Dimension-Ordered Routing (DOR) [9]. In a 3D memory-on-logic CMP, forcing every message to follow the same dimension order could lead to situations where part of the messages could be unnecessarily delayed due to network congestion. To explain this, we will assume that every network message is routed with the following dimension order: X-Y-Z. Every time a L1-Miss occurs, a request must be sent through the network in order to obtain the missing block. A request message is sent to each private L1 controller as well as to the L2[1] bank where data might reside. In a configuration like the one in Fig. 1 this means that 16 messages are generated, the destination of 15 of them is in the Core-L1 layer and only one message is sent to the LLC layer. In those cases where L1 miss rates are significant, network resources belonging to the L1-Core layer will rapidly become congested. As messages sent out to the LLC are also forced to move through the X and Y dimensions first, their delay will be significantly increased due to congested resources before moving to the Z dimension.

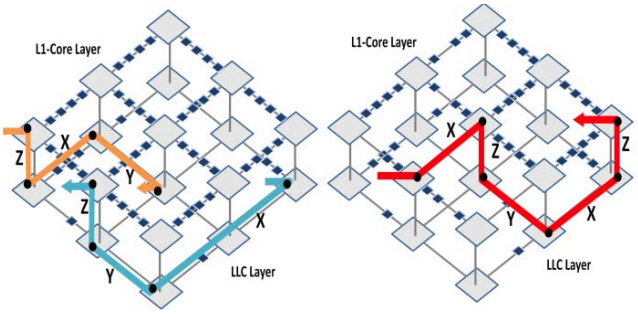

Fig. 3. *(a) Different dimension-order routing according to source-destination pair. (b) Message missrouting due to L1-Core layer congestion.*

[1] We will assume two levels of cache in the on-chip hierarchy, so we will employ indistinctly the term LLC and L2 cache

A straightforward solution to this problem would be simply to interchange the order in which dimensions are traversed, moving from X-Y-Z to Z-X-Y. However, if this change is applied to every message type, the problem is not solved. If we firstly move request messages to the Z dimension, LLC requests avoid the congestion at L1-Core layer, but in this case reply messages from a LLC Bank to a L1 cache will suffer L1-Core layer congestion unnecessarily. In the previous case (X-Y-Z), replies only employ X-Y links from LLC-Layer, performing their last hop in Z dimension to reach their destination in L1-Cache Layer. Moving routing protocols to Z-X-Y, we are forcing replies to traverse most of the network through the most congested layer, increasing latency and congestion in that layer.

Supported by VC separation, a solution to this problem is to employ a different routing protocol depending on the source-destination pair of the messages belonging to that virtual channel. The way to decide the routing for a virtual channel is decided as follows:

- If Source is in congested layer and Destination is in free layer then Routing=Z-X-Y.
- If Source is in free layer and Destination is in congested layer then Routing=X-Y-Z.
- In any other case routing selection is not relevant.

In the case of the broadcast-based coherence protocol this means that the virtual channel devoted to requests caused by L1 misses will be routed in Z-X-Y order. In this virtual channel, the message destination can be in either layers, but the source is always in the L1-Core layer. In contrast, any reply from LLC to an L1 Cache (routed through a different virtual channel) will make use of the opposite dimension order. This kind of reply is always generated in the LLC layer and the destination is in the L1-Core Layer. An example of this message-dependant routing is depicted in Fig. 3.a. Other messages such as LLC requests to main memory, where source and destination are in the same layer, can be routed with any dimension order, because they do not make use of the Z dimension.

The hardware overhead required to implement per-VC routing is minimal. For a fixed topology and routing strategy, algorithmic routing is often more efficient in terms of area and latency [9]. Inside the router, a circuit accepts information concerning direction and distance for each dimension and generates a vector indicating which outputs advance the packet to destination. A secondary circuit selects the appropriate output from the vector according to routing policy. The only additional logic required must modify output selection in order to consider message virtual channel to calculate destination vector. A few gates are enough for the implementation, which makes the area/latency/power overhead negligible.

