The NoX Router

Mitchell Hayenga
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
hayenga@ece.wisc.edu

Mikko Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
mikko@ece.wisc.edu

ABSTRACT

Power efficient, low latency interconnects are increasingly important in a computing era dominated by growing core counts and diminishing power budgets. This paper proposes the use of a novel coding-based crossbar architecture to perform packet arbitration in parallel with switch traversal. The use of a lightweight exclusive-OR (XOR) coding scheme enables the productive transmission of packets, without waiting for arbitration, even under contention. For marginal cost compared to fully speculative techniques, switch arbitration latency can be hidden while eliminating power consuming misspeculations, increasing router throughput, and maintaining fairness.

The new NoX router is compared to traditional sequential and speculative single cycle router implementations on a 64-node CMP mesh. Physical implementation of all routers is modeled using synthesized RTL, detailed floorplans, and accurate channel models. Performance evaluation is carried out utilizing cycle-accurate simulation and detailed power models on both synthetic and application traffic. Overall we find the NoX architecture capable of bettering average packet energy-delay^2 product by 2.7%-34.4% on application workloads as well as improving network throughput by up to 9.9% on synthetic traffic patterns.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Performance

Keywords

On-chip networks, multi-core, routing, arbitration

MICRO 44 December 3-7, 2011, Porto Alegre, Brazil

1. INTRODUCTION

Networks on Chip (NoC) are now a reality within the microprocessor industry and are expected to grow in size and ubiquity in the coming years [2, 26]. As multi-core systems scale, the engineering challenges related to maintaining acceptable communication latencies and per-node bandwidth escalate due to the impacts of limited wire scaling and increasing arbitration overheads [7, 3]. To mitigate the communication limitations facing NoCs, researchers have proposed ever increasingly aggressive speculation techniques to avoid unnecessary arbitration latencies [23, 21, 22, 30, 12, 16]. With such techniques, the relative circuitry and power overheads resulting from misspeculations increase. Meanwhile, with growing core counts, bandwidth demands rise and average communication distances increase, lessening the effectiveness of aggressive speculation techniques.

While latency remains a key optimization objective for on chip networks, diminishing power budgets emphasize the need for power efficient communication substrates. Recent industry projections [3, 8] demonstrate that if left unchecked, the global interconnect could consume substantial power in future multi-core processors. Thus any latency reduction technique must carefully balance its respective performance benefit against any incurred power overhead.

In this paper we propose a novel router technique for achieving ultra-low latency while eliminating many of the energy overheads and scheduling inefficiencies faced by traditional speculative router architectures. This is accomplished through a lightweight coding scheme and new crossbar architecture that replaces the traditional multiplexer or tristate based crossbar with precomputed input gating and an exclusive-OR (XOR) based switch. This minor architecture modification, combined with decoding logic consisting of a single level of 2-input XOR gates, enables the elimination of switch arbitration latency while productively forwarding packets and immediately freeing buffer resources, even under contention. In contrast to speculative architectures [21, 22, 16] which incur wasted cycles and unproductive link transitions due to imperfect arbitration, the new coding-based Network of XORs (NoX) router architecture allows contention to potentially occur within the switch and, through an arbiter run in parallel, simply decides which input an encoded-form output corresponds to. Communicating encoded packets allows for input buffers to be immediately freed, alleviating buffer pressure and reducing the impacts of head-of-line blocking at the current router node. Decoding of encoded packets is achieved by simply XORing contiguous received packets.

Overall, we find that the improved switch efficiency and elimination of unproductive link transitions enables the NoX router to outperform traditional speculative and non-speculative router on an energy-delay product basis, despite the marginal circuitry cost associated with the XOR coding logic.

The major contributions of this paper are:

- We propose a new power efficient coding-based router arbitration technique which successfully hides switch arbitration latency.
- The detailed design and implementation of an on chip interconnection router incorporating this technique with full analysis of area, power, and frequency.
- A thorough network power and performance evaluation against traditional sequential and speculative routers on synthetic traffic as well as scientific and commercial application workloads.

The remainder of this paper is organized as follows: In Section 2, we motivate and define the coding-based arbitration technique used within the NoX architecture. This is followed with a detailed architectural overview of a router incorporating the new coding-based technique. In Section 3 we define multiple speculative and non-speculative routers and perform an in-depth architectural comparison against the NoX router. Section 4 details the evaluation methodology, followed by results in Section 5, physical implementation details in Section 6, related work in Section 7, and conclusions in Section 8.

