Circuit-Switched Coherence

‡Natalie Enright Jerger, ‡Mikko Lipasti, and ⋆Li-Shiuan Peh
‡Electrical and Computer Engineering Department, University of Wisconsin-Madison
⋆Department of Electrical Engineering, Princeton University

Abstract—**Circuit-switched networks can significantly lower the communication latency between processor cores, when compared to packet-switched networks, since once circuits are set up, communication latency approaches pure interconnect delay. However, if circuits are not frequently reused, the long set up time and poorer interconnect utilization can hurt overall performance. To combat this problem, we propose a hybrid router design which intermingles packet-switched flits with circuit-switched flits. Additionally, we co-design a prediction-based coherence protocol that leverages the existence of circuits to optimize pair-wise sharing between cores. The protocol allows pair-wise sharers to communicate directly with each other via circuits and drives up circuit reuse. Circuit-switched coherence provides overall system performance improvements of up to 17% with an average improvement of 10% and reduces network latency by up to 30%.**

I. Introduction

As per-chip device counts continue to increase, chip multiprocessors are becoming a reality. Shared buses and dedicated wires do not provide the scalability needed to meet the communication demands of future multi-core architectures. To date, designers have used packet-switched on-chip networks as the communication fabric for many-core chips [10, 16, 17]. While packet switching provides efficient use of link bandwidth by interleaving packets on a single link, it adds higher router latency overhead. Alternatively, circuit-switching trades off poorer link utilization with much lower latency, as data need not go through routing and arbitration once circuits are set up.

For the suite of commercial and scientific workloads evaluated, the network latency of a 4x4 multi-core design can have a high impact on performance (Figure 1) while the bandwidth demands placed on the network are very low. Figure 1 measures the change in overall system performance as the per-hop delay[1] is increased from 1 to 11 cycles. When a new packet is placed on a link, the number of concurrent packets traversing that link is measured (including the new packet)[2]. The average is very close to 1, illustrating very low link contention given our simulation configuration (See Table I). This demonstrates that wide on-chip network channels are significantly underutilized for these workloads, and that overall system performance is sensitive to interconnect latency. An uncontended 5-cycle per-hop router delay in a packet-switched network can lead to 10% degradation in overall system performance. As the per-hop delay increases, either due to deeper router pipelines or network contention, overall system performance can degrade by 20% or more. This latency sensitivity coupled with low link utilization motivates our exploration of circuit-switched fabrics for CMPs.

Manuscript submitted: 19 Feb. 2007. Manuscript accepted: 19 Mar. 2007. Final manuscript received: 29 Mar. 2007

[1]This comprises router pipeline delay and contention.

[2]This measurement corresponds to a frequently-used metric for evaluating topologies, channel load [6].

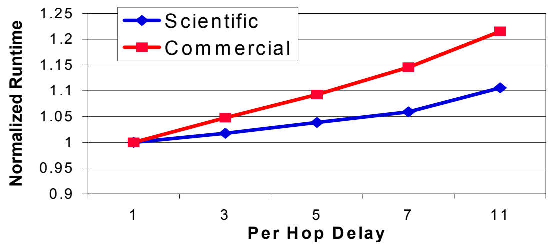


Fig. 1. Effect of interconnect Latency

Our investigations show that traditional circuit-switched networks do not perform well, as circuits are not reused sufficiently to amortize circuit setup delay. This observation motivates a network with a *hybrid router design* that supports both circuit and packet switching with *very fast circuit reconfiguration* (setup/teardown). Our preliminary results show this leading to up to 8% improvement in overall system performance over a packet-switched fabric.

