SCARAB: A Single Cycle Adaptive Routing and Bufferless Network

Mitchell Hayenga
University of Wisconsin-Madison
hayenga@ece.wisc.edu

Natalie Enright Jerger
University of Toronto
enright@eecg.toronto.edu

Mikko Lipasti
University of Wisconsin-Madison
mikko@ece.wisc.edu

ABSTRACT

As technology scaling drives the number of processor cores upward, current on-chip routers consume substantial portions of chip area and power budgets. Since existing research has greatly reduced router latency overheads and capitalized on available on-chip bandwidth, power constraints dominate interconnection network design. Recently research has proposed bufferless routers as a means to alleviate these constraints, but to date all designs exhibit poor operational frequency, throughput, or latency. In this paper, we propose an efficient bufferless router which lowers average packet latency by 17.6% and dynamic energy by 18.3% over existing bufferless on-chip network designs. In order to maintain the energy and area benefit of bufferless routers while delivering ultra-low latencies, our router utilizes an opportunistic processor-side buffering technique and an energy-efficient circuit-switched network for delivering negative acknowledgments for dropped packets.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiprocessors-Interconnection architectures; C.1.4 [**Parallel Architectures**]: Distributed architectures

General Terms

Design, Performance

Keywords

Interconnection networks, multi-core, routing

1. INTRODUCTION

Historically, switched networks have relied on in-router buffering to handle routing conflicts; when two outbound packets are destined for the same link, one must be buffered while the other is transmitted. These traditional buffered routing approaches were derived in an era when the source and destination nodes were far apart, and retransmission due to conflicts was considered either unacceptable or something to be avoided at all costs. In contrast, in on-chip networks, the source and destination nodes are in close proximity of each other. Furthermore, advances in on-chip router microarchitecture have greatly decreased end-to-end latency and provided abundant inter-node bandwidth [11, 17, 22]. Both the low end-to-end latency provided by these microarchitectural improvements and the proximity provided on-chip lessens the power and performance cost of retransmission making it a viable alternative to buffering.

In addition to being less necessary for performance in comparison to their off-chip counterparts, network buffers add pressure to the area and power constraints for on-chip networks. Recent designs from Intel have shown on-chip interconnects consuming as much as 28% of total chip power, with 22% of router power being consumed by network buffering resources [7]. Removing in-network buffering and performing processor-side buffering through utilization of the miss status handling registers (MSHRs) that are already being held for outstanding requests, is attractive as their use can lead to fewer total required buffering resources. Fewer overall buffering resources will reduce the dynamic and leakage power of the on-chip network.

In this paper we propose SCARAB, a single-cycle minimally-adaptive routing and bufferless router for on-chip interconnection networks. SCARAB is a processor-side buffered router which supports a dropping protocol for dealing with routing conflicts. It employs multiple novel mechanisms to reduce the likelihood of packet drops and retransmission costs.

With these mechanisms, the SCARAB network architecture makes the following novel contributions:

Optimized NACK network.
A fixed-delay, circuit-switched negative acknowledgement (NACK) network is utilized for energy-efficient packet retransmissions. As packets progress through the SCARAB network, they reserve NACK wires on a NACK network to trigger retransmission from the packet source upon network contention. Signaling NACKs on a separate, pre-allocated network after routing data on a bufferless minimally-adaptive network results in deterministic latency, which enables successful packet transmissions to be implicitly acknowledged (ACKed). Additionally, time division multiplexing and other techniques are employed to lessen the NACK network overhead.

MICRO'09, December 12–16, 2009, New York, NY, USA.

Opportunistic processor-side buffering.
Although the routers within the network provide no buffering, SCARAB can utilize idle MSHRs at intermediate nodes to opportunistically buffer in-flight packets, provided the ejection port is idle. Once opportunistically buffered, future retransmissions are initiated from the intermediate node rather than the original source, saving latency and power. We detail a heuristic for when to opportunistically buffer which depends upon the successful transmission rate observed from each router port and the number of local MSHRs available.

High-performance minimally-adaptive allocator.
To lessen network contention, while retaining high frequency operation, we detail the implementation of a novel high performance, minimally-adaptive switch allocator design. The short critical delay path through this allocator enables high-frequency single-cycle packet latency at intermediate nodes.

We compare SCARAB to previously proposed bufferless on-chip routers and establish the advantages it offers for performance and energy efficiency. For comparison purposes, we implement a modified version of the router proposed in [26]. This hot-potato (HP) router utilizes fully adaptive routing and a priority protocol to ensure packets reach their destinations in a timely fashion. We also implement a router similar to the Blind Packet Switched (BPS) router presented by Gomez et. al [6].

From synthesized RTL verilog and C-model performance simulations on real-world applications, we show the dynamic energy per active cycle across all evaluated routers is within 1%, but the SCARAB network demonstrates 12.2-17.6% less observed packet latency resulting in fewer active network cycles. Thus we find that the SCARAB network, in addition to delivering better performance, is up to 18.3% more dynamic energy efficient than previously proposed bufferless architectures. Due to their similar overall structure and comparable chip area, we do not expect there to be significant differences in leakage power for these designs, so we only report detailed results for dynamic energy.

Section 2 provides in-depth coverage of our baseline bufferless routers. Section 3 details the microarchitecture of the SCARAB router. Section 4 compares the area, frequency, and power of the router models. Section 5 contains the performance results evaluation and associated discussion. Section 6 details related work. Section 7 summarizes and concludes the paper.

2. BASELINE ARCHITECTURES

This section details the selected baseline routers which were implemented and benchmarked against the SCARAB router. In defining the baseline architectures, the design tradeoffs involved in each are highlighted and discussed. All routers are evaluated in the context of an on-chip 2D mesh.

2.1 Hot-Potato Architecture

Definition.
Hot potato routing is a recently proposed technique for use in bufferless on-chip networks [15, 16, 26] and is derived from the historic HEP's network switch architecture [23]. These switches utilize hot-potato routing to enable both non-dropping and bufferless operation. Hot-potato routing enables bufferless and NACK-less operation by forcing all arriving flits at a node to be routed out on the next cycle, even if a non-productive route must be taken.

The HP router consists of a two-stage pipeline; the first stage is switch allocation (SA) and the second stage is switch traversal (ST). Packet headers contain a priority vector which records the number of router hops a packet has traversed. This priority field is used to ensure older packets are more likely to obtain profitable routing to their destinations. Each cycle, the switch allocator services input requests in order of decreasing priority. If a high-priority packet can select multiple profitable outputs and any lower priority packets have one of these outputs as their only possible profitable route, the high-priority packet will receive the less contested output.

Implementation.
Our implementation of the HP router is designed to be functionally equivalent to the router defined in [26] although it differs with handling multi-flit packets and in some respects to the other hot-potato routers [15, 16].

