Circuit-Switched Coherence

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

Abstract— **Our characterization of a suite of commercial and scientific workloads on a 16-core cache-coherent chip multiprocessor (CMP) shows that overall system performance is sensitive to on-chip communication latency, and can degrade by 20% or more due to long interconnect latencies. On the other hand, communication bandwidth demand is low. These results prompt us to explore circuit-switched networks. 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 setup time can hurt overall performance, as is demonstrated by the poor performance of traditional circuit-switched networks – all applications saw a slowdown rather than a speedup with a traditional circuit-switched network.**

To combat this problem, we propose hybrid circuit switching (HCS), a network design which removes the circuit setup time overhead by intermingling 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 up to 23% savings in network latency which leads to an overall system performance improvement of up to 15%. In short, we show HCS delivering the latency benefits of circuit switching, while sustaining the throughput benefits of packet switching, in a design realizable with low area and power overhead.

I. Introduction

As per-chip device counts continue to increase, many-core chip multiprocessors are becoming a reality. Shared buses and simple rings do not provide the scalability needed to meet the communication demands of these future many-core architectures, while full crossbars are impractical. To date, designers have assumed packet-switched on-chip networks as the communication fabric for many-core chips [11, 23]. 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 relatively low (Workload details can be found in Table V). Figure 1 illustrates the change in overall system performance as the per-hop delay[1] is increased from 1 to 11 processor 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 IV). Wide on-chip network channels are significantly underutilized for these workloads while 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. Looking forward, as applications exhibit more fine grained parallelism and more true sharing, this sensitivity to interconnect latency will become even more pronounced. This latency sensitivity coupled with low link utilization motivates our exploration of circuit-switched fabrics for CMPs.

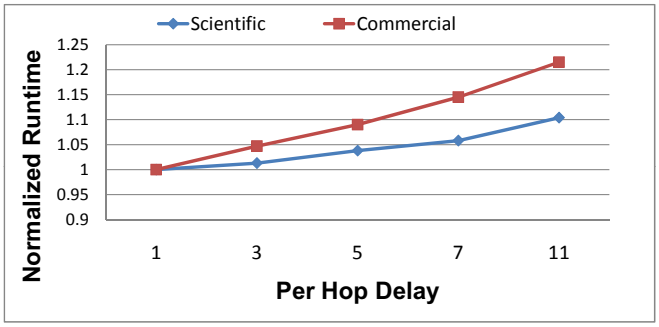

Fig. 1. Effect of interconnect Latency

Our investigations show that traditional circuit-switched networks do not perform well, as circuits are not reused sufficiently to amortize circuit setup delay. As seen in Figure 2, every application saw a *slowdown* when using traditional circuit-switched networks versus an optimized packet-switched interconnect for a 16 in-order core CMP (for machine configuration details see Table IV).

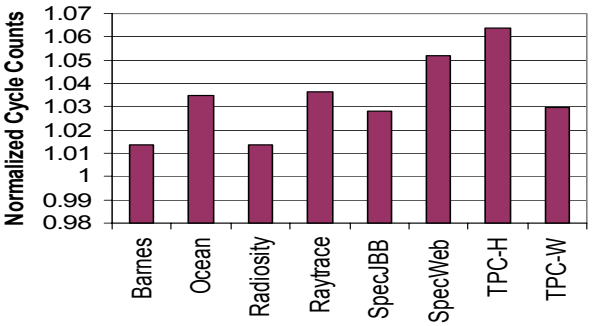

Fig. 2. Performance of Traditional Circuit Switching

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 results show our hybrid circuit-switched network with up to 23% reduction in network latency over an aggressive packet-switched fabric, leading to up to 7% improvement in overall system performance due solely to interconnect

This research was supported in part by the National Science Foundation under grants CCR-0133437, CCF-0429854, CCF-0702272, CNS-0509402, the MARCO Gigascale Systems Research Center, an IBM PhD Fellowship, as well as grants and equipment donations from IBM and Intel.

[1]This comprises router pipeline delay and contention.

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

design. In Section IV, we will demonstrate that while our initial exploration of circuit-switching has been predicated on the low network loads in our applications, our final design works well for high traffic loads as well, demonstrating the effectiveness of our design where packet-switched flits can steal bandwidth when circuits are unused.

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*. 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 (e.g. [3, 7]). Designing a router architecture to take advantage of such sharing patterns can out-perform even a highly optimized packet-switched router.

Figure 3 illustrates the percentage of on-chip misses that can be satisfied by cores that recently shared data with the requester. The numbers on the x-axis indicate the number of sharers maintained in a most recently shared list and the corresponding percentage of on-chip misses that can be captured. For example, with the commercial workloads, the two most recent sharers have a cumulative 65% chance of sourcing data for the next miss. 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 experiencing an indirection to the home directory node. Our simulations show this improves overall system performance by up to 15%.

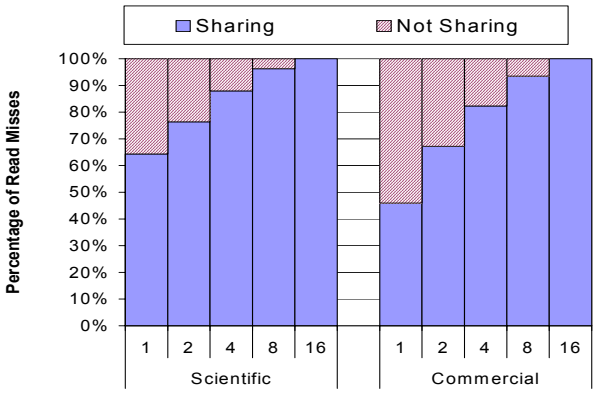

Fig. 3. On-chip misses satisfied by recent sharer(s)

The key contributions of this work are

- Characterizing real-world scientific and commercial workloads on a CMP and showing that on-chip network latency in our configuration is more critical to overall system performance than network bandwidth.
- Proposing a novel router architecture that removes circuit setup delay by interleaving circuit-switched and packet-switched flits on the same physical channels (Section II).
- Co-designing the coherence protocol with the network to further take advantage of low latency circuit-switched paths (Section III).

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. As shown in Figure 4, our design consists of two separate mesh networks: the main data network and a tiny setup network.

