CCNoC: On-Chip Interconnects for Cache-Coherent Manycore Server Chips

Ciprian Seiculescu⋆, Stavros Volos§, Naser Khosro Pour‡,
Babak Falsafi§, Giovanni De Micheli⋆

⋆ LSI, EPFL, http://lsi.epfl.ch
§ PARSA, EPFL, http://parsa.epfl.ch
‡ ELAB, EPFL, http://elab.epfl.ch

ABSTRACT

Manycore chips are emerging as the architecture of choice to provide power-scalability and improve performance while riding the Moore's law. On-chip interconnects are increasingly playing a pivotal role in power- and performance- scalability of such microarchitectures. As supply voltages begin to level off in future technologies, chip designs in general and interconnects in particular are resorting to specialization to provide power- and performance-scalability.

In this paper, we make the observation that cache-coherent manycore chips exhibit a duality in on-chip network traffic. Request traffic typically consists of control packets requiring narrow low-power switches, while response traffic often carries cache block-sized payloads that require wider and higher-power switches. We present Cache-Coherence Network-on-Chip (CCNoC), a design to capitalize on this duality in traffic and provide a pair of asymmetric switches that optimize power and performance over conventional on-chip interconnects. Cycle-accurate simulation results for a 4x4 chip multiprocessor with a shared last-level cache running commercial server workloads indicate 22% improvement in power over a torus and 38% improvement in power over a mesh with larger channel width, while providing similar performance.

Keywords

Server chips, Networks on chip (NoC), topology, routing, power

1. INTRODUCTION

CMOS scaling is projected to continue for another decade resulting in exponential increases in chip integration levels [24]. Unfortunately, while chips continue to integrate more transistors, energy- and delay-scalability have dramatically slowed down, resulting in a paradigm shift in on-chip computing towards tiled chip multiprocessors (CMPs) interconnected using a network-on-chip (NoC). NoCs are not only pivotal to overall system performance, but can also dramatically affect a chip's power. For example, NoC power accounts for 40% in the MIT RAW [19] and 30% in the Intel Tera-scale [20] of the overall chip power. With supply voltages levelling off [24], the key design criteria for chips in general and NoCs in particular will be power.

There are a myriad of techniques that target reducing NoC power [1, 5, 8, 9, 11, 16, 21]. Unlike conventional designs

targeting a single NoC for tiled CMPs, Balfour and Dally [1] recently advocate using multiple sub-networks to optimize for performance, power and area in tiled CMPs. Using two sub-networks allows for optimizing the area and power for crossbars with narrower datapaths while improving wire utilization and bandwidth. Using simple read/write messaging protocols, the authors conclude that a homogeneous dual-tiered NoC where the two sub-networks use identical switches balances the load among short and long messages and as such is optimal for tiled CMPs.

High-end desktop and server chip designers opt for cache-coherent shared-memory CMPs for server software transparency and for facilitating software development and porting. In tiled cache-coherent CMPs, however, message traffic will depend on both the workloads and the distribution of the messages implementing the coherence protocol. In this paper, we make the observation that the protocol traffic in servers indeed does not follow simple read/write messaging protocols and is highly skewed among short and long messages with (short) request messages primarily consisting of block fetch requests and replacement notifications and (long) response messages carrying a cache block-sized payload. These results corroborate prior findings that coherence activity in tiled CMPs is dominated by data and instruction cache block fetches, and otherwise exhibits a relatively low frequency of dirty block evictions [4]. Moreover, dirty block evictions are often not on the critical path and as such do not affect performance as much as data and instruction cache block fetches. As such, homogeneous designs for dual sub-network NoCs would be suboptimal given the cache coherence traffic commonly found in servers.

We propose a heterogeneous *Cache-Coherence Network-on-Chip (CCNoC)* for manycore server chips organized as a tiled CMP, which incorporates two switches at each node to optimize power and performance for the dominant message type in each message class. A request switch is customized for control messages with narrow channels and reduced power, while a response switch, customized for cache block transfers, uses full-width channels. Physically partitioning the resources among the protocol messages also enables removing the virtual channels, otherwise needed to avoid protocol deadlocks.