2. THE NOX ARCHITECTURE

2.1 Motivation

The transition to an NoC design methodology provides computer architects the ability to optimize interconnection substrates in a modular, scalable, and easily-quantifiable manner [4]. As architects continue to refine NoC structures, enabling superior performance in the high bandwidth environment which they operate, multiple design trends have emerged. Interconnection datapaths have become increasingly simplified through the reduction or elimination of buffering resources and pipeline stages. At the same time, router control logic complexity increased as architects have added both functionality and aggressive speculation logic to optimize performance [6, 31, 14, 23, 21, 16].

Following from these trends, many single cycle routers have been proposed [21, 22, 12, 19, 16]. Although capable of transmitting packets in fewer cycles than comparable multi-stage routers, single cycle routers are forced to make sacrifices in terms of clock period or router efficiency in order to realize low cycle counts. The dominant trend in single cycle routers to date has been to favor aggressive speculation in order to hide arbitration latencies, while ceding router efficiency under high utilization or when speculation mechanisms fail.

The primary objective of the NoX architecture is to enable more efficient low latency routers which eliminate arbitration latency while maintaining router efficiency. Through the use of an innovative scheme that overlaps arbitration with a new coding-based XOR switch architecture, router and link efficiency is maintained, even under contention, by productively routing encoded packets. The following sections detail the primary mechanism which enables this as well as the key architectural components necessary for implementation.

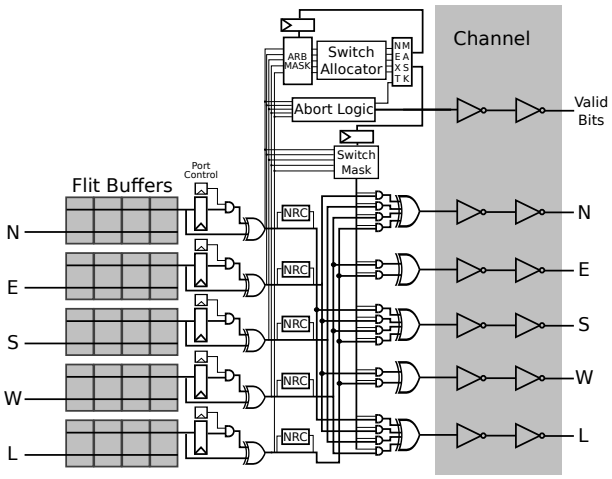

Figure 1: The NoX Architecture

2.2 Overview

The fundamental means that enables the NoX architecture to eliminate arbitration latency lies in its ability to successfully transmit packets without modification or in an encoded form depending on the presence of contention. As is shown in architectural Figure 1, this is achieved by replacing the traditional crossbar switch with an XOR-based circuit and input gating. In the absence of contention only one input will be active for a given output port causing the XOR-based switch to pass the given input unmodified. Conversely, in the presence of contention, the resulting output is the logical XOR of all competing inputs. In the event of contention, an arbiter run in parallel selects which input will not be necessary on subsequent cycles. On following cycles, new competing inputs are prevented by inhibition, and the output becomes the logical XOR of the remaining inputs which collided in the initial transfer cycle. This process of granting and removing individual inputs sequences the router output, such that a receiving router is able to decode all encoded packets by XORing subsequently received packets. Consequently, every cycle results in the productive transfer of information across the communication link.

Packets are able to be decoded because of the following property: If 3 inputs A, B, and C all contend on a given cycle the router produces the encoded output $(A \otimes B \otimes C)$; as one of these inputs will receive a grant, in the following cycle only B and C will be transmitted producing $(B \otimes C)$. The input absent from the second packet, which won arbitration at the previous router, is recreated by the following property $(A \otimes B \otimes C) \otimes (B \otimes C) = A$. Packets decoded by this means are received in the order which they won arbitration, maintaining any fairness or prioritization mechanisms within the network.

2.3 Detailed Timing

In this section we provide timing examples demonstrating the switch traversal and decoding processes used by the NoX router.