The main data network supports two types of traffic: circuit-switched (CS) and packet-switched (PS). 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. A baseline packet-switched network has a channel width of D data bits along with a $logV$ virtual channel ID. Flits on the data network of our hybrid circuit-switched network are D/C wide, plus the virtual channel ID for the packet-switched flits ($logV$) and an additional bit to designate flit type (circuit- or packet-switched). A single setup network is shared by all C circuits.

We compare our design against a baseline packet-switched router shown in Figure 5a and b. Under low loads, the router pipeline is a single cycle (Figure 5a), achieved through aggressive speculation and bypassing. If the input queue is empty and no other virtual channels are requesting the same output channel, the incoming flit can bypass the first two stages and proceed directly to switch traversal. This optimization effectively reduces the pipeline latency to a single stage but only for very low loads (for additional detail see [16]). With the presence of contention due to higher loads, the packet-switched baseline degrades to a three cycle pipeline (Figure 5b). To lower delay, we assume lookahead routing, thus removing the routing computation from the critical path; lookahead routing occurs in the first stage with the buffer write (BW). In the second stage, speculative virtual channel and switch allocation is performed; if speculation fails, virtual channel or switch allocation will be performed again in the next cycle. This single-cycle pipeline is based on an Intel design of a 4-stage pipeline [17], which found it not possible to fit the BW into the VA/SA stage and still maintain an aggressive clock of 16FO4s. Each input contains 4 virtual channels with 4 packet buffers each.

A. Setup Network

Similar to a traditional circuit-switched network, the setup network handles the construction and reconfiguration of circuits and stores the switch configuration for active circuits. However, a packet in our hybrid network does not wait for an acknowledgment that a circuit has been successfully constructed; therefore, data can be piggy-backed immediately behind the circuit setup request. At the injection port, one of the C circuit planes is chosen for the circuit under construction; if there are no unused circuit planes, 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 flits and will remain packet-switched until 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 caused by too many CS flits arriving at a reconfigured node.

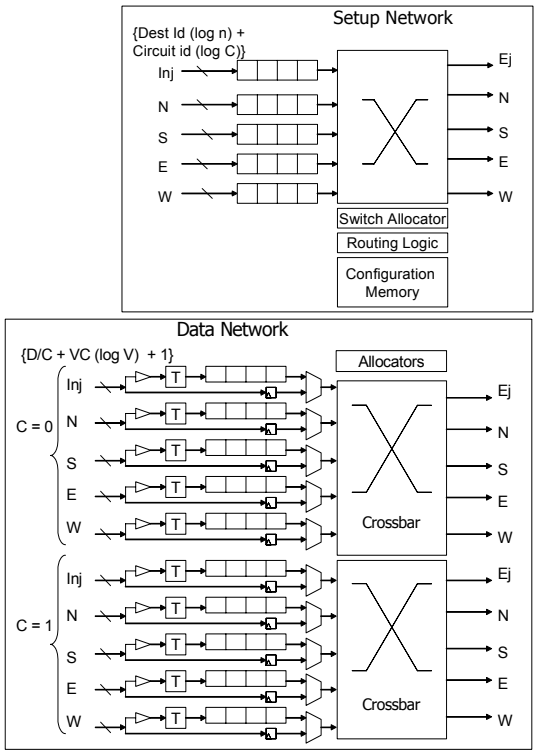

Fig. 4. Proposed router micro-architecture

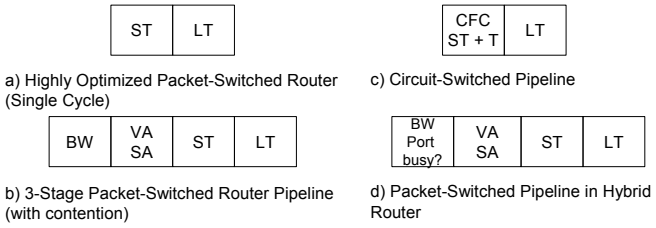

Fig. 5. Router Pipeline [BW: Buffer Write, VA: Virtual Channel Allocation, SA: Switch Allocation, ST: Switch Traversal, LT: Link Traversal, CFC: Circuit Field Check, T: Tagging]

The routers in the setup network have three pipeline stages, similar to that of our baseline packet-switched router, except that there is no speculation or virtual channel allocation (VA). Virtual channels are unnecessary since 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; successful switch arbitration determines the ordering of two setup requests for the same circuit resources. 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.

B. Circuit-switched pipeline on data network

The circuit-switched pipeline in our network is depicted in Figure 5c. To allow circuit- and packet-switched 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 5c, bypassing directly to ST, which was already configured to the appropriate output port when this circuit was originally established. When enabled, 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.

This circuit-switched pipeline is nearly identical to the highly optimized single-cycle packet-switched router in Figure 5a. Note, however, that circuit-switching is able to achieve better performance than a single-cycle packet-switched router under certain sharing characteristics: in the packet-switched baseline, multiple incoming flits prevent bypassing; with circuit-switching, these flits can occupy different circuit planes and proceed simultaneously through the router in a single cycle.

The router architecture with 2 circuit planes is depicted in Figure 4. If the circuit field is set to circuit-switched and the reconfiguration signal has not been asserted from the setup network, than the flit will take the direct path to the crossbar. If the circuit field is set to packet-switched or the circuit has been reconfigured, the flit will take the packet-switched path and will be written into a buffer. Pseudo-code for the operation of the hybrid router design can be found in Table I.