We use FLEXUS [22] for cycle-accurate full-system multiprocessor simulation running server, scientific and multipro-
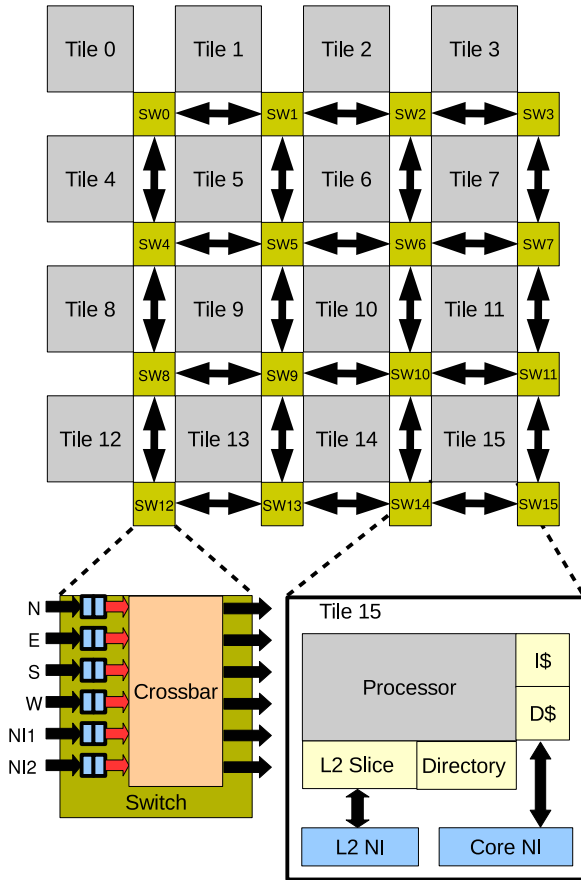
Figure 1: Tiled processor with mesh NoC.

grammed workloads and ORION 2.0 [7] to estimate power consumption and show that:

- Request and response traffic is skewed across a wide spectrum of multiprocessor and multiprogrammed workloads. Short requests account for 93% of the request messages and long responses account for 86% of response messages.

- CCNoC can reduce the power consumption significantly over a traditional mesh configuration by 16% while having similar performance.

- Compared to more aggressive topologies such as torus and mesh with wider channels, CCNoC reduces power consumption by 22% and 38% with a small performance loss of 2% and 8% respectively.

2. RELATED WORK

Several NoC research groups have proposed multi-NoC interconnect where one NoC is packet-switched and the second is circuit-switched [6, 13, 14, 15, 17]. In [6, 17] the packet-switched NoC is used to carry control information to configure the circuit switched NoC. In [13, 14, 15] the packet-switched network is used for non-localized traffic and the circuit switched network is used for localized traffic flows that require high bandwidth. These optimizations target applications with localized traffic patterns. However, in this work

we target server applications running on a cache-coherent CMP. The network traffic is uniformly distributed due to the coherence protocol and the fact that server applications reference a large working set. Therefore, the aforementioned optimizations are not useful for the applications we are targeting.

The MIT RAW architecture [19] also uses multiple NoCs. There are two static networks where the routes are specified at compilation time and two dynamic networks, which use packet-switched flow control. However in the case of the RAW architecture the networks are symmetric. CCNoC uses asymmetric sub-networks for different message classes to reduce power consumption.

Balfour et al. [1] propose splitting network traffic into two heterogeneous or homogeneous sub-networks to improve performance and energy efficiency. The heterogeneous architecture uses one sub-network to transport short packets and the other to transport long packets. The homogeneous architecture uses one sub-network to transport packets associated with read transactions and the other with write transactions. Using simple read/write messaging protocols, the authors conclude that the homogeneous architecture achieves better load-balance between the two sub-networks.