The authors consider only single-flit packets [15], while our adaptation to multiple-flit packets highlights some resulting design complexities. If a multi-flit packet has won arbitration for an output port it must be sent in its entirety before allowing the port to be reallocated. All flits of a multi-flit packet must be routed to the same output port in consecutive cycles since only the head flit contains routing and destination information. This differs from the multi-flit implementations in [16, 26] which can replicate header information in each flit, hence enabling packet fragmentation and switch reconfiguration in response to output port contention.

Also, our HP router must have enough additional injector-side buffering to hold a maximal length packet. This buffering is required because the hot-potato router must be able to temporarily misroute a packet out of the injector port. The injector-misroute case occurs when a multi-flit packet from the injector has won allocation for an outward bound port on one cycle, and all other input ports register an incoming packet on the next clock cycle. All of the incoming packets could be maximum length packets and one of them will require temporary buffering on the local port.

Disadvantages.
The HP router suffers from a number of shortcomings fundamental to operation and the design choices necessary for its implementation.

First, the HP allocator is intrinsically serial since it must support the case where packets on all input ports must be routed to all output ports. That is, given incoming packets on all ports, each must be serviced in order of decreasing priority as the output port selections of the higher priority packets determine which outputs are still available to lower priority packets. Prior work details the difficulty of achieving competitive clock frequencies with hot-potato routing [19, 20, 26]. The reported critical path has a depth of 50 gates [19, 20] and can only achieve a 500 MHz operation in 0.13 *um* technology [26]. Our own implementation, further detailed in Section 4 was only able to achieve operation at 857 MHz, a factor of two slower than our other allocators. No operating frequency analysis was done in [16] or [15].

Second, [26] assumes multi-flit packets can be fragmented in-flight and reassembled at the destination node. This assumption drastically increases the destination buffering requirement as every node must be able to resequence packets from potentially every other node within the network. It should be noted that [16] demonstrates an observed doubling in required destination buffering from benchmark runs, but this is not the maximal bound which must be allocated to guarantee that the network will never drop packets.

Finally, Moscibroda and Mutlu [15] note that their network is livelock free since they have a priority field encoding a total ordering of packets, guaranteeing a packet eventual delivery at its destination node. This statement makes the assumptions that 1) priorities can not saturate and 2) all packets are single flits or packet fragmentation is supported. If priorities can saturate it is possible for multiple maximum priority packets to deflect each other, preventing any from reaching their final destination. The HEP solved the saturating priority issue by forcing all packets to take a deterministic route once they reached a maximum priority [23]. Additionally, if not all packets are single flits, livelock is still possible when outputs are allocated for multiple cycles. It is possible for a packet, regardless of its priority, to never get routed to a destination node since all productive outputs could have been allocated in a previous cycle. Once the packet is misrouted, nothing prevents it from facing this same situation again on all subsequent clock cycles.

One of the advantages of bufferless routing, in addition to power, is being able to operate at higher clock frequencies due to removing buffer writes from the critical path [6]. Unfortunately, the strict allocator imposed by hot-potato routing and complexity in solving livelock issues may limit these performance gains.

2.2 BPS Architecture

Definition.

The BPS router is a NACK-based bufferless router which performs minimally adaptive routing to limit the number of possible collisions. In the event of a collision, a NACK packet is routed through the network back to the source node to trigger retransmission of the original data packet. Because NACKs cannot be dropped, small NACK buffers are present at every input port. The ACKs within the BPS network are implicit since the maximum time necessary to wait to receive a NACK is deterministic. This is because NACK buffers at each router are serviced in a deterministic order and NACKs carry a higher priority than regular data packets. Due to their higher priority, NACK packets can cause data packets to abort leading to higher drop rates.

The BPS proposal also used Space Division Multiplexing (SDM), splitting large 256-bit links into four 64-bit links, in order to limit network contention for shared links. By reducing contention, SDM improves the packet drop rate at the expense of additional allocation and switch configuration logic.

Implementation.

For our implementation we assume that NACK packets only assert a higher priority when NACK buffering at the local router is limited. This leads to lower packet drop rates and does not affect overall network performance assuming additional MSHRs are appropriately provisioned to tolerate a longer, maximally-bounded, implicit ACK latency.

Additionally, our BPS implementation more aggressively targets low latency than the original proposal. The initial proposal suggested using latch chains to temporarily buffer data while routing decisions were made. Instead, we use separate allocation and data networks similar to those in [11]. This enables our BPS baseline to operate as a single-cycle router, sending the allocation packet one cycle ahead of the data packet such that the data switch is fully set up when the data flits arrive.

For purposes of fair comparison, our implemented version of BPS does not use SDM as it is orthogonal to the overall router design and could likewise be applied to the other bufferless routers in order to reduce network contention.

Disadvantages.

Using a shared network for packet transmissions and NACKs hinders the BPS architecture as NACKs from previous routing conflicts further increase total network congestion. The negative impact of NACKs upon the network is likely to accelerate total network saturation under moderate workloads.

Also, though implicit ACKs in BPS have deterministic latency, it is relatively high due to the presence of NACK buffering. This increases pressure on source-side MSHRs as they must remain allocated for longer to ensure successful packet delivery.

3. SCARAB ARCHITECTURE

3.1 Overview

The SCARAB router is designed to eliminate in-network buffering, limit additional processor-side buffering, provide an efficient NACK mechanism for packet retransmission, and scale to high frequencies with low end-to-end network delays.

3.2 Router Pipeline

The SCARAB router implements a single-cycle latency pipeline similar to that in [11]. As illustrated in Figure 1, the packet header travels along a small (17-bit) allocation network one cycle ahead of the data. By optimizing the allocation network to operate with a single-cycle latency, trailing flits on the data network also experience only a single cycle router delay per network hop. Figure 1 demonstrates how multiple data flits can be streamed through multiple hops.

Figure 2 shows that the router can be viewed as three inter-related but physically separate networks: one for allocation, one for data transfers, and one for NACKs. To successfully route a packet, an allocation packet must be sent one cycle before the first data flit is sent on the data network. The cycle before the first data flit arrives, the allocation packet performs switch allocation and traversal. Allocation is successful if a productive output is obtained and free NACK wires exist along this output path. On the subsequent clock cycle, the data flit will immediately proceed to switch traversal, as the allocation packet has preconfigured the data crossbar switch. If allocation cannot be

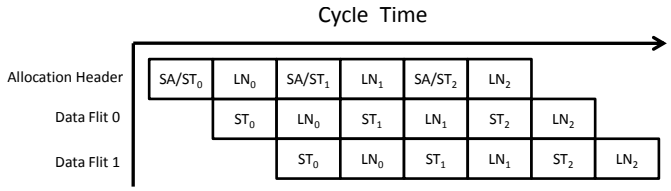


Figure 1: SCARAB single cycle pipeline processing. Allocation header performs allocation (SA) and switch traversal (ST) in a single cycle. Trailing data flits follow pre-configured crossbar switch. All flits observes a single cycle of link delay (LN) for inter-tile latency.