B. *Congestion-aware missrouting*

The previous solution is applicable to those messages that change layer, but there is still a significant amount of traffic traveling only in the X and Y dimensions through the congested L1-Core layer. In the case of a broadcast-based coherence protocol, such as the one employed here, this traffic

corresponds mainly to the L1-miss requests that ask for the data from the private cache levels of the rest of CMP cores. With a Dimension-Ordered routing policy these messages will not be able to take advantage of the under-utilized resources of the LLC Layer.

In this case, we will exploit another feature of this coherence protocol, which is the non-utilization of certain routing directions in a given virtual channel. We will take the L1 broadcast requests as an example. In this case, all source routers are placed in the same layer (L1-Core), while destinations can be found in any layer. This means that this kind of messages makes use of the X-Y links in both dimensions, but only employs the Z links to move from L1-Cache to LLC layer (Z_{DOWN} in Fig. 3.b). A message through this virtual channel never moves from an LLC to a L1 cache, eliminating $Z_{UP} \rightarrow X$ or $Z_{UP} \rightarrow Y$ turns from message routes.

This traffic characteristic allows us to perform non-minimal routing (i.e. misrouting) under certain conditions without causing a routing deadlock. The only condition that must be fulfilled by non-minimal routes is to eliminate any cyclic dependency among network resources [10]. We will make use of this feature to misroute to LLC layer those request messages moving through the L1-Core layer that find congested resources. The mechanism is simple and can be easily explained through the example in Fig. 3.b. In any L1-Core layer network router, any time a packet arbitration for a X or Y link is rejected, the Z_{DOWN} link is also requested. If the Z_{DOWN} link is granted, the message is misrouted to the lower layer. From this moment, the message advances following an XY route through the LLC layer until reaching the router just below its destination. The final hop in the Z_{UP} direction returns the message to the upper layer and to its destination router. Thus, the pressure on congested links is relaxed, and network resource utilization will be more uniform.

As mentioned before, the way to guarantee that this new routing is deadlock free is avoiding $Z_{UP} \rightarrow X$ or $Z_{UP} \rightarrow Y$ turns. This is achieved by only letting misrouted messages return to their original layer once they have consumed the X and Y dimensions. The last hop is performed following a Z_{UP} link and the message reaches its destination, always requesting the consumption port and eliminating the possibility of cyclic dependencies.

Again, the hardware overhead to implement misrouting functionality is minimal. In this case, those messages that have not been misrouted (one bit at header indicates this) and that belong to the requested virtual channel activate two outputs from the destination vector, the one corresponding to conventional DOR and Z_{DOWN}. Then a double request is performed to arbitration logic. If the conventional port is not granted but Z_{DOWN} is, the Z distance is updated and the message marked as misrouted. Once a message is marked, only conventional DOR requests are generated. Since the vertical distance is very short, energy or delay penalization of this misrouting would be minimal.

C. Critical Flit First

Processors usually need one word of a block at a time. In order to make the L1 load miss latency independent of block size, most processors employ strategies to avoid waiting for the full block to be loaded before restarting the processor. A well-known technique, named critical word first (CWF), requests the missed word from memory in first place, sending it to the processor as soon as it arrives. The processor continues execution while the rest of the words in the block are being filled into L1 cache.

As network messages are normally broken into smaller pieces due to the limited on-chip bandwidth, this block re-ordering could be implemented by communication components with very low overhead. In the first place, L1 Load misses must indicate, in request messages, where the missed word is in the block. In those cases where the memory address of a miss already specifies the word offset inside the block, the word position in the block can be inferred from the address directly. According to the position, the flit in which this word would reside with conventional ordering is calculated. This flit number is the information coded in the request message header, requiring only $log_2[flit\text{-}number]$ bits to encode it. Once the LLC or Memory controller provides the network interface with the missing block, this flit number is employed to rotate flit position in the network message, putting the one with the critical word in the first place. The reply message only needs to carry information about the original position of the first message flit. The same number of bits as in the request message is required to encode this information. After reaching the destination, this additional flit is detected by the network controller, forwarding the flit to the cache controller before the whole packet is re-assembled.