As systems become more tightly coupled in multi-core architectures, co-design of system components becomes increasingly important. In particular, coupling the design of the on-chip network with the design of the coherence protocol can result in a symbiotic relationship providing superior performance. We have found that our workloads exhibit *frequent pair-wise sharing* between cores. Prior work has also shown that processor sharing exhibits temporal locality and is often limited to a small subset of processors [2, 7]. Such application behavior inspires a prediction-based coherence protocol that further drives up the reuse of circuits in our hybrid network and improves overall performance. The protocol predicts the likely sharer for each memory request so the requester can go straight to the sharer for data, via a fast-path circuit, rather than having first to go to the home directory node. Our simulations show this improves overall system performance by up to 17%.

II. Hybrid Circuit-Switched Network

The key design goal of our *hybrid circuit-switched network* is to avoid the circuit setup delay of traditional circuit-switching. Our design consists of two separate mesh networks (the switch design is shown in Figure 2), the main data network and a tiny setup network. The main data network supports two types of traffic: circuit-switched and packet-switched. In the data network, there are C separate physical channels, one for each circuit. To allow for a fair comparison, each of these C channels have 1/C the bandwidth of the baseline packet-switched network in our evaluations.

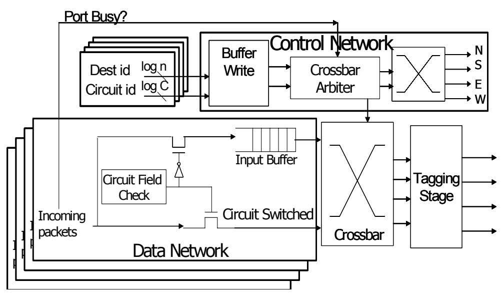


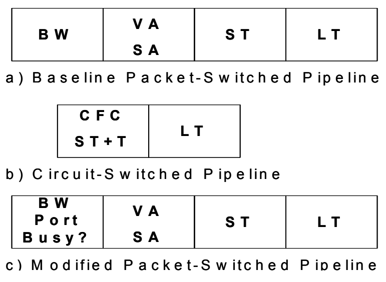
Fig. 2. Switch Design

A. Setup Network

Like in a traditional circuit-switched network, the setup network handles the construction and teardown of circuits and stores the switch configuration for active circuits. However, as our hybrid network does not require waiting for acknowledgment that a circuit has been successfully constructed/torn down, data can be piggy-backed immediately behind the circuit setup request. When a setup request finds that no unused circuits are available, it will trigger a reconfiguration signal so that the LRU circuit will be reconfigured as packet-switched whilst the new circuit request will take over the old circuit. Incoming circuit-switched (CS) flits intended for the old (reconfigured) circuit will henceforth be tagged as packet-switched (PS) flits and will remain packet-switched until they reach their destination. A control flit on the setup network will signal the source of the old circuit, that it must either stop sending CS flits or must re-establish the circuit to prevent buffer overflow due to too many CS flits arriving at a reconfigured node.

The routers in the setup network have three pipeline stages, similar to that of our baseline PS router, except that there is no speculation or virtual channel allocation (VA). This 3-stage pipeline is based on an Intel design [12], which found it not possible to fit the BW into the VA/SA stage and still maintain an aggressive clock of 16FO4s. To lower delay, we assume lookahead routing, thus removing the routing computation from the critical path. Virtual channels are not necessary as traffic on the setup network is low and wormhole is sufficient with lower overhead. When a setup flit arrives, consisting of the destination field (log N, where N is the number of nodes) and the circuit number (log C, where C is the number of physical circuits), it will first be written to an input buffer (BW). Next, it will go through switch arbitration (SA), with each port having a C:1 allocator. This is followed by a circuit reservation on the data network which sets up the data network switch at that hop to route incoming CS flits correctly. The setup flit then traverses the crossbar (ST) and the link (LT) towards the next router in the setup network. In the current design, this network is only six bits wide.

The physical circuit plane is selected at the injection router based on LRU information and the circuit number (log C bits) is stored there for future CS flits injected at that router. A circuit must remain in the same physical circuit plane from source to destination so a log C identifier is sufficient.