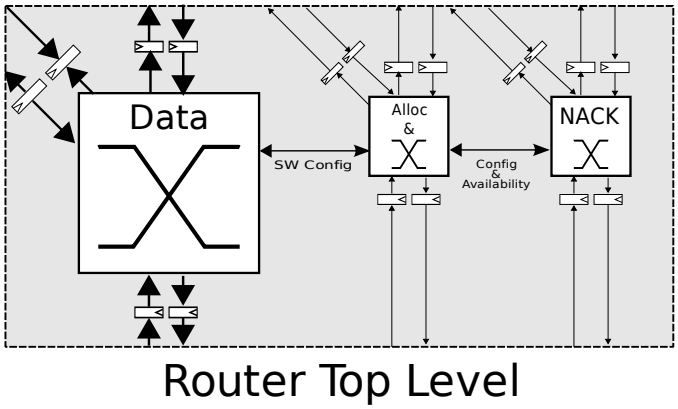


Figure 2: SCARAB router showing the three networks 1) Allocation 2) Data 3) NACK and signaling between them.

performed successfully, the data packet is dropped and on the successive cycle, the NACK wire corresponding to this packet is asserted to the previous node. This NACK signal will travel back across the network to the source to trigger the retransmission of the packet. The NACK network, acting as a pre-configured circuit-switched network, is also only a single router cycle delay from input NACK to output NACK.

3.3 Allocator Design

The SCARAB router employs a minimally adaptive routing algorithm and only allows packets to be routed in productive directions. The input along the allocation network consists of a field containing the following information: a 5-bit one-hot vector encoding productive output ports at this current hop, the priority of the current packet (4 bits), the size in flits of the corresponding data packet (2 bits, allowing 4 packet sizes), and the destination node number (6 bits for an 8x8 network). At 17 bits, the allocation port is reasonably small compared to the 128-bit data port for each input.

At each cycle allocation is performed among incoming allocation packets. During this cycle the allocator:

1. Masks off unavailable outputs (due to previous allocation for a multi-flit packet or lack of available NACK wires) from the incoming request vectors.
2. Counts the number of possible requests per output.
3. Masks off an output port request from requesters with multiple profitable output ports. If the number of requests per outputs are equal, a bit from an LFSR randomly selects one to mask.
4. Computes the maximum priority requesting each output.
5. Presents masked request vectors to a per output round-robin arbiter if their priority is equal to the maximum.
6. Selects a winning request per output using the round robin arbiters.

A packet's allocation priority can be incremented on either a per-hop or retransmission basis. A per-hop basis would provide better fairness to older packets, but in our testing a per-retransmission priority was found to achieve the majority of the benefit and require fewer priority bits. These vectors can also be minimally sized as SCARAB can be made to operate with saturating priority fields. Saturating priorities create potential livelock problems for HP, but do not in SCARAB as processors can implement randomized exponential backoff in the event that a saturated priority MSHR has required multiple retransmissions.

3.4 NACK Network

The NACK network in the SCARAB router operates as a small 5x5 mux-based switch. Every packet that traverses a router configures a 1-bit wire to be asserted from output to input if the packet fails at a future router. This NACK wire is connected along the packet's entire path, such that the source MSHR can be triggered for retransmission in the event of a network collision.

As no buffering is present in the system, the window of time which a packet could be NACKed after initial transmission is deterministically bounded. If the NACK wire to an MSHR is not triggered in $L = 4 \times (N + 1)$ cycles, where N is the number of hops between source and destination, the packet has been implicitly ACKed and the MSHR can be discarded. Each router in the network along the path between source and destination must also allocate these NACK wires for at least $4 \times (N+1)$ cycles before allowing reallocation. L cycles must elapse before reallocating NACK wires because it takes 2 cycles to enter the network from the injection port, $2 \times N$ cycles to progress to the destination router at which the packet could fail arbitration for the local port, leading to an additional $2 \times (N + 1)$ cycles for the NACK to arrive at the source injector.

To highlight the need for an efficient NACK network, we evaluate the storage overhead of a naïve implementation. A naïve implementation would allocate a counter per NACK wire. As derived in the previous paragraph, each counter would require $\log_2(4 \times (max(N) + 1))$ bits. The entire router node would require a F counter flip-flops, $F = PW \log_2(4 \times (max(N) + 1))$, where P is the number of router ports and W is the number of NACK-wires between nodes. Although this equation grows logarithmically with increasing network size, the linear growth with respect to NACK wires is unacceptable. Additionally, for scalable performance, the number of necessary NACK wires per router port does increase with network size as messages are transported through more intermediate nodes. Taking this into

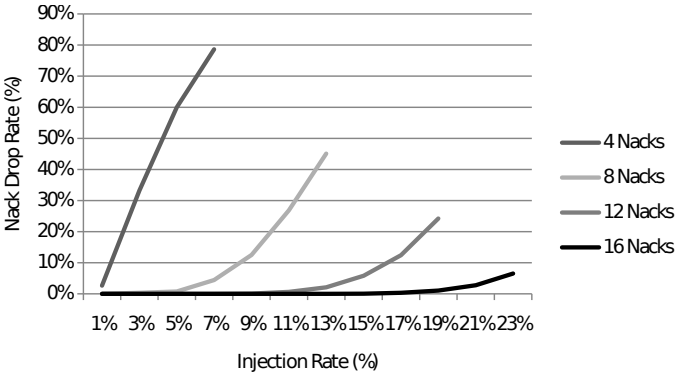


Figure 3: Percentage of packets dropped due to lack of NACK wire availability for an 8x8 network versus the number of available logical NACK wires under a uniform random traffic workload. Drop rates are plotted until network saturation.

account, the number of flip-flops required for scalable performance in a naïve mesh network would scale almost quadratically with increasing network size.

Our router takes advantage of serveral novel techniques to reduce the cost of the NACK network and scale logarithmically with increasing network size. First, NACK wires are managed in groups to reduce the number of cycle-counters required to determine when an individual NACK wire can be reallocated. NACKs for each port are managed in two allocation epochs. All NACKS within an epoch are allocated until no free NACKs exist. Each epoch maintains a counter equal to the maximum number of cycles for any NACK allocated within this epoch. This counter is decremented until it is equal to zero, at which point in time all NACKs wires within this epoch are considered allocatable. Secondly, our network takes advantage of time division multiplexing of the NACK wires to lessen the number of NACK wires needed between routers. Since it takes two cycles to reach the allocation phase of the next router, we implicitly know any packet we send out on an even clock cycle can only be NACKed on a later even cycle. This enables us to halve the number of inter-router NACK wires necessary to sustain network bandwidth. For better management of time-based multiplexing we have four total epoch counters per port, two for even clock cycles and two for odd clock cycles. Finally, our epoch-based allocation assigns NACK wires in an ordered fashion such that neighboring nodes implicitly know the next sequential NACK which will be allocated. This property eliminates the need for communicating a NACK wire identifier in the allocation packet.