As re-assembling hardware is a necessity in network interfaces (i.e., coherence controller), the implementation of flit re-ordering at this point reduces the logic overhead required, reducing the complexity of the finite state machine in the coherence controllers. In an environment where low communication latencies are essential and on-chip bandwidth is usually not enough to move a whole block at a time, this mechanism can have a significant effect.

V. EVALUATION

The framework employed for evaluation allows us to perform full-system simulation with complex workloads running on top of the Solaris 10 O.S. The simulator is based on SIMICS [27], extended with different modules capable of faithfully modeling the architecture of the cores and the memory hierarchy. GEMS [29] provides detailed timing models for state-of-the-art processor (OPAL) and memory hierarchy (RUBY). GEMS module for interconnection network simulation have been replaced with TOPAZ [1], which models network architecture accurately and allows the simulation of 3D topologies.

The main parameters of the simulated system are shown in Table 1. The simulated CMP has 16 aggressive OOO processors with static shared S-NUCA L2. The system layout uses a 3D Mesh to connect the 16 cores and 16 L2 banks. The cores operate at 4GHz and the memory subsystem at 2GHz. The selected protocol was a well-known snoop coherence protocol (Token B) [28] with six different message types, because commercial products such as AMD's Hypertransport [6] or Intel's QPI [17] employ protocols with similar

characteristics and requirements. The workloads used in this study, listed in *Table 2*, are multi-threaded; four commercial and five scientific programs. The numerical applications are part of the NAS Parallel Benchmark (OpenMP implementation) [20], while the commercial benchmarks correspond to the Wisconsin Commercial Workload suite [5], released by the authors of GEMS in version 2.1.

Table 1. Main parameters of the simulated system.

Processor Config.	Number of Cores	16@4GHz
	Functional Units	4xI-ALU / 4xFP-ALU / 4xD-MEM
	IWin Size / Issue Width	128 / 4-way
	Fetch-to-Dispatch	7 cycles
L1 Cache	Size / Associativity / Block Size / Access Time	32KB, 2-way, 64B block, 2-cycle
	Max Outstanding Mem. Operations	16
L2 Cache	Size / Associativity / Block Size	16MB / 16-way, 16 banks, 1 bank per router / 64B
	NUCA Mapping	Static, interlieved across slices
	Slice Access Time	5 cycle
Memory	Capacity / Access Time / Memory Controllers / BW	4GB / 250 cycles / 4 centered / 320GB/s
Network	Topology / Link Latency Link Width	4x4x2 Mesh / 1 cycle / 128 bits (or 64)
	Router Latency / Buffer Size / Routing	3 cycles / 10 flits per VC / DOR

A. Performance Results

The results of the graphs in this section are obtained through a variable number of runs for each application with pseudo-random perturbation (adding a small random delay to each memory access) in order to estimate workload variability [5]. All the results provided have a 95% confidence interval. The *y* axis represents execution time values, normalized against the baseline case where no optimization is applied to the communication substrate.

Table 2. Workloads considered for evaluation

Benchmark	Description
Wisconsin Commercial Workload Suite	
Apache	Task-parallel web server
Jbb	Java middleware application
Zeus	Pipelined web server
Oltp	Pseudo TCP-C on-line trans. processing
NAS Parallel benchmark	
FT	3-D partial diff. eq. solution using FFTs
IS	Integer sort
SP	Scalar Pentadiagonal solver
MG	Multi-grid on a sequence of meshes
LU	LU solver