BW
VA
SA
ST
LT
a) Baseline Packet-Switched Pipeline
CFC
ST+T
LT
b) Circuit-Switched Pipeline
BW
Port
Busy?
VA
SA
ST
LT
c) Modified Packet-Switched Pipeline

Fig. 3. Router Pipeline [BW: Buffer Write, VA: Virtual Channel Allocation, SA: Switch Allocation, ST: Switch Traversal, LT: Link Traversal, CFC: Circuit Field Check, T: Tagging]

B. Circuit-switched pipeline on data network

The circuit-switched pipeline in our hybrid network is depicted in Figure 3b. To allow CS and PS flits to intermingle throughout the network, we add an extra bit field to each flit indicating if this flit is a circuit- or packet-switched flit. When a flit enters the router pipeline, the circuit field is checked (CFC). If the field is non-zero, this is a CS flit and will proceed through the pipeline in Figure 3b, bypassing directly to ST, which was already set to the appropriate output port when this circuit was originally established. The tagging (T) stage flips the circuit bit for incoming data so that flits originally intended for a circuit will now go into the packet buffers. The tagging stage is enabled only when a reconfiguration is needed at that router; this way, in future hops, since the tagging stage is not enabled, the original CS flits will stay packet-switched until they arrive at the destination.

C. Packet-switched pipeline on data network

If the circuit field is zero, this is a PS flit and will be buffered, proceeding through the packet-switched pipeline shown in Figure 3c. The allocator of the packet-switched pipeline is designed to enable PS flits to *steal* bandwidth from CS flits. It receives a signal from the input ports indicating the presence or absence of incoming flits for the physical circuit C that the PS flit has been assigned to and is being stolen. If there are no incoming flits for that circuit, the PS flit arbitrates for the switch. Once a PS flit wins passage through the crossbar, it then traverses the output port and goes to the next hop. The circuit field remains set to zero, so that this flit will continue to be interpreted as a PS flit and buffered appropriately at the next hop. If a CS flit enters the router while the PS flit is traversing the switch, the CS request will have to be latched until it can proceed along its circuit at the next cycle. To prevent the unlikely scenario where PS flits are starved, a timeout is used to trigger the reconfiguration of a LRU circuit into PS flits, so starved PS flits can make progress.

D. Overhead

Power and area are a first order design constraint when dealing with on-chip networks. Preliminary analysis of our router using Orion [19] shows a setup network router (including the configuration memory) consuming less than 2% of the overall router power in 70 nm technology. On the data network, components of the hybrid CS router that increase power consumption and area are C D/C-wide multiplexers that select from either the circuit or the buffers, tagging hardware to reset the circuit bit in each flit and configuration memory to store circuit paths through the switch. The C D/C-wide crossbars in the hybrid router will occupy less area than the D-wide crossbar of the PS router since the area grows quadratically with the width of the crossbar ports. The setup network will also consume additional area due to the addition of buffers, switch allocator, and wiring. However, as setup flits are very narrow, we do not expect significant area overhead.

III. Coherence Protocol

We couple our hybrid network with a directory coherence protocol based on the Origin directory protocol [13], but augmented with an on-chip directory cache for each core and protocol support for shared interventions; these changes minimize off-chip accesses to memory.

A key disadvantage of directory protocols is the indirection through the directory that is required to request data from another processor. Other proposals that look at using prediction to accelerate the coherence protocol are discussed in Section V. Our protocol extensions streamline this process for load and instruction misses by predicting pairs of processors that frequently share data, and directly requesting data from one cache to another, via a circuit-switched link, without first consulting the directory. Sequential consistency is maintained by ordering all coherence events at the directory, but latency is improved by overlapping the circuit-switched data transfer with the directory access.

The decision to designate a new message as circuit- or packet-switched is made based on the nature of the message. Messages such as invalidation requests from the directory are not indicative of a pair-wise sharing relationship and therefore are injected as packet-switched flits. Read requests will initiate the setup of a circuit if one is not present. All types of requests can reuse an existing circuit between a given source-destination pair.