In this paper we show that the heterogeneous architecture (i.e., using narrower network) leads to better efficiency than the homogeneous for the case of cache-coherent CMPs. In fact splitting network traffic to read and write transactions is not applicable in a cache-coherent CMP because the cache coherence protocol generates a blend of messages [4]. Splitting messages to request and response sub-networks leads to better load balance[1] than splitting messages to short and long sub-networks. This allows for better occupancy of the two sub-networks, improving the network performance. Not only that, but our CCNoC design does not require virtual channels to prevent protocol-level deadlock, whereas this is a requirement for any design where request and responses travel through the same physical channels. As virtual channels require additional buffers and buffers account for a significant part of the total network power [16], getting rid of the virtual channels leads to further power savings.

3. BACKGROUND

Figure 1 depicts the general anatomy of a cache-coherent tiled CMP chip architecture that we consider in this paper. Each tile contains a processor core with L1 data and instruction caches, a slice of the L2 cache, a slice of the directory and two network interfaces connecting the node to the NoC. The L2 slice can be either a part of a shared L2 cache or a private L2 for the local core. In case of a shared L2, cache blocks are address-interleaved among the L2 slices. In case of private L2s, each core accesses its own L2 tile. While the choice of caching may affect the overall network traffic, it will not directly impact the inherent duality in message types in cache coherence protocols.

---

[1]We define load balance as the ratio of the number of the flits in one network over the other. The closer this ratio is to one, the better load balance we achieve. Across our benchmark suite Request/Response and Short/Long lead to 0.49 and 0.35 respectively.

An invalidation-based directory protocol maintains coherence among the L1 caches in a shared organization. The choice of directory encoding [23] may also affect the overall network traffic but does not fundamentally affect the breakdown of request and response message types. Each tile uses two network interfaces (NIs), one for the core (i.e., L1 controllers) and the other for the L2 slice and the directory controller to allow parallel access to a tile's caches and directories from remote processors. We assume a switch architecture with four ports for the switch-to-switch connections and two ports that are connected to each NI (Figure 1).

We assume a mesh topology as the baseline configuration and a torus topology as the reference system. A torus provides shorter paths than a mesh due to the wrap-around connections, thus performing better when compared to a mesh with the same flit width. However, a torus is implemented in an interleaved manner so that the wire length between all nodes are equal, as such increasing the wire length significantly compared to a mesh. Therefore, a torus consumes more power than a mesh.

3.1 Duality in Coherence Traffic

In the rest of the section, we qualitatively explain how traffic duality arises in cache-coherence protocols. Recent research indicates that the most common forms of on-chip cache traffic are those involving reading and managing cache blocks that are clean [4]. Control messages are short and data messages carry a cache block-sized payload.

In the common case of reading data or instructions, a reader sends a request for the read-only copy of a cache block, followed by a response from the L2 cache with the data. Similarly, to keep the directory up-to-date with sharer information, clean cache block replacements are also notified with small request messages. In the less frequent case of a read from an active writer of a block, the protocol implements a 3-hop transition. The read request is forwarded to the writer, which then responds directly to the reader and the directory with a data message and a notification response respectively. As such, most requests (reads or eviction notifications) for clean blocks are short messages and most responses carry a cache block.

In general, write requests (i.e., fetch block or upgrade requests) are less frequent. Moreover, writebacks in server workloads account for a negligible fraction of the overall traffic because data is rarely updated and instructions are virtually never modified at runtime [4]. Finally, unlike cache block fetches, writebacks are not latency-sensitive and as such do not impact performance directly.