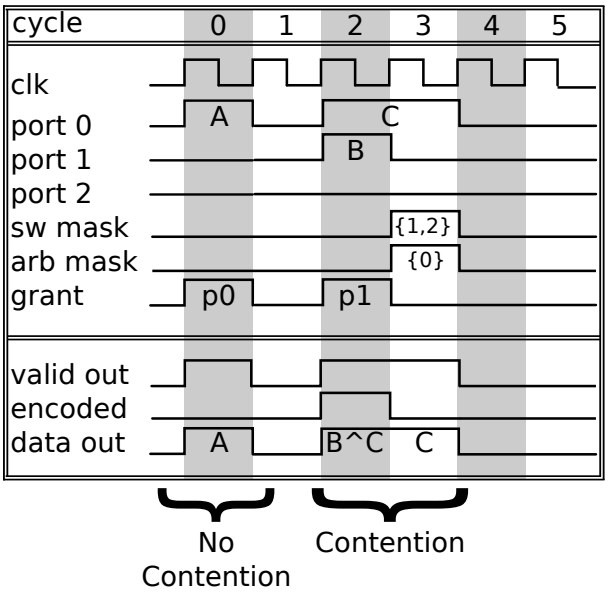

Figure 2: NoX Transmission Timing

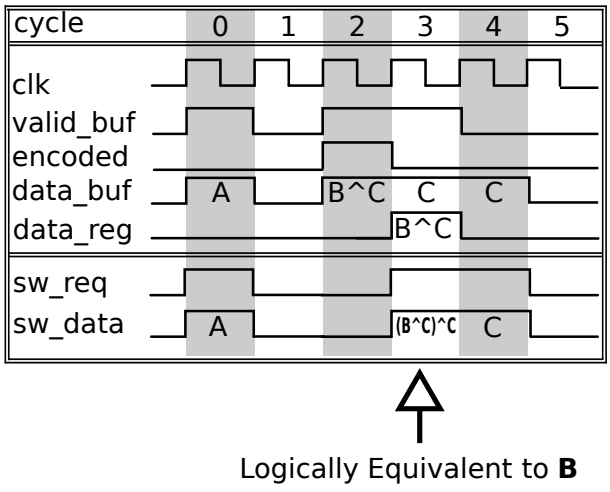

Figure 3: NoX Receive Timing

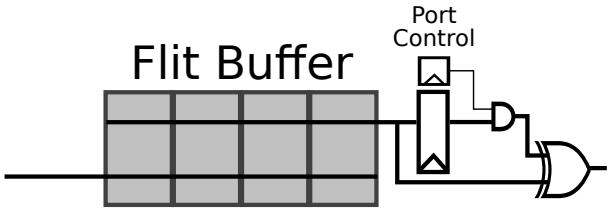

Figure 4: NoX Input Port

2.3.1 Transmission Timing

The timing diagram in Figure 2 provides a detailed description of the NoX router's switch operation in the presence and absence of contention. In this example, all inputs are assumed destined for the same output. With the absence of contention in cycle 0, packet A passes unmodified from *port 0* to the output. Although an arbitration decision was made in parallel with the traversal of A, it was unnecessary since no contention was present. An example of contention handling is given on cycle 2 when packets B and C arrive simultaneously on different inputs. The output during the cycle is the logical XOR of the conflicting inputs ($B \otimes C$). The output is also marked as encoded such that the receiving router properly decodes the encoded packet. In parallel with the production of $B \otimes C$, an arbitration decision is computed resulting in *port 1*, sender of B, receiving a grant. Due to contention, switch and arbiter request masks are computed for the following cycle using the contending requests and grant signal. Full details are provided in subsection 2.6, but in general switch and arbiter masks are set such that only those requests which conflicted but did not win the grant, are allowed to continue colliding within the switch. Finally, on the subsequent cycle, switch masks are set such that C is the only input allowed switch progression. As *port 0* was not inhibited and no contention occurred, its request is registered as serviced and ceases transmission.

2.3.2 Receive Timing

The timing diagram detailed in Figure 3 presents how input buffer logic handles the reception of uncoded and coded packets. The packets processed in this example correspond to the packets sent in Figure 2's example. Additionally, it is assumed that all switch requests are immediately granted once generated.

In cycle 0 packet A is read from the input buffer and immediately results in a switch request since it requires no decoding. Conversely, in cycle 2 coded packet ($B \otimes C$) is read from the input buffer. A switch request is not generated and instead the value ($B \otimes C$) is saved into a separate decoding register. On the following cycle, C is subsequently read and logically XORed with the decoding register, presenting ($B \otimes C) \otimes C$, logically equivalent to B, as a switch request. Finally in cycle 4, uncoded packet C is transmitted from the input buffer.