A. Protocol modifications

To allow directory indirections to be removed, we modified the directory protocol as follows:

1. Allow a cache to request data directly from another cache.
2. Notify the directory that a sharer has been added without having the directory forward the request to the owning cache or initiate a memory request.
3. Allow a cache with a shared block to respond to a shared request.
4. Retry the request to the directory if the direct request fails due to an incorrect prediction or a race condition.

The directory does not need to be aware of which circuit-switched paths are in use so long as it receives notifications of when to add new sharers to the sharing list for each cache line. The above modifications coupled with the fast circuit-switched paths for sharers create new race conditions which have been identified and dealt with in our simulation infrastructure.

B. Sharing Prediction

Circuit-switched paths are set up on demand via a prediction mechanism that predicts the frequency of sharing between two processors. We implement an address region based prediction mechanism. Each processor stores region information alongside its last level cache. When data is sourced from another processor, the region array stores the identity of the sourcing processor. The next time a cache access misses to an address in that region, we predict that same processor will again source the data. Our region array prediction structure is similar to the regions used to determine the necessity of broadcasts in [4]. At the time of the prediction, if no circuit-switched path exists between the requesting and sourcing processors, one is established. If the predicted core cannot provide the cache line requested, it responds to the requesting processor to indicate an incorrect prediction. The requesting core must then retry to the directory. The latency of a mispredicted request is thus the round trip latency on the circuit-switched path plus the indirection latency of the conventional directory request. The prediction array is then updated with the core that actually sourced the data.

IV. Simulation Results

We use a full system multiprocessor simulator [3] built on SIMOS-PPC configured with 16 in-order cores on a 4x4 mesh CMP. Our simulation parameters are given in Table I. Our infrastructure is currently limited to 16 cores; To scale our proposed hybrid circuit-switched router to larger networks, we suggest extending flat meshes with physical express links [5] thereby enriching the connectivity of the underlying network and lowering circuit contention. For our evaluation, we focus on commercial workloads since these present the greatest opportunity, as seen in Figure 1. Preliminary results are presented for the following commercial workloads: TPC-H and TPC-W [18], SPECweb99 and SPECjbb2000 [15]; our current network does not model flit-level contention at the routers but as the observed contention is very low (<6%), we believe it will not affect our findings. However, we do model contention for circuit resources and reconfigure circuits as needed.

TABLE I
Simulation Parameters

Processor	16 in-order cores
Memory System	
L1 I/D Caches (latency)	32KB 2 way set assoc. (2)
Private L2 Caches	1 MB (16 MB total) 8 way assoc. (6)
Memory Latency	200 cycles
Interconnect	
Link Width	PS: 64B, CS: 64B/C (C=2)
Router baseline	3-stage pipeline

A. Evaluation of Hybrid Circuit-Switched Network

Performance comparisons for various interconnection network optimizations are shown in Figure 4 (This shows overall system performance, not just network performance). All cycle counts are normalized to a baseline PS interconnect with a 3-stage router pipeline.

The first bar shows traditional circuit-switching which incurs setup overhead to establish the circuits but is able to reuse dynamic circuit instances. However, there is not sufficient reuse to see performance benefit except in the case of SPECjbb. Traditional circuit switching will drive up the interconnection latency and will therefore have the most harmful impact on those benchmarks which are most sensitive to interconnect latency, namely SPECweb and TPC-H. As a result, we propose a hybrid circuit-switched network. The first transfer in hybrid circuit switching is packet-switched through the network as the circuit is established. Subsequent transfers between the same source-destination pair can reuse the circuit. By overlapping the setup time with the first data packet, our proposed HCS network is able to achieve overall performance improvement over PS of up to 8% and on average 3%.