The results in Fig. 4 represent the performance improvement obtained when the first of the proposed improvements is applied. The OPT-RTG column represents the results obtained when the contention-aware dimension order is applied according to the source-destination pair of each virtual channel. In this particular case, normal and persistent requests are routed in Z→X→Y order (virtual channels 1 and 6) and replies in X→Y→Z order (virtual channel 5). The rest of the virtual channels only move through one layer, the order being irrelevant. The performance benefits are strongly dependent on the congestion levels reached in the L1-Core cache Layer. In the case of those transactional applications where network traffic is low, benefits are less than 5% performance improvement. However, demanding applications such as FT or SP can benefit more from this simple optimization, reducing execution time by 25% in the case of the FT application. The average value strongly depends on the mix of applications selected. For our two benchmark suites this value is ~6%, which is a significant result for an almost cost-free modification.

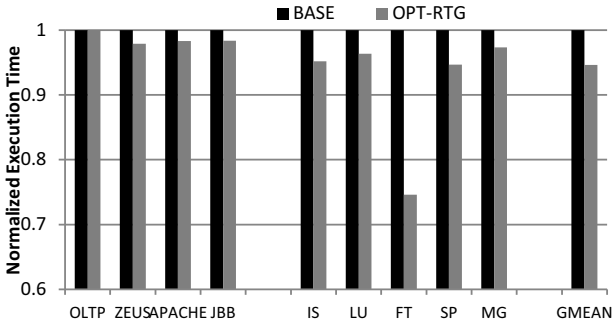

Fig. 4. *BASE-normalized execution time. Traffic-Aware Routing results.*

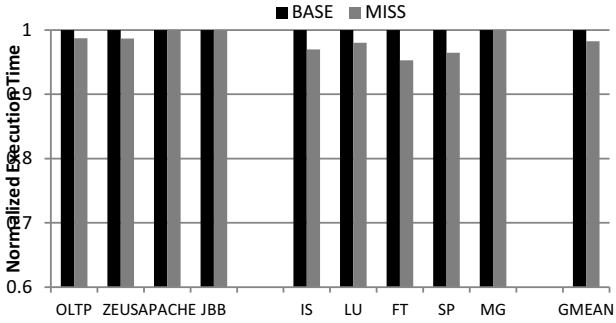

Fig. 5. *BASE-normalized execution time. Congestion-aware missrouting results.*

Fig. 5 represents the results obtained when message misrouting through LLC layer is applied to CMP network traffic. In this case, we obtain a similar tendency as in our previous set of results, the performance benefits being more significant for more network-demanding applications. However, the improvement is less for this mechanism. The reason for this result is the latency overhead induced by misrouting. Every time a message is misrouted, it is forced to traverse a longer path to its destination. These messages must traverse two additional routers and two additional links compared to messages following a minimal route. In our particular case, with a 4x4 mesh topology, a router pipeline of 3 cycles and a link delay of 1 cycle, this means a base latency overhead of more than 20%. Even with this handicap, none of the applications evaluated has obtained worse performance

results than the BASE implementation. In cases such as JBB, APACHE or MG, the improvement is negligible, but again other applications are able to improve their execution time. It should be noted that these performance improvements have been obtained by very simple modifications in the network.

Finally, we have also performed a third experiment analyzing the impact of flit re-ordering in order to avoid message spooling latency in load misses. While previous results have been obtained for 128-bit links, in this case we have employed two different link widths for our evaluation, 128 and 64-bit wires. Depending on link width, reply messages containing a cache block must be broken into a different number of flits. For a 64-Byte block, 5-flit messages will be necessary for 128-bit links and 10-Flit for 64 bits (or using 128 byte blocks with 128-bit wires). Results for these two message lengths are shown in Fig. 6. The first noteworthy aspect is the big difference between 5 and 10-Flit results. In the first case benefits extracted from flit re-ordering are minimal, while results for 10 flits are completely different. Spooling latency of 5-Flit messages seems to be insignificant compared to network and contention latencies, which minimizes benefits. However, it seems that the benefits of spooling avoidance rapidly become significant as message length increases. As can be seen, for 10-Flit messages we are able to obtain a 10% performance benefit on average. Therefore, this should be taken into account if the relationship between cache block size and the width of the network links is modified. Note that we are using a conservative 3-cycle router pipeline. Using a single cycle pipeline [30], the results would be much more relevant.