Even among the writes, there are common transitions that fit the duality in traffic model. For example, write misses to blocks that are not actively shared have exactly the same request/response behaviour as reads of clean blocks. Other transitions include upgrade requests to a non-shared block, requiring a short request message and a short response message as well. Among write requests, those involving other readers and consequently a large number of control messages for both requests and responses are quite rare. Parallel workloads in general and server workloads in particular are highly optimized for reuse within L1 caches, as such sharing across
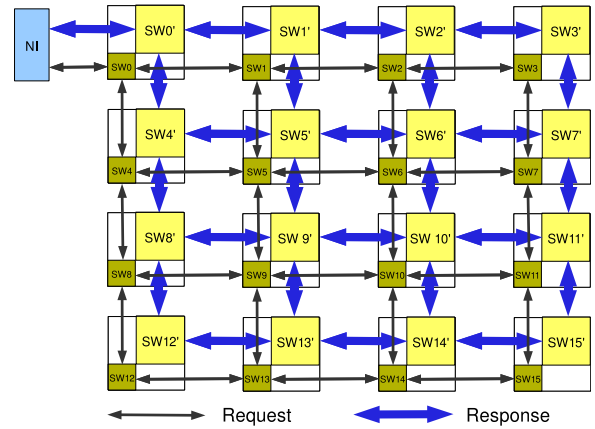

Figure 2: CCNoC with mesh topology.

threads and data migration happen over large windows of time, well beyond L1 residency [4] and writers rarely find blocks shared by others [10].

4. CACHE-COHERENCE NETWORK-ON-CHIP

In this paper, we propose Cache-Coherence Network-on-Chip (CCNoC), a design to capitalize on the duality in cache coherence traffic between requests and responses. We use a pair of asymmetric switches in a dual-switch NoC to optimize for power, area, and performance over a conventional single-switch NoC. Because switch datapath area and power requirements scale faster than wires, using dual sub-networks can help improve wire utilization with narrower switches [1]. Unlike NoCs for simple messaging protocols favouring homogeneous switches, specialization and heterogeneity can help exploit the duality in traffic in cache-coherent CMPs.

Figure 2 depicts the CCNoC with a mesh topology. The routing node consists of two switches, one of them specialized for the request and the other for the response sub-network. Because requests primarily consist of short messages, the request network can be built with switches that have narrower flit width to optimize for power with a minimal impact on the overall performance. The response messages usually carry a cache block and as such need full-sized channel widths.

Splitting the network at the protocol level also obviates the need for virtual channels. To avoid protocol-level cyclic dependencies and deadlocks, conventional NoCs partition the physical resources of every switch among multiple virtual channels to allow independent routing of requests and responses. With requests and responses travelling on separate sub-networks, the coherence protocol no longer needs resource separation with virtual channels and as such the switches can further optimize for power and area with no impact on performance.

In CCNoC, the NI that connects a core to the switch is connected to both the request and response switches in the routing node. The NI pushes requests into the narrow request sub-network and responses in the wide response sub-

CMP Size	16-core for server and scientific workloads
	8-core for multiprogrammed workloads
Processing Cores	UltraSPARC III ISA
	2GHz 8-stage pipeline
	4-wide dispatch / retirement
	32-entry conventional store buffer
	OoO, 96-entry ROB, LSQ
L1 Caches	Split I/D, 64KB 4-way
	2-cycle load-to-use, 3 ports
	32 MSHRs, 16-entry victim cache
L2 NUCA Cache	16-core: 1MB / core, 14-cycle hit latency
	8-core: 2MB / core, 20-cycle hit latency
	16-way set-associative, 64-byte lines
	1 port, 32 MSHRs, 16-entry victim cache
Main Memory	3 GB total memory
	45 ns access latency
Memory Controller	one per 4 cores
	round-robin page interleaving

Table 1: System parameters for 16-core and 8-core CMP.