To demonstrate the scalability of this NACK network design, Figure 3 shows the percentage of failed transmissions due to unvailable NACK wires on an 8x8 network with 4, 8, 12, and 16 logical NACK wires. The traffic workload was uniform random traffic and these tests were performed on a SCARAB network with priorities, but without the opportunistic buffering feature mentioned in the next section. For low injection rates, a small number of NACK wires is sufficient to limit packet drops due to NACK wire contention. Also, with more NACK wires in a single epoch, NACK network performance scales despite wires being allocated for a longer duration due to the group management of epochs. Additionally, the need to scale much further beyond 16 logical NACK wires for larger networks is likely small as it is highly workload dependent and performant applications running on large scale networks would have a limited number of global references spanning the entire network. For example, server consolidation workloads are likely to exhibit a higher percentage of local requests compared to global requests.

Using a 1-bit NACK network results in a very energy efficient means of signaling retransmission in comparison to the BPS network which would need to send a full data packet to request retransmission. In addition to being very small, the NACK network experiences very little activity assuming network contention is kept at reasonable levels. Finally, separating the NACK network from the data network creates deterministic delays and a tighter bound on how long MSHRs must remain allocated for processor-side buffering. In the BPS network, this bound exists but is significantly larger due to buffering delays that NACKs may face in-flight to their source node.

3.5 Opportunistic Buffering

To increase the scalability and performance of SCARAB, we further optimize the network with a novel processor-side buffering technique that comes at almost no hardware cost. *Opportunistic buffering* is the temporary borrowing of another node's MSHR buffers such that retransmissions occur not from the packet's source, but from an intermediate node in the network. Intermediate buffering leads to both lower energy consumption and lower latency per retransmission. This opportunistic buffering should not be confused with traditional in-network buffering as no additional buffer resources are necessary; opportunistic buffering puts otherwise idle buffers to use.

Implementing opportunistic buffering requires only small router changes. First, opportunistic buffering only occurs when a packet is determined to benefit from buffering, the local ejection port is idle, and sufficient free MSHRs exist. Signaling free MSHRs involves the addition of a single wire from the injection port to the local router. From experimentation on real application traces (see Section 5.2 for application descriptions), we have derived a heuristic for when to opportunistically buffer. To ensure a node is not starved of its own MSHRs we limit opportunistic buffering to only use up to one quarter of the total available MSHRs. Additionally, we do not signal the availability of buffers if the local node is using more than half of its MSHRs. This policy leaves sufficient free MSHRs for a local burst in traffic.

To determine when it is potentially beneficial to opportunistically buffer a packet, we have designed a novel heuristic to identify *likely to fail* packets. For each output direction we maintain a single counter which keeps the running average priority of all packets routed out of the output port. If a packet's priority is less than or equal to this average priority, the packet is deemed at risk and should be opportunistically buffered if possible. Since multiple output directions may match this heuristic during a given cycle, we grant the MSHR to the requestor with the highest priority. The packet is then routed out both the output and local ports. The local port must check the incoming packet's destination to determine if they are being opportunistically buffered or destined for this current node. To simplify logic, we only al-

low single-flit packets to be opportunistically buffered; support for multi-flit packets would require logic to truncate partially buffered packets and rollback alterations to the NACK network necessary for opportunistic buffering. With our application workloads, ∼70% of packets are single-flit, so limiting processor-side buffering to only single-flit packets will capture the majority of opportunity with low hardware cost.

The NACK network protocol must be modified to support opportunistic buffering. Once a packet is selected for opportunistic buffering, instead of tying the NACK wire from the packet's incoming port to its outgoing port, the protocol ties the input and output port NACK wires to the local port so that it can intelligently manage NACK actions from future nodes.

Additionally, the source node for a packet must be notified not to treat the MSHR as retired. To do this we extend the pulse-based NACK protocol to include level-based signaling to leave NACK wires intact for $4 \times (N + 1) + 2$ cycles. If a pulse is received, a packet is interpreted as having been NACKed. If a level raise is received, it indicates a downstream node is opportunistically buffering the packet and intermediate nodes should not tear down their associated NACK wires or treat the source MSHR as retired. Once the opportunistically buffering node is implicitly ACKed indicating successful transmission, this NACK level is lowered signaling completion. An opportunistically buffering node can itself receive a level raise from a future node, at which point it leaves the NACK wires configured, but is allowed to free the intermediate MSHR as it is no longer necessary. This event implies that a packet has been again opportunistically buffered at a downstream node. For clarity, these steps are explained as a detailed graphical example in Figure 4.

Opportunistic buffering allows our router to scale to higher saturation points, better deal with network congestion, and better scale to larger network sizes; opportunistic buffering is evaluated in Section 5.

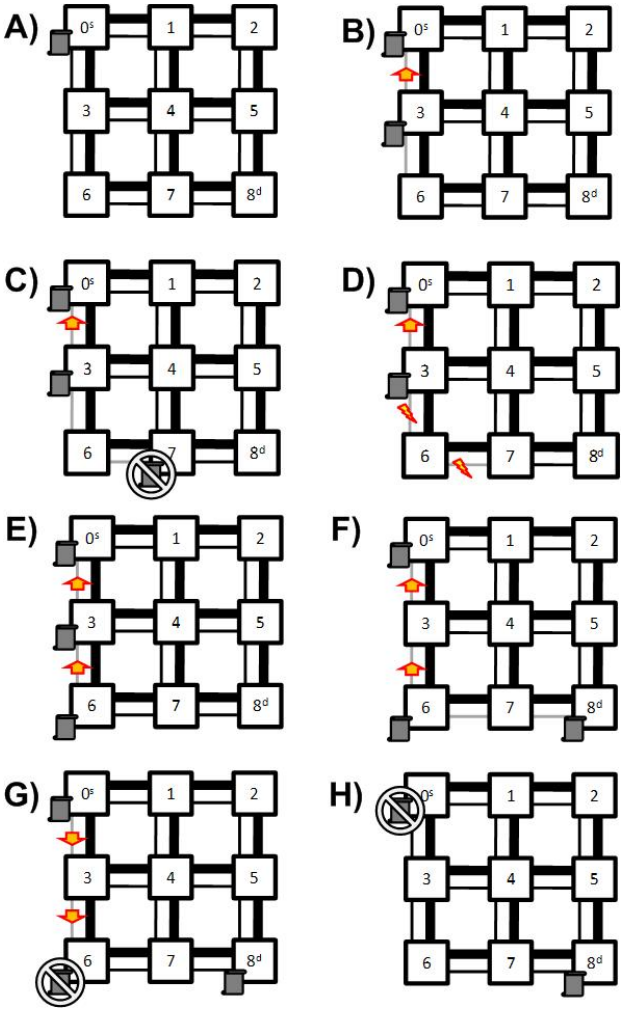


Figure 4: Example NACK operation in the SCARAB network. A) Packet awaits transmission from source node 0 to destination node 8. B) Packet has progressed to 3 and becomes opportunistically buffered. 3 raises the level of the associated NACK to signal that the MSHR cannot be deallocated and the NACK wire must remain intact. C) Packet progresses successfully onward to node 7 before failing. D) 7 pulses the NACK causing 3 to retransmit the packet E) The packet, retransmitted from 3 reaches 6 where it is again buffered. 6 raises the level of the associated NACK wire causing 3 to deallocate its MHSR. F) Packet successfully reaches destination node 8. G) The NACK wire at 6 implicitly times out, causing it to lower the NACK wire back to the source. H) Source sees the NACK level drop and deallocates the original MSHR.