2.4 Input Port

The input port of the NoX router provides the necessary services of request buffering and decoding. As shown in Figure 4, the primary components of the input port are a single read, single write port SRAM FIFO, a single register, and XOR decoding circuitry. The SRAM FIFO implements basic wormhole buffering as is commonly used in many interconnection networks. The single register allows that, if necessary, any two consecutively received values can be XORed with the decoding circuitry.

The process of transmitting packets to the primary switch fabric is straightforward. If the packet read from the FIFO is not encoded, it immediately passes through the decoding circuitry without XORing the value from the register. Otherwise, if the buffered packet is encoded, it can not be immediately forwarded and is latched into the register. On the subsequent clock cycle, the saved value from the register and the consecutive packet read from the FIFO are XORed with the decoding circuitry, presenting the original

uncoded packet to the switch fabric. As the possible need for value latching is known early within the clock cycle, the decoding register can be clock gated in most circumstances. The input port does not consider its input granted until it is not inhibited by the primary switch logic and one of two conditions is met: 1) No contention for the desired output port was encountered or 2) Contention for the desired output port occurred, but it received the grant from the output arbiter.

2.5 Switch Logic

The switch logic, shown earlier in Figure 1, consists of inhibiting certain inputs and the XORing of uninhibited inputs. The switch requests are ignored if they require decoding or are presently inhibited by the precomputed switch mask. The NoX router switch primarily differs from a standard multiplexer crossbar switch by substituting XOR logic gates in place of multiplexers. The substitution of XOR gates causes multiple design costs and opportunities to arise. First, XOR logic gates have higher logical effort than comparable tristate based multiplexers, consuming marginally more power and delay. However, as observed within other works, router datapath timings have considerable slack as control logic comprises the critical path in speculative routers [22, 8]. Second, the XOR based switch does offer its own benefits with respect to layout and power. In a standard crossbar, control signals must be quickly routed across the switch fabric such that tristates or multiplexers select proper inputs for the respective outputs. The routing of control signals in a mux based design requires driving time critical signals over long capacitive wires that fan out to many gates. Conversely, in the NoX switch, inhibition signals can be locally computed at each port with pre-computed input masks and simple routing logic. Overall, as detailed in Section 5.3, we find the power of the XOR-based crossbar to closely match the conventional multiplexer-based baseline.

2.6 Output Arbitration and Masking

For each output, an arbiter circuit and two request masks exist to control allowed packet transmission sequences. Each control mask is used to inhibit requests to the arbitration and switch fabric respectively, effectively limiting the allowed collisions within the XOR based switch.

The output arbitration and masking logic performs its task by operating in one of two modes: *Recovery* and *Scheduled*. *Recovery* mode is a reactive mode whereby collisions within the switch can freely occur and the arbitration and masking logic takes necessary action to properly resolve contention. Conversely, *Scheduled* mode is fully pre-scheduled and results in uncontested outputs. In this section we describe how the NoX arbitration and masking logic can successfully operate within and transition between these modes.

In *Recovery* mode, switch and arbitration masks are identical and multiple inputs might contend within the XOR based switch for a given output. In this mode it is the responsibility of the output arbiter to select a winning input, notify the input port of successful transmission, and set the next arbitration and switch masks to only allow the other inputs which contended but lost arbitration. If the resulting switch and arbitration masks would inhibit all inputs or no grants are generated, the masks are instead set to enable all inputs once again. Conversely, if the resulting switch and arbitration masks would enable only a single input, the logic instead transitions into *Scheduled* mode and forces the arbitration mask to be the bitwise complement of the switch mask.

In *Scheduled* mode, the switch mask and arbitration masks are bitwise complements of each other. This results in only a single input being enabled for switch traversal, whereas all other inputs are enabled for arbitration. If an input is granted by the output arbiter, on the following clock cycle it will be the only input enabled for switch traversal but also the only input inhibited from arbitration. This has the impact of allowing a packet to progress through the switch during the current cycle, while using the arbitration logic to schedule the packet to be transmitted on the next clock cycle. If a grant is ever not generated, then the arbitration logic immediately transitions again to *Recovery* mode and enables all switch and arbitration inputs.

By revisiting the timing example listed earlier in Figure 2 the logical breakdown and application of these arbiter modes becomes clear. Initially output arbitration and masking logic begins in *Recovery* mode with masks to enable all inputs to arbitration and switch traversal logic. In cycle 0, a grant for *port 0* is generated, but masking all remaining inputs would result in all inputs being inhibited, thus the masks are set to enable all inputs. In cycle 2, two inputs collide within the switch. As the next switch enable mask would only allow a single input to be enabled, *input 0*, the arbiter transitions to *Scheduled* mode. If any new requests had arrived on cycle 3, they would have been presented to the arbiter allowing a request to be pre-scheduled in cycle 4. As no requests are presented to the arbiter though, resulting in no grants, the arbitration logic transitions back to the optimistic *Recovery* mode.