C. Packet-switched pipeline on data network

If the circuit field is zero, this is a packet-switched flit and will be buffered, proceeding through the packet-switched pipeline shown in Figure 5d. The allocator of the packet-switched pipeline is designed to enable packet-switched flits to *steal* bandwidth from CS flits. The packet-switched flit will perform speculative Virtual Channel Allocation and Switch Allocation each subsequent cycle until it is able to steal bandwidth through the switch. It receives a signal from the input ports indicating the presence or absence of incoming flits for the physical circuit C that the packet-switched flit has been assigned to and is being stolen. If there are no incoming flits for that circuit, the packet-switched flit arbitrates for the switch. Once a packet-switched 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 packet-switched flit and buffered appropriately at the next hop. If a CS flit enters the router while the packet-switched flit is traversing the switch, the CS request will have to be latched until it can proceed along its circuit at the next cycle. Only a latch is needed on the circuit-switched path as a CS flit can be stalled at most one cycle by a packet-switched flit traversing the crossbar. To prevent the unlikely scenario where PS flits are starved, a timeout is used to trigger the reconfiguration of the starved

TABLE I
PSEUDO-CODE FOR HYBRID ROUTER DESIGN

Buffer Write Stage	**VA/SA Stage**
if Virtual Channels are not empty **then**	
PS flits are waiting to steal link	**if** Input Virtual Channels are not empty **then**
Proceed to Speculative Virtual Channel & Switch Allocation Stage	Perform Speculative Virtual Channel Allocation
if Input port not idle **then** Incoming flit	And Switch Allocation
if flit type == Packet-Switched **then**	**Switch Traversal Stage**
Write flit to input VC	
else if flit type == Circuit-Switched **then**	**if** Input Idle **then**
if Reconfigured Signal Asserted **then**	**if** Waiting PS flit with successful VA/SA **then**
Write to Input VC	Configure Switch for Stealing
Flip Circuit field to zero (this flit is now a Packet-Switched Flit)	Traverse Switch
else Proceed directory to switch traversal	Restore Circuit Configuration to Switch

plane from circuit-switched to packet-switched so starved PS flits can make progress.

D. Discussion

D.1 Circuit Thrashing

If two different source/destination pairs try to establish a circuit at the same time sharing a common link, these requests will be serialized in the setup network. Effectively, the first request through the setup router will only hold the circuit for a single cycle making reuse impossible. The following cycle, the second request will reconfigure the link and hold that circuit until a subsequent request wishes to claim the link. In the event that these two source/destination pairs continue thrashing by trying to establish a circuit over the same link, their latency will degrade to the baseline packet-switched latency. Circuit thrashing is less likely with more available circuit planes.

D.2 Circuit Fragments due to Reconfiguration

Our reconfiguration mechanism only reclaims the contentious links for a new circuit. The remaining fragments of the old circuit are left intact, thus leaving partial circuits in the network. In the scenario where a link in a circuit has been reconfigured at a hop other the first hop, the source of that circuit is unaware of the reconfiguration and will keep sending circuit-switched flits on the circuit fragment. These circuit-switched flits will be converted to packet-switched flits at the reconfiguration site; this can potentially cause a buffer back-pressure situation back to the source node if there is not enough bandwidth available for the packet-switched flits to steal and continue on to their destination. As such, upon reconfiguration, we send a notification flit on the setup network back to the source of the circuit. The source of the circuit can then choose to re-establish the circuit or send packet-switched flits. If the policy is to always establish a circuit, the source will re-establish its circuit; however, if the policy is to only establish circuits for a limited class of messages, the source will send packet-switched flits until an appropriate message triggers a setup. The trade-offs between these two policies will be explored in Section IV-B.

D.3 Scalability

Given the same traffic load on a larger network, there can potentially be higher circuit contention requiring more circuits. Additional circuits will lead to higher serialization delay. In large systems, circuit switching has the potential for greater latency savings. As the hop count increases from source to destination, the number of cycles saved will also increase. To scale our proposed hybrid circuit-switched network to very large networks, we suggest moving to a more scalable topology. Enriched connectivity will not only lower circuit contention and thrashing, but can potentially reduce global cross-chip latency to just pure wire delay, as circuits can be formed with mostly express links. Future many-core workloads such as server consolidation are unlikely to require global communication [9]. This limited sharing will reduce the need for large numbers of circuits and allow our design to scale further. Unfortunately, our full system simulation infrastructure is currently limited to 16 cores, so we cannot explore this aspect quantitatively.

E. Power and Area Overhead

Power and area are a first order design constraint when dealing with on-chip networks. Using Orion and a 70nm technology [25], a setup network router (including configuration memory) consumes less than 2% of the overall router power. An activity factor of 1 was used to provide a worst case analysis. The configuration memory in the setup network consists of 25 bits for each of the C circuits. For each input port, 5 bits drive the select signal of the crossbar to activate the correct output port. On the data network, components of our router that increase power consumption and area are C D/C-wide multiplexers that select from either the circuit or the buffers, and tagging hardware to reset the circuit bit in each flit.To reduce the area and power consumed by the C D/C multiplexers, we add an additional input to the 4:1 multiplexers in the baseline router that select between the VCs. Replacing 4:1 multiplexers with 5:1 multiplexers increases the power consumption of the router less than 1%. Power can potentially be lowered further by reducing the buffers and VCs in the hybrid router; however we chose to stick with the same configuration as the packet-switched baseline for a consistent comparison.

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 (each crossbar has the same number of ports as the baseline, i.e. 5). The setup network also consumes additional area due to the addition of buffers, switch allocator, and wiring. However, as the setup network is very narrow, we do not expect significant area overhead.

In the data network, the buffers are distributed evenly among the multiple circuit planes with the same total buffer capacity the packet-switched baseline. Since the crossbar ports are 1/2 and 1/4 as wide for 2 and 4 circuit planes respectively, the area of each crossbar is reduced quadrati-

cally, giving our design extra area for the switch configuration memory. The setup network, including configuration memory and 4 buffers per input port consumes only 0.2% of the baseline packet-switched router area. In short, we do not impose any additional overhead over the baseline packet-switched router and less than 3% increase in router power.

III. Coherence Protocol

We couple our hybrid network with a directory coherence protocol based on the Origin 2000 [18], but augmented with a distributed directory cache that is stored along-side the shared last level (L3) cache bank at each node.

While directory-based protocols are more scalable than broadcast-based ones, a key performance disadvantage of directory protocols is the indirection through the directory that is required to request data from another processor.

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. Other proposals that look at using prediction to accelerate the coherence protocol are discussed in Section V.

A. Protocol modifications

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

- Allow a cache to send a request directly to another cache.
- 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.
- 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 active as long as it is notified to add a new sharer to the sharing list for each cache line. The protocol implementation is decoupled from changes to the interconnection network. 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 (see Section III-C).