4. PHYSICAL IMPLEMENTATION

4.1 Methodology

To ensure accurate hardware modeling all compared routers were implemented in RTL Verilog. Results for area, power, and frequency are derived from synthesizing to the TSMC 65nm standard cell library using the Synopsys Design Compiler ® with high optimization enabled. For dynamic power calculations, uniform switching activity was assumed on all router input ports and we report the clock-normalized dynamic energy per cycle.

For all three routers, equal effort was placed into optimizing the designs for cycle time. For synthesis, the configuration parameters as specified in Table 2 in Section 5 were used, with the only deviation being that a fixed 4-bit priority field, rather than 64-bit priority used in the performance evaluation, was used in the synthesis of the HP router.

4.2 Implementation Results

BPS and SCARAB.

Due to the fact that both the BPS router and SCARAB router are dropping, minimally adaptive routers, a similar allocator structure was utilized for both, allowing each to

Table 1: Router Synthesis Results

Router	Area (um^2)	Energy (pJ)	Frequency
SCARAB	34.1K	24.32	1.9GHz
BPS	33.7K	24.42	1.9GHz
HP	31.2K	24.46	857MHz

be a single-cycle router. This minimally adaptive allocator, when synthesized by itself, achieves a 2.5GHz frequency. The combined allocator and crossbar latency is the critical path for both the BPS and SCARAB routers, as was expected for such an aggressive design. As shown in Table 1 the combined allocator/crossbar of SCARAB and BPS routers limit their frequency to 1.9GHz, with the reduction in allocator frequency due to a combinational path necessary to signal the availability of NACK wires(SCARAB) or buffers(BPS) to neighboring nodes that is excluded from the standalone allocator module. The 1.9GHz frequency for each of these routers corresponds to a delay of 16.8FO4, given our library's 31.3ps FO4 delay. A 16.8FO4 delay is competitive with aggressive router implementations which target 20FO4 [22] and 35FO4 [17]. The SCARAB network is slightly more expensive in terms of area because its NACK network consumes more area than the NACK buffers present in the BPS router.

HP.

As discussed in Section 2.1, the implementation of a hot-potato switch allocator presents difficulties in achieving an acceptable clock frequency. To lessen the serial nature of allocating outputs in the HP router, we have leveraged insight from high speed arithmetic circuits. Figure 5 details the general algorithm used for both the naïve and more advanced allocator used in our HP implementation.

In Figure 5a, the serial version simply performs a parallel sort of the incoming packets based upon priority and then allocates each output in descending order. Synthesis of this naïve serial version results in a maximum allocator frequency of 670MHz. Figure 5b details a carry-select version of the allocator in which each lower priority packet calculates 5 possible port selections in parallel with the higher priority packet's allocation stage. Once the higher priority packet's output port is selected, the appropriate allocation for the lower priority packet is selected. This carry-select version increases the maximum realizable frequency of the HP allocator to 990MHz. If the sort of the incoming packets based upon priority is not performed within the same clock cycle as allocation, lengthening the router pipeline to 3 stages, the carry-select hot-potato allocator can synthesize at 1.3GHz. When all logic necessary to implement the HP allocator is synthesized, the maximum achieved frequency is 857MHz for the 2-stage design.

Perhaps with further modifications to the allocation algorithm, the HP allocator could be made to operate at an aggressive clock frequency. Giving the HP router the benefit of the doubt, all later performance evaluations within this paper assume that the HP router can operate at the same 1.9GHz frequency as the BPS and SCARAB routers. It should be noted that frequency resulting from a physical implementation was not specified in HP [15, 16].

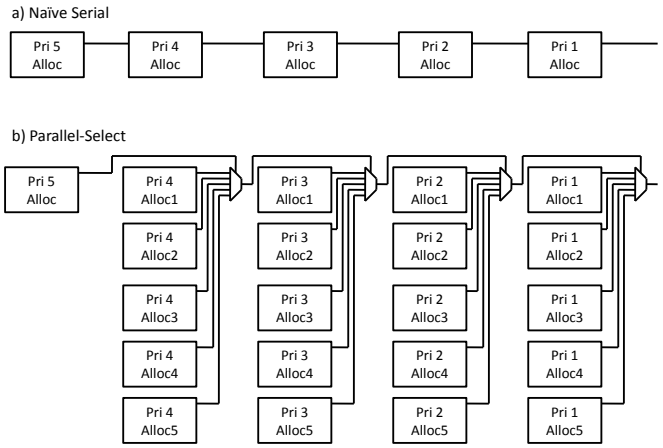


Figure 5: HP allocator alternatives.

Power Analysis.

Table 1 shows all three routers expend approximately (within 1%) the same amount of dynamic energy for an individual router node on a per-cycle basis assuming a constant and uniform load, irrespective of total network performance. As will be shown in Section 5.2, for real world application workloads the SCARAB network results in 17.6% less latency than HP and 12.2% less latency than BPS. As all three routers are bufferless, lower latency directly corresponds to fewer dropped or misrouted packets that lead to additional network events. Taking the absolute number of network events for the application traces in Section 5.2 and combining them with the given dynamic energy expended on a per-cycle basis we can derive the total dynamic energy necessary to execute the given traces. We find SCARAB to be at least 18.3% more efficient than HP and 12.6% more efficient than BPS on a dynamic energy basis.

5. EVALUATION

We evaluate the SCARAB router against BPS and HP with both synthetic traffic and with traces collected from real workloads. To perform these evaluations, a cycle accurate C++ model of each router was constructed. For SCARAB, two versions, one with opportunistic buffering with retransmission-based priorities and another with no opportunistic buffering or priorities, are used for comparison. If no version is specified, the SCARAB router used is the opportunistic buffering, priority-based version. Also, the evaluated BPS router assumes only a single link between nodes, rather than multiple, smaller links. All routers are assumed to be able to operate at the same 1.9GHz frequency, despite the fact that our verilog implementations indicate that the maximum implementable frequency of the HP router is much less than either the BPS or SCARAB routers. To highlight the fact that the HP router is operating at an idealized frequency, its results are shown as dotted lines and should be regarded as an upper bound on its potential performance. Table 2 specifies all configuration parameters used in the performance evaluation.