Figure 3: Traffic distribution

network. To ensure that the cache coherence protocol is not affected, the receiver NI keeps the order between the request and response messages coming from the same source. Such a requirement also holds when message classes are separated using virtual channels.
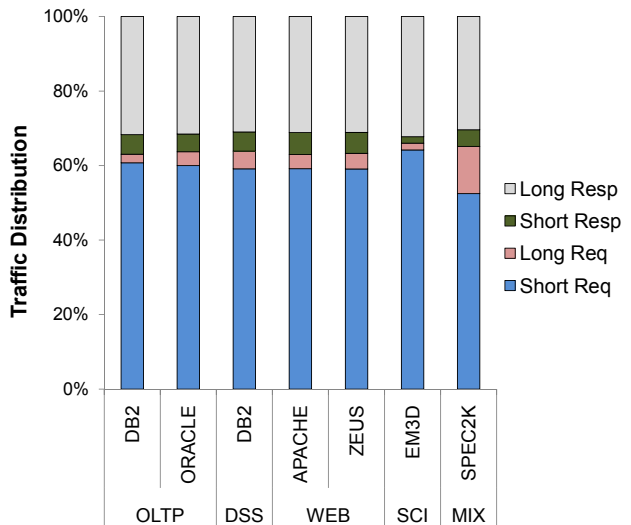
We assume a switch architecture similar to a previously proposed one [18]. The switch uses wormhole flow control which implements the *on/off* transmission protocol [2] and uses only two deep input buffers. Arbitration and channel allocation are considered to be done in one clock cycle and routing is done statically. Other more complex architectures for the switches might be considered, however, that does not change the benefits that CCNoC provides. The dual asymmetric switch configuration of the CCNoC router node holds regardless of the switch architecture, as this is based on the typical traffic patterns of a cache coherent manycore processor.

5. METHODOLOGY

We use FLEXUS [22] for cycle-accurate full-system simulation of a tiled CMP (Figure 1) executing parallel server, scientific and multiprogrammed workloads. FLEXUS extends the Virtutech Simics functional simulator booting Solaris 8, to model timing of all components in a tile. Table 1 summarizes our system architecture.

Server workloads are not likely to benefit from large aggregate on-chip caches beyond their needs to keep instruction working sets [3, 4]. Multiprogrammed desktop workloads, however, are likely to benefit from large aggregate on-chip caches to minimize cache footprint interference among programs. Therefore, we assume a 16-core configuration with 1MB L2 slices for the server and scientific workloads and an 8-core configuration with 2MB L2 slices for the multiprogrammed SPEC CPU2000.

We use the TPC-C v3.0 OLTP benchmark [25] on IBM DB2 v8 ESE and Oracle 10g Enterprise Database Server. We run a mix of queries 1, 6, 13, and 16 from the TPC-H bench-

mark [25] on DB2. Queries 1 and 6 are scan-bound, query 16 is join-bound, and query 13 exhibits hybrid behaviour. To evaluate web server performance, we use SPECweb99 benchmark running on Apache HTTP Server v2.0 and Zeus Web Server v4.3. We use a separate client system to drive the web servers and hence do not include client activity in our measurements. Our multiprogrammed workload consists of SPEC CPU2000 applications (i.e., two copies from each of gcc, twolf, mcf, and art) running the reference input. Finally, we include em3d, a scientific application, as a frame of reference for our server workload results.

We use ORION 2.0 [7] to estimate NoC power consumption and area. We assume that the technology node is 65nm. The clock frequency and the supply voltage of the switches are 2GHz and 1.2V respectively. We use six ports for each switch and we set the depth of each buffer to two flits. We assume that buffers are fabricated using SRAM.

We refer to our baseline single-switch mesh and torus topologies with a channel width of 128 bits as Mesh-128 and Torus respectively. We choose 128 bits, because it gives a suitable area performance ratio for transferring 64-byte cache blocks. Our CCNoC uses dual-tiered switches with 48-bit and 128-bit channels for request and response messages respectively. As a reference point for a higher performance NoC, we also evaluate a mesh with 176-bit channels (i.e., the sum of the two channels width used in CCNoC). We refer to this topology as Mesh-176. We consider a packet to have one or five flits for the 128-bit flit width channel, two or 13 for the 48-bit flit width channel, and one or four for the 176-bit flit width channel.