This structuring of the arbitration logic allows for the use of the XOR based switch to make productive use of unpredictable contention cycles while performing similarly to an aggressively speculative single cycle baseline router [21] when input requests can be non-speculatively pre-scheduled.

2.7 Multi-Flit Packet Support

As described thus far, the NoX architecture lacks proper support for multiple flit packets. In order to support multiple flit packets, multiple implementation alternatives are available. As done some in some architectures which support packet fragmentation [19], routing information could be appended each packet and no additional architecture modification would be necessary. Instead we take a more conventional approach and require that multiple flit packets be sent contiguously without fragmentation. This requirement introduces the possibility of aborts, though significantly less frequent than in purely speculative architectures as the majority of packets are single-flit control packets in cache coherent systems. Aborts are triggered upon any collisions involving multiple flit packets, and mask generation logic gracefully handles aborts by immediately transitioning to *Scheduled* mode and enforcing no other arbitration winners until the tail flit has passed.

2.8 Virtual Channel Support

Within this paper, all evaluated routers are wormhole architectures and thus do not support virtual channels. Instead, networks utilizing such routers must use multiple physical channels or higher level communication constructs to successfully avoid protocol level deadlock. Multiple works

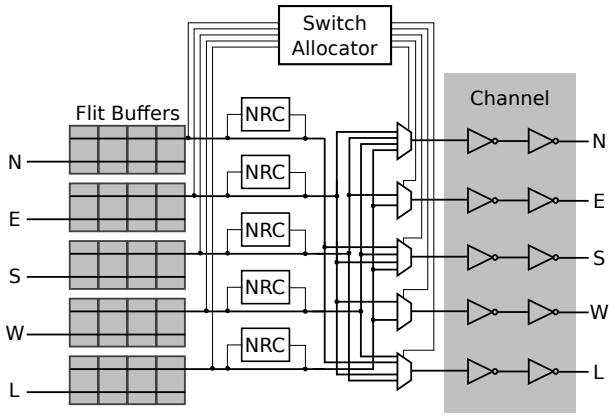

Figure 5: Non-Speculative Router Architecture

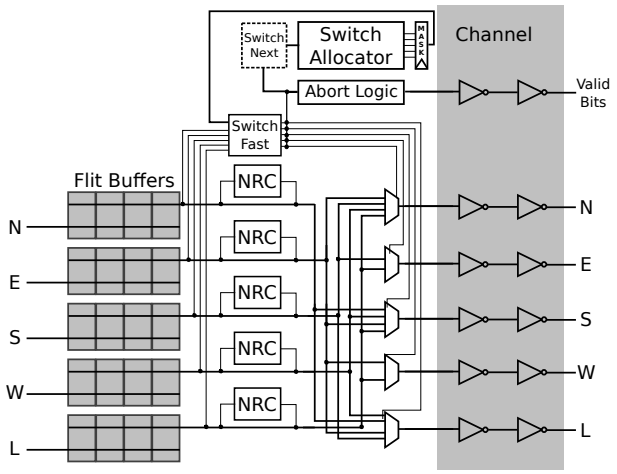

Figure 6: Speculative Router Architecture

have highlighted using multiple physical channels as a potentially more power efficient alternative to conventional virtual channel routers [1, 17, 27, 29]. Logically, support for virtual channels is largely orthogonal to the switch mechanism of the NoX router, but detailed exploration of virtual channel routers is left to future work.

3. ARCHITECTURE DISCUSSION

In this section we discuss the key performance and architectural differences between the NoX router and alternative router implementations. In order to form a basis for comparison, Section 3.1 details the implementation and general operation of multiple router architectures. To capture the impact of latency and performance efficiency trade-offs, we have chosen to define a non-speculative and two speculative single cycle routers. After defining these architectures that illustrate key points within the design space, the remainder of this section performs a qualitative comparison of all with respect to the NoX router.