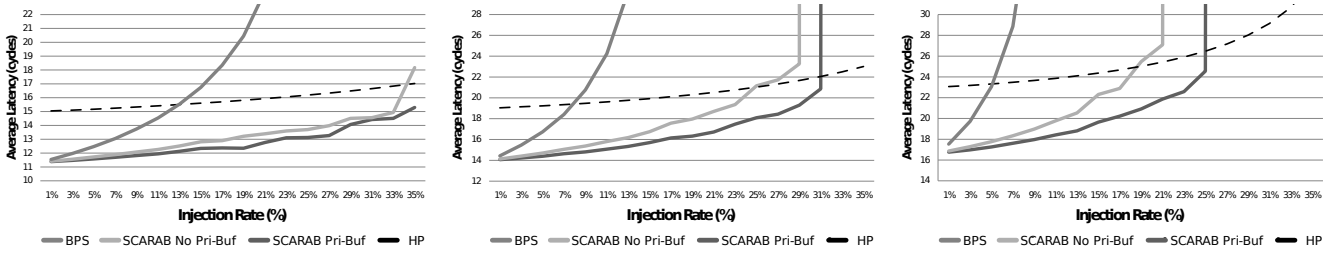


Figure 6: Average packet latency for single-flit uniform random traffic on 4x4, 6x6, and 8x8 networks.

Table 2: Network Configuration

Common Parameters	
Processor Frequency	3.8 GHz
Router Frequency	1.9 GHz
Flit Size	16 bytes (128 bits)
Coherence Packet Size	1 flit
Data Packet Size	5 flits
Network Sizes	4x4, 6x6, 8x8 Mesh
SCARAB Router	
Pipeline Latency	1-cycle
Packet Priority	4-bits, Retransmission-based
Routing Algorithm	Minimally Adaptive
MSHRs	16
NACK Wires	8 per port (16 logical)
BPS Router	
Pipeline Latency	1-cycle
Packet Priority	4-bits, Retransmission-based
Routing Algorithm	Minimally Adaptive
MSHRs	16
NACK Buffers	8 per port
HP Router	
Pipeline Latency	2-cycle
Packet Priority	64-bits, Hop-based
Routing Algorithm	Fully Adaptive
MSHRs	Infinite

5.1 Synthetic Workload Evaluation

To determine how the routers compare across multiple network sizes they were directly compared with networks of 16, 36, and 64 nodes using uniform random, tornado, and neighbor traffic traces. Single-flit traffic should result in the highest total network utilization for all routers. For all synthetic traffic patterns we evaluated the performance of four router models. 1) SCARAB with priorities and opportunistic buffering 2) SCARAB without priorities or opportunistic buffering 3) HP and 4) BPS.

MSHR modeling in the SCARAB and BPS routers greatly impacts the synthetic traffic workload performance in comparison to the HP router. As synthetic, single-flit traces operate independently of coherence or consistency constraints, the HP router can free MSHRs as soon as packets are injected into the network. In a real system, MSHRs are held until the miss is satisfied (data is returned). This makes MSHRs on HP for synthetic traffic effectively unlimited. SCARAB and BPS can not free MSHRs upon injection due to their requirement to retransmit from MSHRs when network collisions occur. In the following traffic patterns, the SCARAB and BPS routers saturate in many cases once their MSHRs are fully occupied. It should be noted that for realistic network workloads, which guard against race conditions, this artificial advantage of HP does not apply.

Uniform Random Traffic.

Figure 6 shows the average packet latency for each router model on uniform random traffic traces for networks of 4x4, 6x6, and 8x8. The simulations across all network sizes produce consistent results with the relative performance rankings of the routers unchanged. For all network sizes BPS has low zero-load latency, but quickly saturates at low injection rates. This phenomenon supports the hypothesis that transporting NACKs along the data network adds significant network pressure, quickly leading to saturation. The HP router, as a 2-cycle router, has a significantly higher zero-load latency but approaches saturation very gradually and demonstrates the best average packet latency at high network injection rates. This characteristic of the HP router suggests it is well suited to latency insensitive, high-bandwidth applications. For more typical injection rates of less than 15% [4, 5, 9], the SCARAB router demonstrates the best average packet latency. Additionally, the relative benefit of opportunistic buffering and retransmission-based priorities are shown to increase as network size increases.

Tornado Traffic.

To model potential worst-case performance, the tornado traffic pattern was evaluated against the router models on an 8x8 network. Figure 7 shows that the saturation point for all routers is significantly less than their corresponding saturation point on uniform random traffic. This adversarial traffic pattern has no impact on the respective ordering of the routers' performances. For this workload and network size, the BPS router performs very poorly and saturates just beyond a 3% injection rate. SCARAB and HP scale much better, achieving reasonable latencies with moderate traffic.

Neighbor Traffic.

To approximate the best traffic pattern for bufferless routers, we use the neighbor traffic pattern on an 8x8 network. In this trace all nodes randomly transmit to their immediate neighbors. Figure 8 shows that all routers perform quite well; all are able to achieve a 30% injection rate prior to

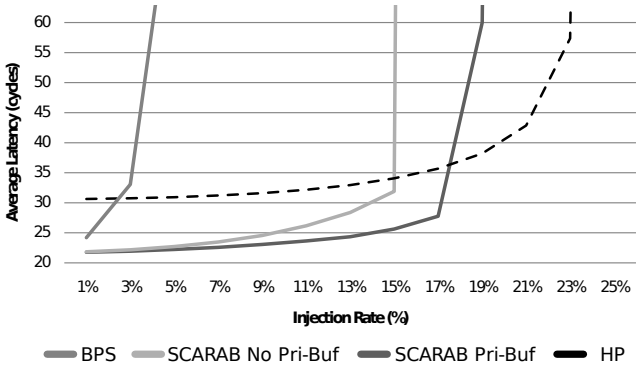


Figure 7: Average packet latency for tornado traffic on an 8x8 network.

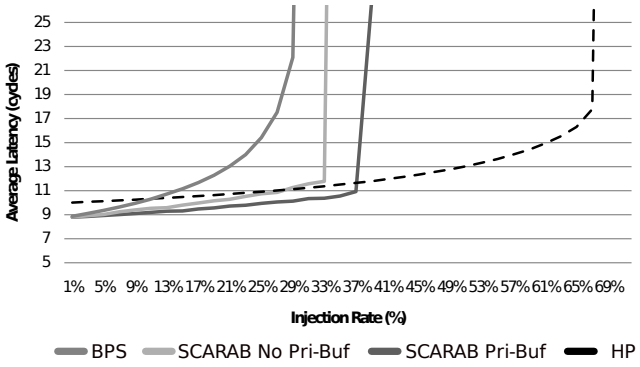


Figure 8: Average packet latency for immediate neighbor traffic on an 8x8 network.

saturating. This figure exhibits two notable features. First, the single cycle routers realize a one cycle zero-load latency benefit over the the 2-cycle HP router. This single cycle benefit is obtained from the second crossbar traversal required to reach an immediate neighbor. Second, SCARAB scales similarly to the HP router until it quite suddenly saturates around 40%. This occurs because the average service time of a request causes the MSHRs at local nodes to be exhausted. As the HP router releases MSHRs upon injection, it does not experience this sudden saturation on the synthetic traffic trace.