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 information about sharers of an address region alongside its last level cache. When data is sourced from another processor, the predictor stores the identity of the sourcing processor for the address region of the data. 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 prediction structure is similar to the regions used to determine the necessity of broadcasts in [5], but is easier to maintain since it is not required to be correct. 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 from cache to cache on the interconnect plus the indirection latency of the conventional directory request. The prediction array is then updated with the core that actually sourced the data. An example of the protocol and prediction mechanism is given in Table II.

TABLE II
Prediction and Protocol Walk-through Example

1. Processor 1 misses to Address A
First miss to A → No Prediction Available
2. Send Request to Directory
3. Directory forwards request to Processor 4
4. Processor 4 responds with data
5. Processor 1 receives data and stores Pred(Region A) = 4
6. Processor 1 misses to A+8 → Predicts Processor 4
Send request to Processor 4
Notify Directory
7. Processor 4 receives request
Prediction is correct → responds with data
8. Directory adds Processor 1 to sharing list
9. Proc. 1 receives data from Proc. 4 and ACK from directory

In the example in Table II, if Processor 1's prediction is incorrect, the directory will have already added Processor 1 to the sharing list due to the decoupling of the data request from the ordering request. The directory protocol supports silent evictions of shared lines, so having Processor 1 added as a sharer will not result in any incorrect coherence actions; the sharing list must include all processors caching that block but can contain additional processors at the expense of more invalidation messages should an upgrade occur. Processor 1 will retry its request to the directory once Processor 4 has acknowledged its incorrect prediction.

C. Verification of Consistency

C.1 Protocol Invariants

Validation of correctness is an important yet difficult issue when developing or modifying a coherence protocol. To simplify verification, we built our modifications on top of an existing protocol [18]. In both the baseline and the modified protocol, the directory serves at the sole ordering point for requests. Our modifications decouple the data request message from the ordering message. The data response may be received prior to an acknowledgment from the directory that the request has been properly ordered, but the requesting processor cannot consume the data without that acknowledgment. To illuminate the protocol changes, as well as demonstrate that ordering properties have been preserved, we include pseudo-code for a read request in both the baseline and modified protocols in Table III.

C.2 Simulation Verification

In addition to preserving the directory as the sole ordering point for all coherence requests, we verify that sequential consistency is maintained through our simulation environment. Our simulation infrastructure includes tools to verify sequential consistency and execution correctness. This tool executes two side-by-side execution-driven full-system simulations: the first simulation is the application executing on a modified version of our integrated functional and timing simulator, which produces a memory reference

TABLE III
PSEUDO-CODE COMPARISON OF BASELINE PROTOCOL TO PREDICTION PROTOCOL

Baseline Protocol - Read Request (RdReq)	**Prediction Protocol - Read Request (RdReq)**
1. RdReq goes across network to home directory cache	1a. RdReq goes across network to predicted sharer 1b. Directory update goes across network to home node
Directory Actions	**Decoupled Directory Actions (cont. from 1b)**

2. Directory cache performs look up
if Directory state is Unowned **then**
initiate memory request
When memory request completes send data to requester
if Directory state is Exclusive **then**
Transition to Busy
Send intervention to owner
Add requester to sharing list
Transition to Shared when cache response is received
if Directory state is Shared **then**
Forward request to owner
Add sharer to sharing list
if Directory state is Busy **then** Send NACK to requester
	2. Directory cache performs lookup
	if Directory state is Unowned **then**
	Prediction is incorrect → initiate memory request
	When memory request completes send data to requester
	if Directory state is Exclusive **then**
	if Prediction is correct **then**
	Transition to Busy (wait for ack from predicted node)
	Send ACK to requester
	else Prediction is incorrect
	Transition to Busy & send intervention to owner
	Transition to Shared when cache response received
	Add requester to sharing list
	if Directory state is Shared **then**
	if Prediction is correct **then**
	Add sharer to sharing list & send ACK to requester
	if Prediction is incorrect **then**
	Forward request to owner & send NACK to requester
	if Directory state is Busy **then** Send NACK to requester

Cache Response to Directory request	**Decoupled Cache Actions (cont. from 1a)**
3. Owning cache receives intervention Send data to requester & transition to Owned Send ACK to directory	3. Owning cache receive predicted request **if** Prediction is correct **then** Send Data to Requester **if** Block is Exclusive or Modified **then** Send Ack to directory **else if** Prediction is incorrect **then** Send NACK to Requester
Requesting Node	**Requesting Node**
4. Receives Data from either Directory or Owning Cache Transitions to Shared & forward data to L1 RdReq Complete	4. **if** Data received from Predicted Cache **then** **if** Directory NACK received **then** Discard data (stall copy) & wait for valid response **else if** Directory ACK received **then** Forward Data to L1 & RdReq Complete **else if** NACK received from Predicted Cache **then** Re-initiate RdReq without prediction

stream that is used to drive the second simulation which uses an unmodified, known-to-be-correct simple functional simulator. This memory reference stream contains monotonically increasing memory version numbers that are used by the second simulator to reconstruct a sequentially consistent order. If no sequentially consistent order can be found then we know that our protocol modifications have violated our consistency model and an error is flagged. Finally, all updates to architected state performed by both simulators are compared instruction by instruction, and any discrepancies flag errors and aid in debugging the protocol. We have simulated billions and billions of instructions on top of this checking infrastructure without incurring errors. This simulation time coupled with the preservation of ordering through the directory, leads us to safely conclude that our protocol modifications and our hybrid circuit-switched interconnect are substantially correct.

IV. Simulation Results

We use a full system multiprocessor simulator [4] built on SIMOS-PPC configured with 16 in-order cores on a 4x4 mesh CMP. Included in our simulation infrastructure is a cycle-accurate network model including pipelined routers, buffers, virtual channels and allocators. Hybrid Circuit-Switched modifications such as circuit stealing are also faithfully modeled. Our simulation parameters are given in Table IV. Results are presented for the following commercial workloads: TPC-H and TPC-W [24], SPECweb99 and SPECjbb2000 [22] and several Splash2 workloads [28]. Details for each workload are presented in Table V. We use statistical simulation as described by [2].