3.1 Baseline Designs

3.1.1 Non-speculative Router

The architecture of our first baseline, a non-speculative router, is defined in Figure 5. A canonical router architecture, the non-speculative router results in a simple implementation consisting of only structures for input buffering, route computation, switch arbitration, and switch traversal. In our chosen implementation, look-ahead route computation (NRC) [5] is used to overlap route computation and switch arbitration latencies. As a fully non-speculative architecture, output efficiency is high with outputs capable of being active every cycle, regardless of internal router contention, provided sufficient downstream buffering. Resulting from its non-speculative nature, the router architecture incurs a substantial latency penalty in exchange for performance efficiency.

3.1.2 Speculative Routers

Representative of trade-offs frequently made between latency and efficiency, we implement two speculative router designs from [22]. Originally proposed for use within a virtual channel router, both designs have been adapted for wormhole router implementation. Fully detailed by Mullins et al. in [21] and [22], the key insight is that contention is infrequent at low utilization and so arbitration is typically unnecessary, as only one input is likely to active for a given output. Using this intuition, requests can be allowed to speculatively traverse the switch toward their respective outputs. In the event that this speculation is wrong and contention occurs for an output, router efficiency is compromised by stalling all contending packets. Resulting from contention and the single cycle nature of the router, during this time an indeterminate and invalid value is driven across the contended output channels. In order to guarantee forward progress in the event of contention, an arbiter is run in parallel with the speculative switch traversal. In the event speculation fails, the arbiter ensures only a single input is allowed undergo switch traversal in the following clock cycle.

The general architecture of both speculative routers is shown in Figure 6. The two router designs, *Spec-Fast* and *Spec-Accurate*, differ primarily in implementation by the functional modification of the *Switch Next* logic block. The *Switch Next* logic block determines which requests are presented to the allocator. In the *Spec-Fast* design, the *Switch Next* logic block simply passes all requests not masked by the *Switch Fast* logic to the allocator, potentially resulting in unnecessary switch reservations on the proceeding clock cycle. In the *Spec-Accurate* design, the *Switch Next* logic block is passed the same requests as *Switch Fast* logic block and removes requests that successfully undergo switch traversal in the current cycle, resulting in a more accurate switch scheduling. The differing requests passed to *Switch Next* from *Switch Fast* results from properly supporting wormhole flow control without packet fragmentation in each architecture. As the *Spec-Fast* architecture is designed to attain the minimal clock period at all costs, multiple flits from the same packet are guaranteed contiguous transmission by simply masking all other requests from arbitration. Though to ensure fairness given this wormhole mechanism, rather than limit abort logic as done in [22], newly exposed packets on an input port may not request arbitration in the *Spec-Fast* architecture. The *Spec-Accurate* architecture, like the NoX and non-speculative routers, ensures wormhole flow control by incurring marginal logic overhead to override arbitration

if a multi-flit packet is under transmission on a given output port. Overall, our implementation of the *Spec-Accurate* is designed as a compromise between the efficiency of the non-speculative router and latency of the *Spec-Fast* router.

3.2 Performance Efficiency

Although all implemented designs are single cycle routers, it is important to understand relative impact contention can have on their performance and power efficiency. Figure 7 presents example timings of each baseline router architecture in the presence and absence of contention. All inputs are assumed to be destined for the same output port. The input stimuli to these examples is identical to that used to demonstrate the NoX architecture in Figures 2 and 3 from Section 2.3. As expected, in the absence of contention all routers are capable of successfully routing packet *A* from input *port 0* without delay on clock cycle 0.

In the presence of contention, however, performance can vary significantly. As shown at the beginning of cycle 2 when multiple contending inputs arrive simultaneously, the non-speculative and NoX router architectures both productively forward a packet and free a buffer from input *port 1*. Although the packet may or may not experience an additional cycle of delay at the downstream router of the NoX network depending on later contention. Conversely, both speculative routers *Spec-Fast* and *Spec-Accurate* incur a wasted clock cycle due to contention within the switch. Additionally, both architectures waste power by driving the output channel with an indeterminate and invalid value. This overhead is significant since, as observed in multiple works, the interconnection channel is the most energy consuming component and frequently accounts for over half of all network energy [1, 17, 18]. In cycle 3, the NoX and non-speculative architectures again perform similarly transferring a packet which had been delayed one cycle due to contention. For the speculative routers, both are now capable of transmitting packet *B* having pre-scheduled it a cycle earlier in cycle 2. The performance of the speculative routers differs though in the transmission of the final packet. The *Spec-Accurate* architecture successfully transmits the final packet in the following cycle, whereas the *Spec-Fast* router incurs an additional wasted cycle due to its low latency, but inefficient mechanism for supporting potential multi-flit packets.