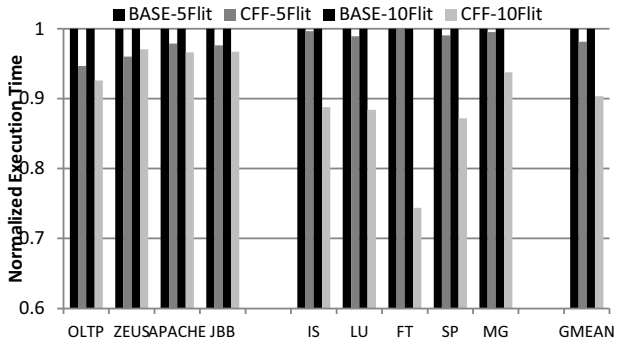

Fig. 6. *BASE-normalized (independently for each message size) execution time. Critical Flit First results.*

B. Putting it all together

The previous section provides some insight into the effect of each technique on overall system performance. For this final experiment, we have included all the mechanisms proposed in this work in the same router micro-architecture, comparing it against the baseline case. The final set of results is shown in Fig. 7. Average values show a performance improvement of 10% in the case of 5-Flit reply messages, while this improvement grows to 20% when network link width is reduced from 128 to 64 bits. Applications with large communication demands can obtain significant performance benefits from the solutions proposed in this work. With a minimal overhead, we have been able to halve the execution time of the FT application. Even in those cases where network pressure is less relevant, benefits are still extracted from protocol-aware routing and flit re-ordering.

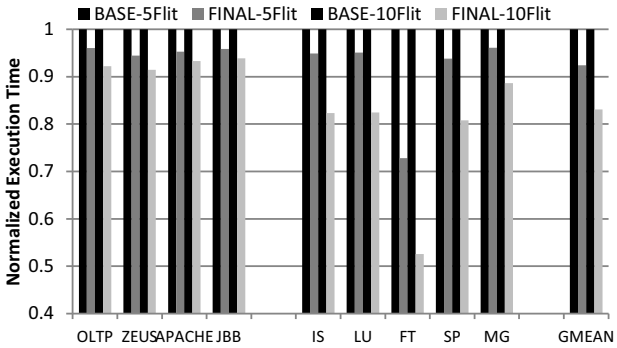

Fig. 7. *BASE-normalized (independently for each message size) execution time results. All mechanisms together.*

The main reason for these results is the more efficient utilization of network resources. Through a more intelligent routing policy we are able to distribute traffic volume in a more uniform way. To prove this, we have measured the link utilization in each layer for both BASE and FINAL configurations. In the results presented in Fig. 8, the y-axis represents the fraction of time that the links of each layer are occupied by a message in transit. It can be seen that the routing techniques proposed in this work are able to reduce the initial difference in link utilization, partially reducing the pressure exerted on L1-Core layer.

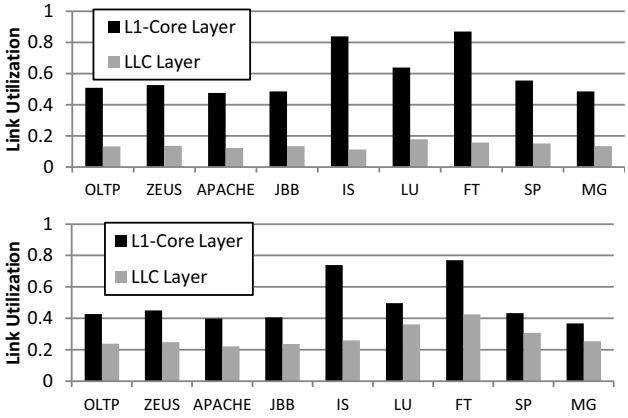

Fig. 8. *Fraction of time network links are busy in each layer. (above) BASE configuration, (below) FINAL configuration.*

VI. CONCLUSIONS AND FUTURE WORK

We present a set of network enhancements that take advantage of 3D stacking properties in order to improve cache coherent CMP performance. Through these solutions, network resources are utilized more efficiently, distributing protocol messages among the different network layers. With a negligible hardware modification, the router is able to implement per-virtual channel routing policies and also miss-routing strategies that help to alleviate congestion in the L1-Core CMP Layer. Results show that for some real applications,

our proposals can improve overall system performance by up to 48%.