TABLE IV
SIMULATION PARAMETERS

Cores	16 in-order
Memory System	
L1 I/D Caches (lat)	32 KB 2 way set assoc. (1 cycle)
Private L2 Caches	512 KB (8 MB total 4 way set assoc. (6 cycles), 64 Byte lines
Shared L3 Cache	16 MB (1MB bank at each tile) 4 way set associate (12 cycles)
Memory Latency	100 cycles
Interconnect	
Link Width	Packet Switching 64 Bytes Circuit Switching 64B/C (C=2,4)
Link Latency	1 cycle
Optimized PS Router Baseline	1-3 stages, 4 Virtual Channels w/ 4 Buffers each
Setup Network	Wormhole with 4 Buffers

A. Evaluation of Hybrid Circuit-Switched Network

One of the primary goals of this work is to reduce the interconnect latency by removing the router overhead. Our hybrid circuit-switched network can reduce interconnect latency by as much as 23% as shown in Figure 6. We measure the average interconnect latency for 2 and 4 circuit planes as compared to an optimized baseline packet-switched interconnect. Average packet delay is calculated as the time between the injection of the head flit and when the critical word arrives at the destination. As the total link bandwidth is kept constant at 64 bytes, the 2 and 4 circuit planes will have 32 and 16 bytes respectively per plane; packets will consist of 2 and 4 flits respectively. The four circuit plane

TABLE V
BENCHMARK DESCRIPTIONS

Bench.	Description
SPECjbb	Standard java server workload utilizing 24 warehouses, executing 200 requests
SPECweb	Zeus Web Server 3.3.7 servicing 300 HTTP req.
TPC-W	TPC's Web e-commerce benchmark, DB Tier Browsing mix, 40 web transactions
TPC-H	TPC's DSS Benchmark, IBM DB2 v6.1 running query 12 w/ 512MB DB, 1GB Mem
Barnes	8K particles, full end-to-end run i.e. init
Ocean	514x514 full end-to-end run (parallel phase only)
Radiosity	-room -batch -ae 5000 -en .050 -bf .10 (parallel only)
Raytrace	car input (parallel phase only)

configuration gives us additional benefit as circuits can be maintained longer between reconfigurations and see more reuse, thus reaping more benefit. To combat the increased serialization delay that moving to 4 circuits would cause, we send the critical word first through the network.

In addition to comparing HCS against PS, we also present results for Narrow Packet Switching (NPS); in NPS we partition a PS network into 4 narrower networks (similar to the 4 narrow circuit planes in HCS). Each narrow network in NPS has 2 virtual channels with 2 buffers per VC. When a packet is injected into the network, it selects an NPS network in a round robin fashion and continues on the same NPS network until reaching its destination. NPS provides modest improvements over PS but underperforms when compared to HCS.

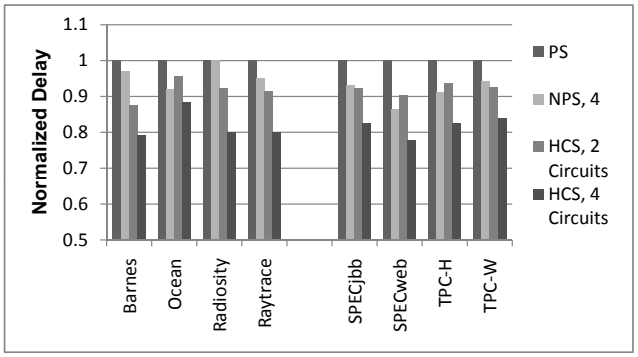

Fig. 6. Reduction in Interconnect Latency due to HCS

Circuit-switched connections are designed to enable fast connections between frequently sharing cores; however, we do not want to sacrifice the performance of messages that do not involve frequent sharing. Our design is able to circuit-switch 18 to 44% of all flits with an average latency of 4.3 cycles. The remaining 56 to 82% of flits that are packet-switched or partially circuit-switched through the network still achieve a reasonable average interconnect latency of 7.9 cycles. Figure 7 gives the contribution of circuit-switched flits and non-circuit-switched flits to overall average network latency. Non-circuit-switched flits encompasses both packet-switched flits and reconfigured circuit-switched flits. The x-axis shows the percentage of overall network messages that are either circuit-switched or not. With 4 circuits, we see a larger contribution of purely circuit-switched flits; this is expected as reconfigurations are less frequent. This larger contribution of circuit-switched flits causes the overall average network latency to be lower. Policies that take further advantage of circuit-switched links and reduce unnecessary reconfigurations can further reduce interconnect latency.

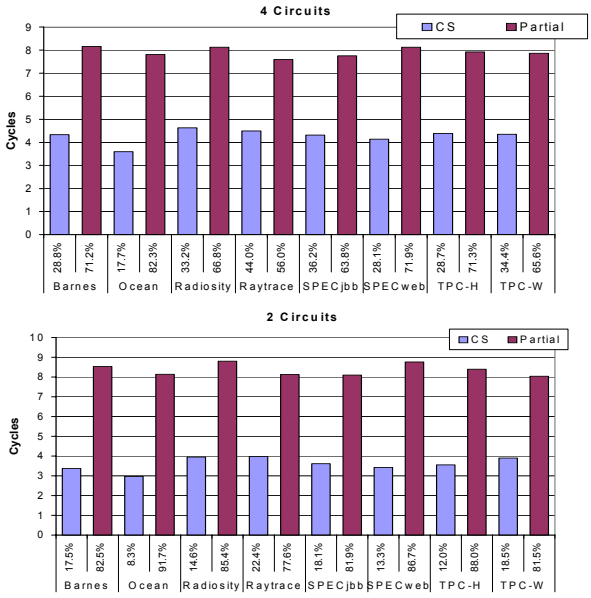

Fig. 7. Latency Breakdown of Circuit-Switched Flits and Packet-Switched/Partial Circuit-Switched Flits

Figure 8 shows the overall system performance for hybrid circuit-switching with 2 and 4 circuit planes and narrow packet-switching with 4 narrow networks, all normalized to the baseline PS network. Moving from 2 circuits to 4 circuits shows an average of 3% reduction in execution time (up to 7%) for commercial workloads. Frequent reconfiguration prevents TPC-H from seeing any benefit from 2 circuits. When characterizing the sharing patterns in TPC-H, 4 recent sharers are needed to satisfy 64% of on-chip misses in contrast to only 2 for the other commercial workloads (Figure 3). Ocean does not benefit from circuits since most misses go off-chip causing the miss latency to be dominated by the memory access time. The other scientific workloads see little benefit from hybrid circuit-switching due to low overall miss rates and low numbers of coherence misses. Increasing the number of circuits further would likely yield little benefit due to the increase in flits/packet. For most workloads, little improvement is gained from the NPS configuration; NPS does not provide the added performance improvement for pair-wise sharing that HCS targets.