Area estimations done with ORION 2.0 [7] show that CC-NoC requires only 10% more area when compared to Torus and 41% less area when compared to Mesh-176. Due to the wraparound connections, Torus requires 256 wires in the same direction, whereas CCNoC and Mesh-176 require 176 wires and Mesh-128 requires 128 wires.
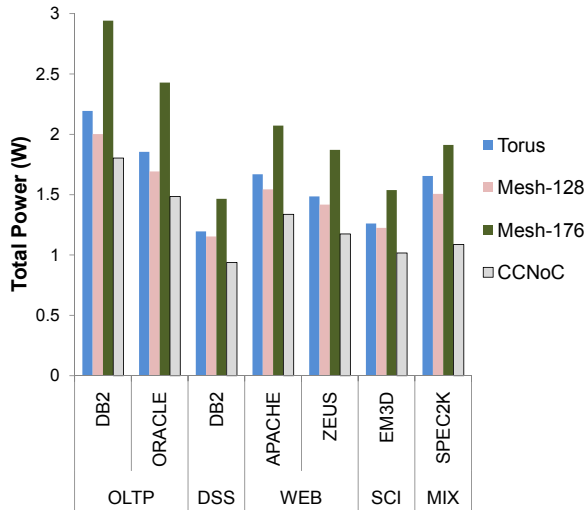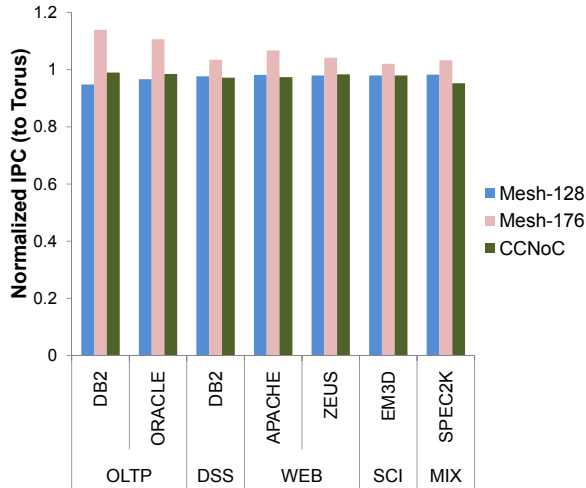
Figure 4: NoC power consumption

Figure 5: Overall performance

6. EVALUATION

We studied our proposed design for both shared and private cache organizations. However, due to space limitations and the fact that the results of the two cache organizations were similar, we present the evaluation of CCNoC design only for the shared cache organization.

6.1 Request/Response Traffic Characterization

Figure 3 shows the distribution of short and long messages across on-chip request and response messages. The request traffic primarily consists of instruction fetch, read cache block, and eviction clean requests (i.e., short messages). Short messages account for 93% of the request messages. In OLTP and Web workloads, both with big instruction footprint, instruction block requests dominate the request traffic. In DSS, the scientific application, and the multiprogrammed workload, all with tiny instruction footprints, read requests are the majority of the request messages. In all workloads except the multiprogrammed workload, the writeback traffic is negligible, implying that the eviction clean requests dominate the

total eviction traffic. The figure indicates that the response traffic is also highly skewed. On average, long response messages account for 86% of the response messages. The majority of long responses are messages carrying instruction or read data (not written) blocks.

Overall, the results corroborate our intuition that the request traffic primarily includes short messages (i.e., control messages), whereas the response traffic primarily consists of long messages (i.e., carrying cache blocks).

6.2 Power & Performance

In this section we evaluate the power consumption of CC-NoC and its impact on performance as well. We use mesh topology for both request and response networks with 48-bit and 128-bit flit widths respectively. We compare our CC-NoC with the Torus, Mesh-128, and Mesh-176 topologies.