From this synthetic testing, it can be seen that all three bufferless routers could see use cases for certain applications and network sizes. Overall, SCARAB obtains lower packet latencies and saturates later than the tested version of the BPS router. BPS with multiple links per direction could improve its situation, but would not displace the SCARAB router since the same techniques could be applied to it. HP typically has higher latencies, but offers higher saturation points making it competitive with the SCARAB router for very high injection rates.

5.2 Application Driven Evaluation

To obtain a better view of how the different router models affect real-world application performance, this section

Table 3: Benchmark Descriptions

SPECjbb	Standard java server workload utilizing 24 warehouses, executing 200 requests
SPECweb	Zeus Web Server 3.37 servicing 300 HTTP requests
TPC-W Tier	TPC's Web e-commerce benchmark, DB Browsing mix, 40 web transactions
TPC-H	TPC's Decision Support System Benchmark, IBM DB2 v6.1 running query 12 w/ 512MB database, 1GB of memory
Ocean	514x514 full end-to-end run (parallel phase only)
Radiosity	-room -batch -ae 5000 -en .050 -bf .10 (parallel phase only)
Raytrace	car input (parallel phase only)

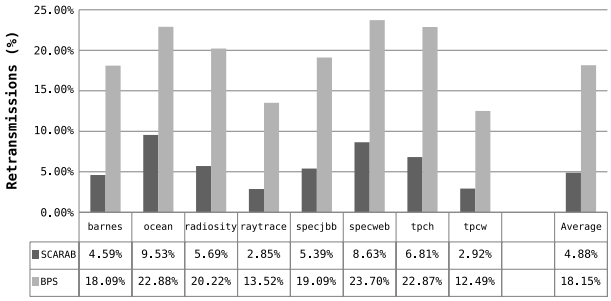
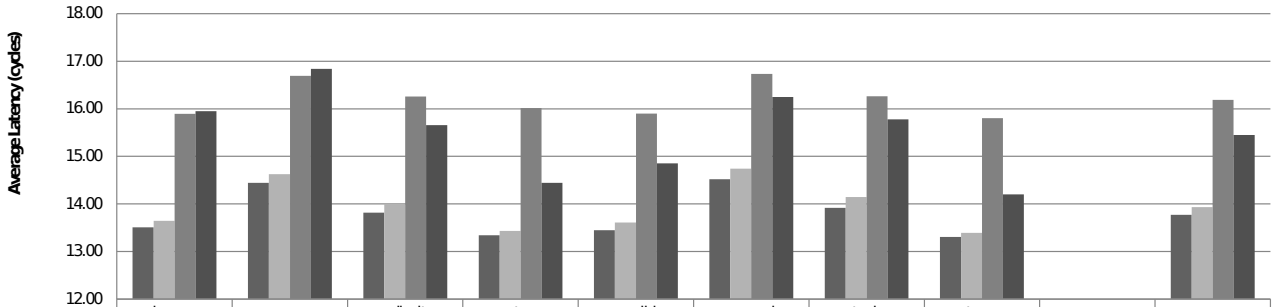


Figure 9: Average packet drop rate on SPLASH-2 and Commercial benchmark traces on a 4x4 node network for the BPS and SCARAB routers.

utilizes traces from simulated applications. The on-chip network traffic traces are collected from a full system simulator [3] for end-to-end runs of 8 workloads including 4 commercial workloads [24, 27] and 4 scientific workloads [30]. Workload details are presented in Table 3. These workloads are for a 4x4 CMP with a directory protocol modeled after the SGI Origin [13]. We simulate 32KB L1 I/D caches and private 256KB L2 caches. Addresses are distributed across 16 directories with one directory located at each processor tile. These traces contain a mix of packet sizes, coherence messages create single-flit packets, while cache line transfers (64 bytes) create 5-flit packets.

Figure 10 shows the average packet latency of all routers across the collected application traces. From this graph, we infer that network utilization is quite low since the BPS router module delivers on-average better performance than the HP router. This low network utilization was verified and of the selected applications, Ocean had the highest injection rate of around 5%. The SCARAB routers deliver the best performance across all applications. The difference between the opportunistic buffering and non-buffering SCARAB router is very slight and this can partially be attributed to the small 4x4 network size used in these traces. On average, the opportunistic buffering SCARAB router is 12.2% faster than the BPS router and and 17.6% faster than the HP router on real-world applications.

Since both packet dropping routers, BPS and SCARAB, perform the best on the application traces, their respec-



	barnes	ocean	radiosity	raytrace	specjbb	specweb	tcph	tpcw		average
SCARAB Hop-Pri, Ad-Buff	13.51	14.45	13.82	13.34	13.45	14.52	13.92	13.31		13.77
SCARAB No-Pri, No-Buff	13.64	14.62	14.00	13.43	13.61	14.74	14.15	13.39		13.93
HP	15.89	16.70	16.26	16.02	15.90	16.74	16.26	15.81		16.19
BPS - 1 Link	15.95	16.84	15.66	14.44	14.85	16.25	15.78	14.20		15.45

Figure 10: Average packet latency for SPLASH-2 and Industry benchmark traces on a 4x4 node network.

tive drop rates correspond to how much power each router wasted. Figure 9 shows the percentage of packet retransmissions necessary to successfully complete the application traces. On average the SCARAB router has to perform retransmission for 4.88% of all packets. The BPS router scales more poorly, due to the fact it has to send NACKs along the same data network, and has to retransmit 18.15% of all packets. The presence of data-network NACKs in the BPS network degrades performance in times of congestion and bursty traffic, since the NACKs add congestion to the network.

6. RELATED WORK

Bufferless routers.

Circuit switching is another bufferless routing technique which has been evaluated in the context of on-chip networks. Enright Jerger et al. [5] evaluate traditional circuit switching and compare it to a hybrid combination of circuit switching and packet switching. Banerjee et al. [1] evaluate the area and power of an on-chip circuit-switched network in comparison to a wormhole router and the router from [17]. Circuit switching impacts low-load latency by requiring a circuit setup phase; SCARAB has the advantage of providing low latency at low and moderate loads by not requiring a distinct setup phase.

Bouhraoua and Elrabaa [2] propose a bufferless on-chip network based upon a fat tree topology that targets throughput, rather than latency, as its primary objective. The Nostrum network on-chip [14] also proposes a deflection-based, bufferless router structurally similar to the HP router, but augmented it with a protocol to provide two classes of network transactions: guaranteed bandwidth and best-effort. Moscribroda et. al propose a hot-potato based bufferless router which extends prior bufferless proposals in order to provide wormhole-based, livelock free routing [16].