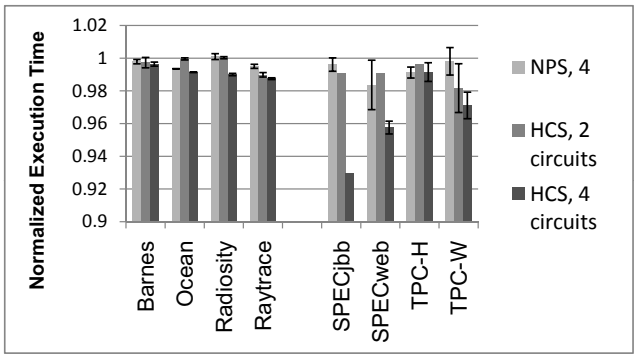

Fig. 8. Overall Performance of HCS

B. Circuit Setup Policies

Two different policies regarding the setup of new circuits are explored. In the first policy, the decision to allow a new message to construct a circuit-switched path (if one is not already present) is made based on the nature of the message. Invalidation requests from the directory are not indicative of a pairwise sharing relationship and therefore are injected as packet-switched flits. Read requests and data transfers will setup a circuit if one is not present. All types of requests can reuse an existing circuit between given source-destination pairs. This policy is contrasted with a policy that allows all types of messages to construct a new circuit in Figure 9 for 2 circuits. Limiting the construction of new circuits to certain message classes causes a loss in opportunity as noted by a average increase in network latency of 3% with a maximum increase of 7%. Up to 10% more flits take advantage of circuit-switched paths when new circuits are always constructed. The limited setup policy does drive up reuse by 30%. Additional policies to simultaneously drive up reuse while maximizing utilization of circuits may further lower interconnect latency.

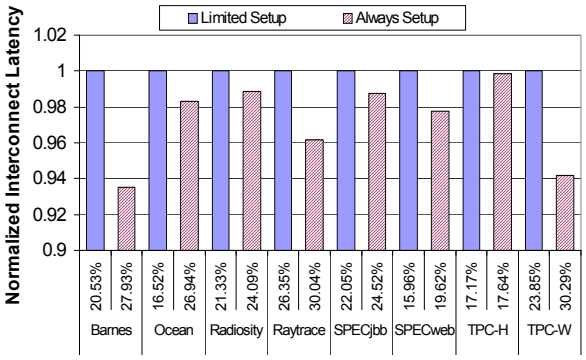

Fig. 9. Impact on Network Latency of Restricting Circuit Setup to Certain Classes of Messages

Limiting the setup of circuits increases the percentage of packet-switched flits and reduces the average time interval between instances of link stealing at each input port. Scientific workloads see longer intervals between link stealing than commercial workloads due to the lower communication demands in the former.

C. Packet-Switched Flit Starvation

As described in Section II, to prevent the unlikely scenario of packet-switched flit starvation, a timeout mechanism reconfigures the circuit starving the packet-switched flits to allow packet-switched flits to make forward progress. To characterize the potential for starvation, we measure the waiting time of packet-switched flits (with the timeout mechanism disabled). The percentage of all PS flits that must wait at least 1 cycle to steal bandwidth from a circuit, in all cases is less than 1%. Average wait time for this small number of flits is less than 7 cycles across all workloads. The average wait time increases slightly when the number of circuit planes is increased from 2 to 4; this is explained by higher circuit utilization.

D. Setup Network Evaluation

In Section II we assert that a wormhole network is sufficient for the setup network. Figure 10 supports this claim. Utilization of the setup network is particularly low when circuit construction is limited to a subset of messages. The data in Figure 10 is measured as the average number of flits waiting at a router when a new flit arrives (including the new flit). The always setup policy drives up the load on the setup network but it is still at an acceptable level for wormhole switching. The average delay through the setup network is 10.5 cycles which reflects the very low utilization of the setup network. The setup network does not need significant buffering resources since utilization is low coupled with the small setup flit size (6-8 bits).

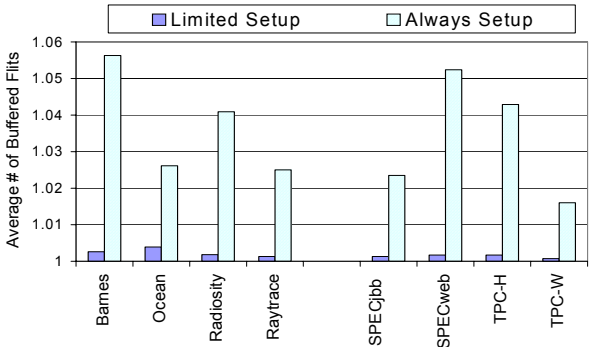

Fig. 10. Channel Load of Setup Network

E. Interactions between Network and Coherence Protocol

Overall system performance when HCS is combined with the protocol optimization is shown in Figure 11. Since the protocol optimizations are largely independent from the interconnect design, we compare against the baseline packet-switched network plus protocol optimizations with results normalized to the packet-switched baseline. The HCS results are given for 4 circuit planes and the always setup policy. We see up to 15% improvement in overall system performance with an average improvement of 12% for commercial workloads and 4% for scientific workloads. The commercial workloads see greater benefit due to their larger miss rates and greater sensitivity to miss latency. TPC-H derives most of its benefit as a result of 82% of misses being satisfied on-chip coupled with a very low contribution of store misses to the overall miss rate. As our protocol optimization only targets load and instruction misses, TPC-H will see more benefit than those benchmarks that have a larger contribution of store misses. Ocean performance is dominated by off-chip misses; the injection of additional traffic due to the protocol optimization further delays these misses resulting in a slight performance degradation.