Figure 4 shows the network power consumption for all topologies across our benchmark suite. CCNoC has the lowest power consumption across all the network topologies. In Torus, switches have to drive longer wires, thus consuming more power than Mesh-128. Since the crossbar power is dominant when having large flit widths, Mesh-176 consumes more power than Torus as well, and hence it has the highest power consumption. The power savings of CCNoC compared to the network topologies are two-fold. First, CC-NoC does not require virtual channels to break message-level deadlock, because request and response messages go through separate networks. Second, as we said in Section 6.1, the majority of messages are requests and travel through the narrower network, thus requiring less power, and hence the power consumption of CCNoC is smaller than Mesh-128's too. Across all workloads, the results indicate 22%, 16%, and 38% improvement in power over Torus, Mesh-128, and Mesh-176 respectively.

We evaluate the performance of all network topologies by showing the IPC (instructions per cycle) across our benchmark suite in Figure 5. The Torus is considered as the reference topology and hence the IPC of each network topology is normalized to that of Torus. The figure shows that the Mesh-176, due to the bigger flit width, offers the best performance. However, the performance improvement increases the power consumption of the network as well. Torus, due to the wrap-around connections, ranks second in performance. The average loss of CCNoC in performance is only 2% and 8% against Torus and Mesh-176 respectively. In a network under low contention, latency primarily depends on the number of hops, whereas under high contention, latency primarily comes due to waiting for other packets to be serviced. Therefore, the performance of CCNoC is very close to that of Torus for workloads with high traffic (i.e., OLTP, Web), whereas the loss in performance when running low-traffic workloads (i.e., DSS, scientific, and multiprogrammed) is slightly bigger. CCNoC performs similar to Mesh-128.

CCNoC slightly loses against Mesh-128 when running the multiprogrammed workload. In the multiprogrammed workload, as stated in Section 6.1, writeback traffic is high (i.e., long requests such as evictions of dirty blocks). In CCNoC long requests require more flits and consequently need more

cycles to arrive to their destination than the same messages in a 128-bit flit width network topology. In the case of the multiprogrammed workload, this affects the performance, because long requests are a significant portion of the request messages. Because CCNoC is highly optimized for server applications (i.e., flit width of request network is 48 bits), the effect of the writeback traffic in the multiprogrammed workloads becomes more visible.

We calculate the energy-delay^2 product of all network topologies across our benchmark suite. The CCNoC has the lowest energy-delay^2 product among the four network topologies. Particularly, on average CCNoC improves energy-delay^2 product by 18%, 26%, and 17% compared to Torus, Mesh-176, and Mesh-128 respectively. This is because the loss in performance against the rest of topologies is small, whereas the power improvement is significant. Torus and Mesh-176 increase the performance by increasing the power consumption significantly, as explained earlier, whereas the power consumption of Mesh-128 is higher than CCNoC because it constantly powers up the 128-bit crossbars for all messages. On the other hand, CCNoC uses a narrow network to send the majority of short messages and the wider network to send the majority of long messages, thus saving power while achieving similar performance as Mesh-128.

7. CONCLUSION

As supply voltages are levelling off, in future CMOS technology nodes, the key design criteria for chips in general and NoCs in particular will be power. NoCs already account for a significant, if not a dominant fraction of on-chip power for manycore processors. Therefore decreasing the interconnect power consumption is necessary. Interconnect power consumption has to be decreased at the architectural level through specialization. In this paper we present dual-tiered *Cache-Coherence Network-on-Chip (CC-NoC)* for cache coherent manycore server chips. The CC-NoC capitalizes on the duality in on-chip network traffic between requests and responses in cache coherent multiprocessors and optimizes the routing nodes accordingly. CC-NoC uses two asymmetric networks to separate the message classes (request/responses) and reduces the channel width of the request network to save power without impacting performance. Using full system simulation we show that CCNoC, when compared to a torus and a mesh with wider channels, reduces power consumption significantly (on average 22% and 38% respectively) with a minimal performance loss (on average 2% and 8% respectively) .