Overall, we have shown that, though clock periods may differ, on a cycle utilization and energy basis that the non-speculative router is the most efficient, followed in succession by the NoX, *Spec-Accurate*, and *Spec-Fast* routers. Later in Section 6 we provide exact details on the cycle time penalties incurred by each architecture.

4. METHODOLOGY

Performance, power and area results are obtained through a combination of cycle-accurate C++ router simulation, verilog synthesis using Synopsys Design Compiler, manual floorplanning, and analytical energy and delay models. Channel delay and energy estimation models from [1] and [20] are utilized along with extracted parameters from a target TSMC 65nm standard cell library to determine necessary channel repeater sizing and spacing. Input buffer SRAM area, power, and timing parameters are computed through memory compiler generation and SPICE model extraction. Overall router layout is computed using a floorplan simi-

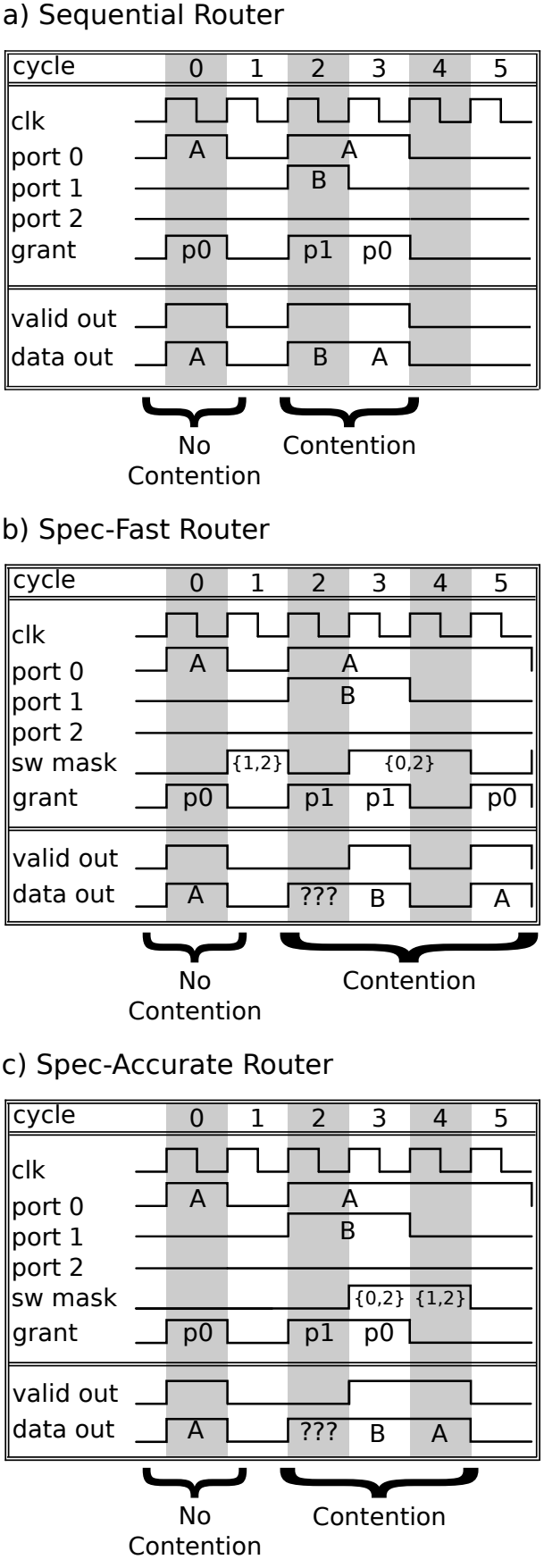

Figure 7: Baseline Router Timings

lar to that of Balfour et al. [1]. Crossbar switch area, latency, and power is computed using manual floorplanning, standard cell selection, and wiring estimation. Taking into account output driver capacitance and switch wiring capacitance by annotating specific nets, verilog synthesis is then performed for the remaining router logic structures. Estimated frequency parameters come from this verilog delay coupled with timing models for the other structures. Finally, with energy and delay models for all router components, a cycle-accurate C++ simulation model is complemented with necessary event counters to form an accurate power model.

A 64-node, 8x8 mesh network with 2mm 64-bit interconnection links is assumed in all simulations. All routers contain four 64-bit buffers per input port, the minimal necessary to cover the round trip credit loop. For application network simulation a second physical network is used to isolate classes of coherence traffic necessary for deadlock prevention. Further details on synthetic and application performance modeling can be found in Section 5. For all evaluations we assume each interconnection network operates at its maximum frequency asynchronous with respect to processor tiles [9].