Comparing Figures 8 and 11, we see that in many cases, the improvement for HCS with the protocol optimization is greater than the sum of the HCS alone and the protocol optimization with packet switching. Given our co-design of both the coherence protocol and the interconnect, hybrid circuit-switching has more opportunities for reuse when the protocol optimization is added, resulting in superior performance; circuit reuse increased by up to 64% when HCS is combined with protocol optimizations. Protocol optimizations reduce coherence miss latency by 50%; with hybrid circuit switching, those miss latencies are further reduced by 10%.

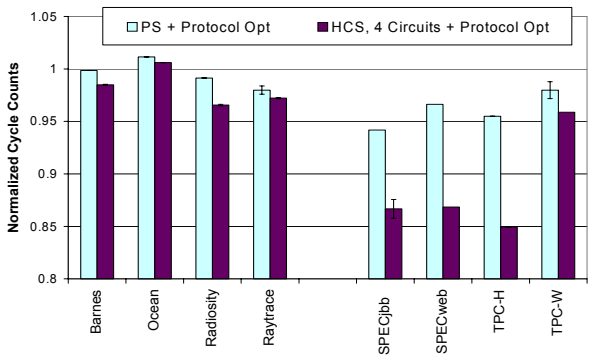

Fig. 11. Performance of Protocol Opt. + Hybrid Circuit Switching

F. HCS with Synthetic Traffic

In addition to performing full-system simulation with commercial and scientific workloads, we further explore the performance of HCS through synthetic traffic patterns in Figures 12 and 13. Both figures show average interconnect latency in cycles across all injected packets. All simulations run for 1 million cycles with increasing injection rates (or channel load) as a percentage of maximum link capacity along the x-axis. With uniform random traffic in Figure 12, we reduce latency by 10-15% over packet switching across all loads prior to saturation, despite the fact that there is hardly any reuse of circuits due to the randomized traffic. HCS has lower latency at low to moderate loads primarily because circuits are always given priority on their first use–i.e. the first packets on circuits always zoom through–whereas in packet-switching, bypassing within the router will not be possible if there is more than one flit in the entire router. At very low utilization circuit-switching outperforms our PS baseline since we piggyback data flits long with the setup flits which shaves one cycle off the serialization latency. Figure 12 also shows network latency results for Narrow Packet Switching (NPS).At moderately low loads, approximately 40% more incoming packets are able to bypass directly to the crossbar with HCS than with NPS or PS. When the load reaches roughly 30%, HCS and PS bypass similar number of packets resulting in similar network saturation points. This attests to the effectiveness of packet-switched flits in HCS stealing idle circuit bandwidth. At moderate loads (20-40%), NPS is able to bypass 5-18% more packets than PS, delivering higher saturation throughput. This matches intuition since NPS trades off serialization delay with bandwidth.

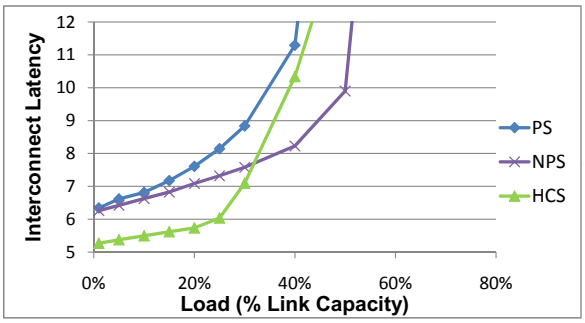

Fig. 12. Performance of HCS with Uniform Random Traffic

In Figure 13, we simulate the performance of HCS under permutation traffic, where each node communicates with one other node. Since HCS specifically targets pair-wise sharing, we would expect this type of traffic to benefit from circuit reuse and perform very well. At low to moderate loads HCS improves network latency by 20% over PS, and saturates at higher utilization. NPS performs slightly better than PS at very low utilization. As the load increase, NPS performs considerably better than PS (10-15% lower latency) since the load is distributed across multiple narrow network allowing speculation to be more effective. With round-robin placement of packets on NPS networks, those networks eventually saturate at a similar load to PS.

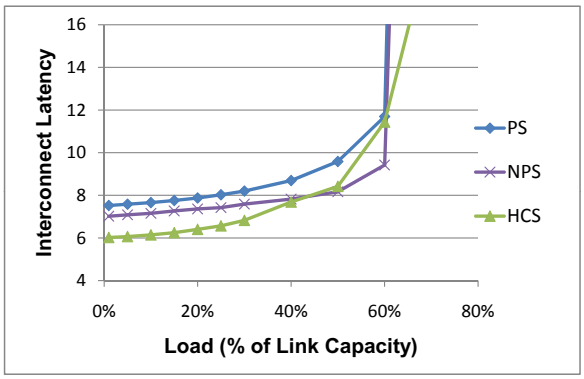

Fig. 13. Performance of HCS with Permutation Traffic

To reiterate, Figure 12 represents worst-case circuit reuse behavior (we observe much higher reuse for real applications), so we are not surprised that HCS saturates somewhat earlier than the NPS case, and are pleased with its robust latency reductions at low utilization. In contrast, Figure 13 demonstrates the robustness of HCS over both PS and NPS under all traffic loads when there is significant circuit reuse.

V. Related Work

Our work proposes a hybrid circuit-switched router that interleaves circuit- and packet-switched flits on the same physical network with low area and power overhead. Network design for CMPs is an area of significant research effort. Several other hybrid network designs have been proposed in the literature. Here, we highlight some of the key differences between those proposals and our work. SoCBus [26] 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 [27] propose a design that has both circuit- 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 [10] requires that a setup and acknowledgment message be sent and received before data can use the circuit. HCS differs from these proposals since it obviates the need for setup and acknowledgment messages to be sent and received prior to the data transfer. Additionally, in HCS, PS and CS flits share the same physical network.