This paper has provided a set of routing modifications favored by protocol peculiarities. Different strategies, such as adaptive routing policies, could be evaluated for this environment, analyzing whether the improved performance outweighs the increased coherence protocol complexity by eliminating the restriction of in-order delivery. Finally, special routing features could also be implemented with an alternative final target to performance, such as temperature control, a first order design constraint in three-dimensional environments.

ACKNOWLEDGMENT

The authors would like to thank Jose-Angel Herrero for his valuable assistance with computing environment (HPC cluster Calderon within the datacenter 3Mares) and the anonymous reviewers for many useful suggestions. This work has been supported by the MICCIN (Spain) under contract TIN2010-18159, MECD (Spain) under grant PRX12/00006, and by the HiPEAC European Network of Excellence.

REFERENCES

[1] P.Abad, P. Prieto, L.G. Menezo, A. Colaso, V. Puente, and J.A. Gregorio, "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers", NOCS 2012.

[2] P.Abad, V. Puente, J.A. Gregorio, "Reducing the Interconnection Network Cost of Chip Multiprocessors", International Symposium on Networks-on-Chip (NOCS), March 2008.

[3] D. Abts, N.E. Jerger, J. Kim, D. Gibson, M. Lipasti, "Achieving predictable performance through better memory controller placement in many-core CMPs", International Symposium on Computer Architecture (ISCA), June 2009.

[4] N. Agarwal, L.S. Peh, N.K. Jha, "In-Network Snoop Ordering (INSO): Snoopy Coherence on Unordered Interconnects", International Symposium on High Performance Computer Architecture (HPCA), February 2009.

[5] A.R. Alameldeen, et. al., "Simulating a $2M Commercial Server on a $2K PC", IEEE Computer, February 2003.

[6] A. Ahmed, P. Conway, B. Hughes, F. Weber, "AMD OpteronTM Shared Memory MP Systems," Conference Record of Hot Chips 14, Stanford, Aug. 2003.

[7] K. Asanovic, R. Bodik, and B. Catanzaro, "The landscape of parallel computing research: A view from Berkeley," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-183.

[8] W.J. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", Design Automation Conference, June 2001.

[9] W.J. Dally, B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kaufmann Publishers, Inc. , 2004.

[10] J. Duato, "A theory of deadlock-free adaptive multicast routing in wormhole networks," IEEE Transactions on Parallel and Distributed Systems, vol. 6, pp. 976-987, 1995

[11] N. Eisley, Li-Shiuan Peh. and Li Shang, "In-Network Cache Coherence", IEEE Computer Architecture Letters, 5 no.1 2006.

[12] P. Gratz, C. Kim, R. McDonald, S.W. Keckler, D. Burger, "Implementation and Evaluation of On-Chip Network Architectures", International Conference on Computer Design (ICCD), October 2006.

[13] A. Gupta, W. Weber, and T. Mowry, "Reducing memory and traffic requirements for scalable directory-based cache coherence schemes," International Conference on Parallel Processing, 1990.

[14] S. Gupta, M. Hilbert, S. Hong and R. Patti, "Techniques for Producing 3D ICs with High-Density Interconnect" International VLSI Multilevel Interconnection Conference, September 2004.

[15] Y. Hoskote, et. al., "A 5-GHz Mesh Interconnect for a Teraflops Processor", IEEE Micro, vol. 27, issue 5, November 2007.