8. ACKNOWLEDGEMENTS

9. REFERENCES

[1] J. Balfour and W. J. Dally. "Design Tradeoffs for Tiled CMP On-Chip Networks". In Proc. of ICS, June 2006.
[2] W. J. Dally and B. Towles. "Principles and Practices of Interconnection Networks". Morgan Kaufmann, 2004
[3] N. Hardavellas, I. Pandis, R. Johnson, N. Mancheril, A. Ailamaki, and B. Falsafi. "Database servers on chip multiprocessors: limitations and opportunities". In Proc. of CIDR, January 2007.
[4] N. Hardavellas et al. "Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches". In Proc. of ISCA, June 2009.
[5] M. Hayenga, N. E. Jerger, and M. Lipasti. "SCARAB: A Single Cycle Adaptive Routing and Bufferless Network". In Proc. of MICRO, December 2009.
[6] N. E. Jerger, L. Peh, and H. Lipasti. "Circuit-Switched Coherence". In Proc. of NOCS, May 2008.
[7] A.B. Kahng et al. "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration". In Proc. of DATE, April 2009.
[8] J. Kim, W. J. Dally, and D. Abts. "Flattened butterfly: a cost efficient topology for high-radix networks". In Proc. of ISCA, June 2007.
[9] A. Kumar, L. Peh, P. Kundu, and N. K. Jha. "Express virtual channels: towards the ideal interconnection fabric". In Proc. of ISCA, June 2007.
[10] P. Lotfi-Kamran, M. Ferdman, D. Crisan, and B. Falsafi. "TurboTag: Lookup Filtering to Reduce Coherence Directory Power". In Proc. of ISLPED, August 2010.
[11] G. Michelogiannakis, J. Balfour, and W. J. Dally. "Elastic buffer flow control for on-chip networks". In Proc. of HPCA, February 2009.
[12] G. Michelogiannakis, D. Shanchez, W. J. Dally, and C. Kozyrakis. "Evaluating Bufferless Flow Control for On-chip Networks". In Proc. of NOCS, May 2010.
[13] M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol. "An Efficient Dynamically Reconfigurable On-Chip Network Architecture". In Proc. of DAC, June 2010.
[14] M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol. "Low-power and High-Performance On-Chip Communication Using Virtual Point-to-Point Connections". In Proc. of NOCS, May 2009.
[15] M. Modarressi, H. Sarbazi-Azad, and M. Arjomand. "An SDM-Based Hybrid Packet-Circuit-Switched On-Chip Network". In Proc. of DATE, April 2009.
[16] T. Moscibroda and O. Mutlu. "A case for bufferless routing in on-chip networks". In Proc. of ISCA, June 2009.
[17] N. Muralimanohar and R. Balasubramonian. "Interconnect design considerations for large NUCA caches". In Proc. of ISCA, June 2007.
[18] S. Stergiou et al. "×pipesLite: A Synthesis Oriented Design Library for Networks on Chips". In Proc. of DATE, March 2005.
[19] M. Taylor et al. "The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs". IEEE Micro, April 2002.
[20] S. Vangal et al. "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS". In Proc. of ISSCC, February 2007.
[21] H. Wang, L.-S. Peh, and S. Malik. "Power-driven design of router microarchitectures in on-chip networks". In Proc. of ISCA, June 2003.
[22] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. "SimFlex: statistical sampling of computer system simulation". IEEE Micro, Jul-Aug 2006.
[23] J. Zebchuk, V. Srinivasan, M. K. Qureshi, and A. Moshovos. "A Tagless Coherence Directory". In Proc. of MICRO, December 2009.
[24] Semiconductor Industry Association. The International Technology Roadmap for Semiconductors (ITRS). http://www.itrs.net/, 2007 Edition.
[25] TPC Benchmarks. http://www.tpc.org