Hybrid circuit-switching is able to remove the buffer write and virtual channel/switch allocation stages for up to 44% of flits. Other recent work [15] also successfully removes a significant portion of this overhead through Express Virtual Channels (EVCs). EVCs create express virtual paths that bypass nodes for a given number of hops allowing them to reduce both delay and energy consumption in the network. In contrast to this work, which specifically tries to optimize for sharing patterns in conjunction with the coherence protocol, EVCs provides a more general framework to accelerate messages. To achieve maximum benefit, EVCs are limited to a small number of hops; our network will show increasing gains as the hop count goes up under low loads. Flit reservation flow control [20] avoids router overhead by sending a control flit to reserve network resources ahead of the data flits; however this design suffers from increased router complexity and high overhead. Another hybrid network [21] combines a bus architecture with a switched network; the bus allows processing elements with communication affinity to transfer data quickly without high overhead while the switched network provides scalability. Our design provides greater flexibility as sharing cores do not need to be physically close to experience low latency communication.

Topologies such as a flattened butterfly [14] can be used to reduce the number of router traversals; however, this design suffers from the use of long global wires. HCS is able to bypass routers without the drawback of long global wiring.

We advocate the co-design of the network with a prediction based coherence protocol. Prior work looks at predicting sharing and communications patterns in shared memory multiprocessors [3, 13]. 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 more aggressive than a recent Intel router design [17] which has a 4-stage pipeline to accommodate an aggressive 16-FO4 clock cycle; At low loads, it has a single-stage pipeline. Recent routers have aggressively pursued a single-cycle pipeline, but only at low loads: TRIPs uses lookaheads and bypassing to realize a single-stage router [11], while Mullin's Lochnest router uses aggressive speculation to shorten the pipeline to a single cycle at 35-FO4 clock cycle [19]. RAW's dynamic network consists of a 3-stage pipeline which resembles our 3-stage non-optimized baseline pipeline. Hybrid circuit-switching can be seen as another technique to further shorten the router pipeline in the presence of certain sharing patterns; the performance of hybrid circuit-switching is enhanced through protocol optimizations. The same is not possible for aggressive packet-switched routers.

VI. Conclusions

This work demonstrates the potential of circuit-switched networks for multi-core architectures. Hybrid circuit-switching is able to reduce network latency by up to 23% and improve overall performance by up to 7%; when combined with our protocol optimizations overall performance improves by up to 15%. HCS achieves these performance gains over a highly optimized baseline packet-switched router with single-cycle delay for low-loads.Our HCS network successfully overcomes some of the drawbacks associated with circuit switching, specifically: avoiding setup overhead, reconfiguring circuits *on-the-fly*, and interleaving circuit- and packet-switched flits on the same physical resources. The co-designed coherence protocol drives up circuit reuse and reaps better performance than the sum of the benefit from the circuit-switched interconnect and the protocol modifications, when applied separately.

Looking forward: while current applications exhibit fairly coarse-grained parallelism, future applications are likely to have more fine-grained parallelism [12] resulting in more frequent sharing and greater latency sensitivity. This increase in sharing will heighten the need for a fast interconnect of the type that our hybrid circuit-switching provides. Server consolidation workloads, an emerging class of applications for CMPs, will exhibit limited sharing and see greater benefits from our hybrid circuit-switching by allowing circuit-switched pairs to persist longer and reap even greater benefit [9].

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 conf. on SC*, 2002.

[2] A. R. Alameldeen and D. A. Wood, "Variability in architectural simulations of multi-threaded workloads," in *HPCA*, 2003.

[3] E. Bilir et al, "Multicast snooping: A new coherence method using a multicast address network," in *Proc. of ISCA*, May 1999.

[4] H. Cain, K. Lepak, B. Schwarz, and M. H. Lipasti, "Precise and accurate processor simulation," in *CAECW*, 2002.

[5] J. F. Cantin et al, "Improving multiprocessor performance with coarse-grain coherence tracking," in *Proc. of the 32th ISCA*, 2005.

[6] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. SF, CA: Morgan Kaufmann Pub., 2003.

[7] Z. Ding et. al, "Switch design to enable predictive multiplexed switching in multiprocessor networks," in *IPDPS*, 2005.

[8] J. Duato et. al, "A high-performance router architecture for interconnection networks," in *ICCP*, 1996.

[9] N. Enright Jerger et. al, "An evaluation of server consolidation workloads for multi-core designs," in *Proc of IISWC*, 2007.

[10] P. Gaughan and S. Yalamanchili, "A family of fault tolerant routing protocols for direct multiprocessor networks," *TPDS*, vol. 6, no. 5, May 1995.

[11] P. Gratz et al, "Implementation and evaluation of on-chip network architectures," in *ICCD*, 2006.

[12] C. J. Hughes et. al, "Physical simulation for animation and visual effects: Parallelization and characterization for chip multiprocessors," in *ISCA*, 2007.

[13] S. Kaxiras and C. Young, "Coherence communcation prediction in shared-memory multiprocessors," in *Proc. of HPCA-6*, 2000.

[14] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *MICRO-40*, 2007.

[15] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Toward the ideal interconnection fabric," in *In Proceedings of ISCA-34*, 2007.

[16] A. Kumar et al, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *ICCD*, 2007.

[17] P. Kundu, "On-die interconnects for next generation CMPs," in *Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems*, December 2006.

[18] J. Laudon and D. Lenoski, "The SGI Origin: a ccNUMA highly scalable server," in *Proceedings of the 24th ISCA*, 1997.

[19] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *ISCA*, 2004.

[20] L.-S. Peh and W. J. Dally, "Flit reservation flow control," in *In Proceedings of the 6th HPCA*, 2000.

[21] T. D. Richardson et. al, "A hybrid SoC interconnect with dynamic TDMA-based transaction-less buses and on-chip networks," in *Conference on VLSI Design*, 2006.

[22] SPEC, "SPEC benchmarks," http://www.spec.org.

[23] M. B. Taylor et al, "Scalar operand networks: On-chip interconnect for ilp in partitioned architectures," in *HPCA*, Feb 2003.

[24] TPC, "TPC benchmarks," http://www.tpc.org.

[25] H.-S. Wang et al, "Orion: A power-performance simulator for interconnection networks," in *Proc. of MICRO-35*, 2002.

[26] D. Wiklund and D. Liu, "SoCBus: Switched network on chip for hard real time embedded systems," in *IPDPS*, 2003.

[27] P. T. Wolkotte et. al, "An energy efficient reconfigurable circuit-switched network-on-chip," in *Proc. of IEEE IPDPS*, 2005.

[28] S. Woo et. al, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. of the 22th ISCA*, 1995.