[16] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, S.W. Keckler, "A NUCA Substrate for Flexible CMP Cache Sharing", International Conference on Supercomputing (ICS), June 2005.

[17] Intel Corporation, "An Introduction to the Intel Quickpath Interconnect", White paper, Document Number 320412-001US, 2009.

[18] N.E. Jerger, L.S. Peh, M. Lipasti, "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support", International Symposium on Computer Architecture (ISCA), June 2008.

[19] N.E. Jerger, L.S. Peh, "On-Chip Networks, Synthesis Lectures on Computer Architecture", Morgan & Claypool Publishers, 2009.

[20] H. Jin, M. Frumkin, J. Yan, "The OpenMP Implementation of NAS Parallel Benchmarks and its Performance", NAS Tech. Report, 1999.

[21] R. Kalla, B. Sinharoy, W. J. Starke, and M. Floyd, "Power7: IBM's Next-Generation Server Processor" IEEE Micro, vol. 30, no. 2, 7-15, 2010.

[22] T. Krishna, L.S. Peh, B.M. Beckmann, S.K. Reinhardt, "Towards the Ideal On-Chip Fabric for 1-to-Many and Many-to-1 Communications" International Symposium on Microarchitecture (MICRO), 2011.

[23] A. Kumar, et. al., "A 4.6 Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS" International Conference on Computer Design (ICCD), October 2007.

[24] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, M.S. Lamet, "The Stanford Dash Multiprocessor", Computer, vol. 25 no. 3, March 1992.

[25] M. Lodde, J. Flich, M. Acacio, "Heterogeneous NoC Design for Efficient Broadcast-Based Coherence Protocol Support", International Symposium on Networks-on-Chip (NOCS), May 2012.

[26] G. H. Loh and Y. Xie, "3D Stacked Microprocessor: Are We There Yet?," IEEE Micro, vol. 30, no. 3, pp. 60–64, May 2010.

[27] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgen, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, B. Werner, "Simics: A Full System Simulation Platform", IEEE Computer, Vol. 35, no. 2, February 2002.

[28] M.M.K. Martin, M.D. Hill, D.A. Wood, "Token Coherence: Decoupling Performance and Correctness", International Symposium on Computer Architecture (ISCA), June 2003.

[29] M.M.K. Martin, et. al., "Multifacet´s General Execution-driven Multiprocessor Simulator (GEMS) Toolset", Computer Architecture News (CAN), September 2005.

[30] R. Mullins, A. West, S. Moore, "Low-Latency Virtual-Channel Routers for On-Chip Networks", International Symposium on Computer Architecture (ISCA), June 2004.

[31] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. Micheli, and L. Raffo, "Designing Message-Dependent Deadlock Free Networks on Chips for Application-Specific Systems on Chips," International Conference on Very Large Scale Integration, 2006.

[32] C. Park et al. "A 1.2 TB/s on-chip ring interconnect for 45nm 8-core enterprise Xeon® processor" In 2010 IEEE International SolidState Circuits Conference(ISSCC), 180-181, 2010.

[33] K. Puttaswamy, G.H. Loh, "Thermal analysis of a 3D die-stacked high-performance microprocessor" ACM Great Lakes symposium on VLSI, 2006.

[34] B. Rogers, A. Krishna, G. Bell, K. Vu, X. Jiang, Y. Solihin, "Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling", Proceedings of the 36th annual international symposium on Computer architecture (ISCA 09), pp 371-382, 2009.

[35] Y.H. Song, T.M. Pinkston, "A Progressive Approach to Handling Message-Dependent Deadlock in Parallel Computer Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 14, no. 3, 2003.

[36] P. van der Stok, "Dynamic and Robust Streaming in and between Connected Consumer-Electronic Devices", Kluwer, Dordrecht, 2005.

[37] S. Volos, C. Seiculescu, B. Grot, N.K. Pour, B. Falsafi, G. De Michelli, "CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers", International Symposium on Networks-on-Chip (NOCS), May 2012.