Performance sensitivity to setup and reconfiguration latency is shown by the bars labeled traditional circuit switching (TCS) and hybrid circuit switching (HCS). The TCS case has an initial setup latency of 8 cycles per hop (4 cycles for setup + 2 cycles for acknowledgment + 2 cycles to send the data) + reconfiguration latency; subsequent data transfers can reuse the established circuit. As described earlier, the setup time per hop for HCS is 4 cycles (the same as packet switching). With each reuse of an established circuit, the average per hop delay is further reduced and asymptotically approaches the ideal interconnect delay of 2 (switch traversal + link traversal). For SPECjbb and TPC-W, the average per hop delay for hybrid circuit switching is 2.9 and 2.6 respectively.

Ongoing results with flit and circuit level contention modeled in the network are shown in Table II. Although con-

TABLE II
Reduction in network latency for HCS relative to PS

SPECjbb	SPECweb	TPC-H	TPC-W
26%	30%	19%	23%

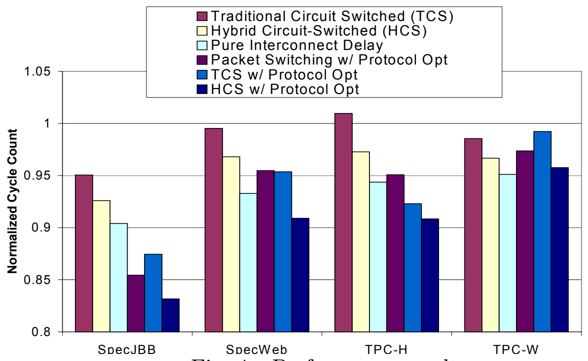


Fig. 4. Performance results

tention reduces the impact of HCS, we still see performance gains over packet switching. Our design reduces network latency by 19-30% with an average reduction of 24% over the packet-switched baseline.

B. *Interactions between Interconnect and Coherence Protocol*

Applying our protocol optimization to each interconnect design point yields results shown by the 3 rightmost bars in Figure 4. The packet switching case results in some performance improvement. However, more improvement can be achieved through the use of circuits, so we apply the protocol optimizations to both traditional circuit switching and our hybrid switching to show the cumulative effect.

SPECjbb sees the most significant performance improvement due to a large percentage (54%) of on-chip misses to clean blocks. Clean misses can be satisfied faster than dirty misses, which will results in more benefit. TPC-H derives most of its benefit as a result of 87% of misses being satisfied on-chip coupled with a very low contribution of stores to the overall miss rate.

V. Related Work

Several hybrid network designs have been proposed. Here, we highlight the differences. SoCBus [20] only packet-switches the configuration message but holds the data at the source until setup is acknowledged. All data in their proposal must be circuit-switched through the network. Wolkotte et. al [21] propose a design that has both circuit-switching and packet-switching; however, it is our understanding that these two networks are physically separate. The packet-switched network is used for reconfiguration and best-effort traffic while the circuit-switched network is used for guaranteed-throughput communications. Wave-switching [8] combines circuit-switching and wave-pipelining but in their design, wormhole-routed and circuit-switched data do not interact and have physically separate resources. Pipeline circuit switching [9] requires that a setup request and acknowledgement message be sent and received before data can travel along the circuit.

We advocate the co-design of the network with a prediction based coherence protocol. Prior work as been done on the ability to predict sharing and communications patterns in shared memory multiprocessors [2, 11]. Work by Acacio et. al [1] also looks at taking the directory access off the critical path for DSM designs. In their work, only lines held in the exclusive or modified state can be accelerated through prediction; our optimization is extended to include shared cache lines; additionally our work co-designs the interconnect which is not a component of their work.

Our baseline packet-switched router is based on a recent aggressive, speculative Intel router design [12] which has a 4-stage pipeline with an added request setup stage over our baseline to accommodate an aggressive 16-FO4 clock cycle. TRIPs supports a single-stage router, partly due to a very lightweight design (4 flit buffers/port, no VCs), and partly as their design requires a control flit ahead of the data to reserve the path which essentially adds a cycle [10]. RAW's dynamic network consists of a 3-stage pipeline. Mullin's Lochnest router is single-cycle, but for a 35-FO4 clock cycle [14]. With an aggressive clock, speculation cannot realistically reduce the pipeline to a single stage.