Buffer reduction.

Many works have focused upon reducing, but not eliminating, the buffering requirements of on chip networks. Kodi et al. [10] advocate multipurposing repeater logic on internode links as storage elements in order to reduce the in-router buffering requirement. Dynamically Allocated Multi-Queue (DAMQ) [25] buffering reduces the total buffer requirement per router port by sharing buffer space across multiple virtual channels. ViChaR [18] works to improve buffering efficiency by dynamically adjusting the depth and number of virtual channels in response to network traffic. Also, application specific buffering approaches have been proposed in order to optimize their allocation [8].

Power optimized.

Optimizing on-chip network parameters to lessen total power consumption has been widely studied in recent years [1, 21, 29]. Additionally, many microarchitectural techniques have been proposed to reduce dynamic power. Kumar et al. [12] avoids dynamic buffer activity in scenarios where buffer reads and writes can be avoided. Wang et al. [28] detail multiple microarchitectural techniques impacting crossbar design and write buffers. Bufferless networks such as SCARAB reduce both the dynamic power consumed by buffers as well as the leakage power.

7. CONCLUSION

We propose SCARAB, a single-cycle bufferless router that relies on adaptive routing, a novel circuit-switched NACK network, priority-based arbitration, and opportunistic processor-side buffering to provide the lowest possible latency and reasonable saturation bandwidth relative to two previously-proposed bufferless networks: BPS [6] and HP [26]. Detailed design and evaluation shows that SCARAB is amenable to very high frequency implementation, provides the lowest end-to-end latency under low and moderate loads, and scales to higher utilization than the BPS router [6], due to its novel circuit-switched NACK network. In contrast, the HP router requires a complex arbiter that is likely to pose a cycle-time bottleneck and suffers from worse latency at low and moderate utilization, but does scale to higher saturation throughput. In summary, SCARAB appears most attractive for memory latency-bound commercial workloads which are an important target for future many-core architectures.

In future work, we plan to explore a broader set of policies for opportunistic buffering in larger networks and different topologies. Furthermore, we will evaluate these networks in a full-system, execution-driven simulator, and will investigate the impact of bufferless networks on coherence proto-

col implementation, processor core and buffer design, and memory consistency models.

8. ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation under grant CCF-0702272, an NSERC Discovery Grant, as well as grants and equipment donations from Hewlett-Packard, IBM, and Intel.

9. REFERENCES

[1] A. Banerjee, R. Mullins, and S. Moore. A power and energy exploration of network-on-chip architectures. In *NoCS '07*, pages 163–172, Washington, DC, USA, 2007. IEEE Computer Society.

[2] A. Bouhraoua and E. Elrabaa. A high-throughput network-on-chip architecture for systems-on-chip interconnect. In *System-on-Chip, 2006. International Symposium on*, pages 1–4, Nov. 2006.

[3] H. W. Cain, K. M. Lepak, B. Schwarz, and M. H. Lipasti. Precise and accurate processor simulation. In *Workshop on Commercial Architecture Evaluation using Commercial Workloads*, 2002.

[4] S. Cho and L. Jin. Managing distributed, shared L2 caches through OS-level page allocation. In *MICRO-39*, pages 455–468, Washington, DC, USA, 2006. IEEE Computer Society.

[5] N. Enright Jerger, L.-S. Peh, and M. H. Lipasti. Circuit-switched coherence. In *Proceedings of the 2nd Annual Network on Chip Symposium*, April 2008.

[6] C. Gomez, M. Gomez, P. Lopez, and J. Duato. An efficient switching technique for nocs with reduced buffer requirements. In *ICPADS-14*, pages 713–720, Dec. 2008.

[7] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. A 5-GHz mesh interconnect for a teraflops processor. *Micro, IEEE*, 27(5):51–61, Sept.-Oct. 2007.

[8] J. Hu and R. Marculescu. Application-specific buffer space allocation for networks-on-chip router design. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 354–361, Washington, DC, USA, 2004. IEEE Computer Society.

[9] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. In *MICRO-40th*, pages 172–182, Dec. 2007.

[10] A. K. Kodi, A. Sarathy, and A. Louri. iDEAL: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures. In *ISCA-35*, pages 241–250, Washington, DC, USA, 2008. IEEE Computer Society.

[11] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. Jha. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *ICCD-25*, October 2007.

[12] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha. Express virtual channels: Toward the ideal interconnection network. In *ISCA-34*, June 2007.

[13] J. Laudon and D. Lenoski. The SGI Origin: a ccNUMA highly scalable server. In *ISCA-24*, 1997.

[14] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 20890, Washington, DC, USA, 2004. IEEE Computer Society.

[15] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. Technical report, Microsoft Research, 2008.

[16] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *ISCA-36*, Washington, DC, USA, 2009. IEEE Computer Society.

[17] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *ISCA-31*, page 188, Washington, DC, USA, 2004. IEEE Computer Society.

[18] C. Nicopoulos, D. Park, J. Kim, V. Narayanan, M. S. Yousif, and C. Das. ViChaR: A dynamic virtual channel regulator for network-on-chip routers. In *MICRO-39*, pages 333–344, December 2006.

[19] E. Nilsson. Design and Implementation of a hot-potato Switch in a Network on Chip. Master's thesis, Royal Institute of Technology, IMIT/LECS 2002-11, Sweden, June 2002.

[20] E. Nilsson and J. Öberg. Reducing power and latency in 2-D mesh NoCs using globally pseudochronous locally synchronous clocking. In *CODES+ISSS-2*, pages 176–181, New York, NY, USA, 2004. ACM.

[21] C. Patel, S. Chai, S. Yalamanchili, and D. Schimmel. Power constrained design of multiprocessor interconnection networks. In *ICCD-16*, pages 408–416, Oct 1997.

[22] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA-7*, page 255, Washington, DC, USA, 2001. IEEE Computer Society.

[23] B. J. Smith. Architecture and applications of the hep mulitprocessor computer system. In *Readings in computer architecture*, pages 342–349. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[24] SPEC. SPEC benchmarks. http://www.spec.org.

[25] Y. Tamir and G. Frazier. High-performance multiqueue buffers for VLSI communication switches. In *ISCA-15*, pages 343–354, May-2 Jun 1988.

[26] S. Tota, M. R. Casu, and L. Macchiarulo. Implementation analysis of NoC: a MPSoC trace-driven approach. In *GLSVLSI-16*, pages 204–209, New York, NY, USA, 2006. ACM.

[27] TPC. TPC benchmarks. http://www.tpc.org.

[28] H. Wang, L.-S. Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *MICRO-36*, page 105, Washington, DC, USA, 2003. IEEE Computer Society.

[29] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *MICRO-35*, pages 294–305, 2002.

[30] S. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *ISCA-22*, June 1995.