VI. Conclusions

This work demonstrates the potential of circuit-switched networks for multi-core architectures. Our hybrid circuit-switched network successfully overcomes some of the drawbacks associated with circuit switching, specifically, avoiding setup overhead and reconfiguring circuits *on-the-fly*. Our coherence protocol modifications further drive up circuit reuse and reap higher savings.

In the future, we plan to evaluate our protocol on server consolidation workloads as well as provide more in-depth analysis of the network.

VII. Acknowledgments

This research was supported in part by the National Science Foundation under grants CCR-0133437 and CCF-0429854, the MARCO Gigascale Systems Research Center, as well as grants and equipment donations from IBM and Intel.

References

[1] M. E. Acacio, J. Gonzalez, J. M. Garcia, and J. Duato, "Owner prediction for accelerating cache-to-cache transfer misses in a cc-numa architecture," in *Proc. of ACM/IEEE conf. on SC*, 2002.
[2] E. Bilir, R. Dickson, Y. Hu, M. Plakal, D. Sorin, M. Hill, and D. Wood, "Multicast snooping: A new coherence method using a multicast address network," in *Proc. of ISCA*, May 1999.
[3] H. Cain, K. Lepak, B. Schwarz, and M. H. Lipasti, "Precise and accurate processor simulation," in *Workshop On Computer Architecture Evaluation using Commercial Workloads*, 2002.
[4] J. F. Cantin, M. H. Lipasti, and J. E. Smith, "Improving multiprocessor performance with coarse-grain coherence tracking," in *Proceedings of the 32th ISCA*, 2005.
[5] W. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Trans. on Comp.*, vol. 40, no. 9, 1991.
[6] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann Pub., 2003.
[7] Z. Ding, R. Hoare, A. Jones, D. Li, S. Shao, S. Tung, J. Zheng, and R. Melhem, "Switch design to enable predictive multiplexed switching in multiprocessor networks," in *Proc. of 19th IEEE IPDPS*, 2005.
[8] J. Duato, P. Lopez, F. Silla, and S. Yalamanchili, "A high-performance router architecture for interconnection networks," in *ICCP*, 1996.
[9] P. Gaughan and S. Yalamanchili, "A family of fault tolerant routing protocols for direct multiprocessor networks," *TPDS*, vol. 6, no. 5, May 1995.
[10] P. Gratz, C. Kim, R. McDonald, S. Keckler, and D. Burger, "Implementation and evaluation of on-chip network architectures," in *International Conference on Computer Design*, October 2006.
[11] S. Kaxiras and C. Young, "Coherence communcation prediction in shared-memory multiprocessors," in *Proc. of HPCA-6*, 2000.
[12] P. Kundu, "On-die interconnects for next generation cmps," in *Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems*, December 2006.
[13] J. Laudon and D. Lenoski, "The sgi origin: a ccnuma highly scalable server," in *Proceedings of the 24th ISCA*, 1997.
[14] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. of the 31st ISCA*, 2004.
[15] SPEC, "SPEC benchmarks," http://www.spec.org.
[16] S. Swanson, K. Michelson, A. Schwerin, and M. Oskin, "Wavescalar," in *MICRO-36*, 2003.
[17] M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, "Scalar operand networks: On-chip interconnect for ilp in partitioned architectures," in *Proceedings of HPCA*, February 2003.
[18] TPC, "TPC benchmarks," http://www.tpc.org.
[19] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proceedings of MICRO-35*, 2002.
[20] D. Wiklund and D. Liu, "Socbus: Switched network on chip for hard real time embedded systems," in *Proc. of 17th IEEE IPDPS*, 2003.
[21] P. T. Wolkotte, G. J. Smit, G. K. Rauwerda, and L. T. Smit, "An energy efficient reconfigurable circuit-switched network-on-chip," in *Proceedings of the 19th IEEE IPDPS*, 2005.