5. ROUTER PERFORMANCE

To analyze the impacts of router architecture on latency and energy efficiency under different performance constraints, both synthetic and application-level traffic analyses are performed. For all all traffic scenarios, both latency and energy-delay^2 performance numbers are provided.

5.1 Synthetic Performance

For synthetic traffic evaluation a 64-core, 64-bit mesh interconnect is evaluated using standard single-flit traffic patterns [4]. Additionally we utilize a self similar pareto-based traffic pattern commonly used in networking evaluations and occasionally used in on chip evaluations [15, 6, 11]. Network traffic has been widely shown to follow a self similar distribution, having a random bursty nature. The self similar traffic was generated using $\alpha = 1.4$, $b = 8$ and varying T_{off} to obtain desired injection rates.

Figure 8 presents average latencies versus injection bandwidth and Figure 9 reports energy-delay^2 product relative to injection bandwidth. Properly accounting for cycle time differences amongst the architectures, Figure 8 reports all latency results in terms of nanoseconds and injection rates in terms of megabytes per second per node (MB/s/node). From the latency results, we observe that at low injection bandwidths the two speculative and NoX routers offer approximately equal performance with marginal differences due to their respective clock periods. In Figure 8a for uniform random traffic, the *Spec-Fast* architecture offers superior latency performance up to an injection rate of 575 MB/s/node, followed by the *Spec-Accurate* architecture offering superior performance on injection bandwidths ranging from 575 MB/s/node to 750 MB/s/node. For all injection rates above 750 MB/s/node until saturation at 2775 MB/s/node the NoX architecture offers superior performance. This relative ranking holds for most traffic patterns in Figure 8. With higher injection bandwidths, the misspeculation inefficiencies grow and the two speculative router architectures cede performance relative to the NoX and non-speculative architectures. The *Spec-Fast* architecture, though having the highest clock frequency, suffers the most and frequently saturates at less than half the bandwidth as the other router architectures. On traffic patterns such as uniform random and self similar traffic, where there is frequently path variation between consecutively injected packets, the NoX architecture's reduction in head-of-line blocking, by its ability to immediately free buffers under contention, extends maximum throughput significantly. On other traffic patterns which offer little to no path variation between consecutive packets within the network, the relative efficiency benefit of the NoX architecture approach is normally sufficient to compensate for the marginal clock penalty its decoding hardware entails.

Parameter	Value
Cores	64
Topology	8x8 mesh
Processor	3GHz in order PowerPC
L1 I/D Caches	32KB, 2-way set associative
L2 Cache	256KB, 8-way set associative
Cache Line Size	64-bytes
Memory Latency	100 cycles
Interconnect	64-bit request, 64-bit reply network
Packet Sizes	8 byte control, 72 byte data
Buffer Depth	4 64-bit entries/port
Channel Length	2mm
Routing Algorithm	Dimension Ordered Routing

Table 1: Common System Parameters

Figure 9 shows the energy-delay^2 performance of all router architectures on the synthetic traffic patterns. In general, existing trends present earlier in Figure 8 are amplified and the marginal energy benefits of the NoX and non-speculative routers in comparison to the *Spec-Fast* and *Spec-Accurate* architectures enhance their performance, although latency differences account for most variation.

5.2 Application Performance

To measure the performance of each router on realistic traffic patterns, multiple scientific and commercial application traces were run on a cache coherent CMP network with 2 64-bit physical wormhole networks [28, 24, 25]. Full simulation system parameters are provided in Table 1. Processor tiles are assumed to operate on their own frequency domain asynchronously with respect to the interconnection network, allowing each network design to operate at its maximum frequency, similar to recent industry many-core chips [9]. Processor packet events are injected into the interconnection network on their corresponding network clock cycles, keeping CPU injection bandwidth constant across all interconnection networks.

Packet latency results shown in Figure 10, demonstrate the NoX architecture to be the optimal network given our application workloads injection bandwidth requirements and traffic patterns. The *Spec-Fast* architecture is also shown to be overly aggressive for these workloads, as its performance efficiency is hurt enough that the non-speculative router is able to outperform it despite a large clock period disadvantage. It should also be noted that these latency results are conservative due to our trace-based methodology and the self-throttling nature of interconnection networks. As the *Spec-Accurate* and NoX router architectures greatly out-

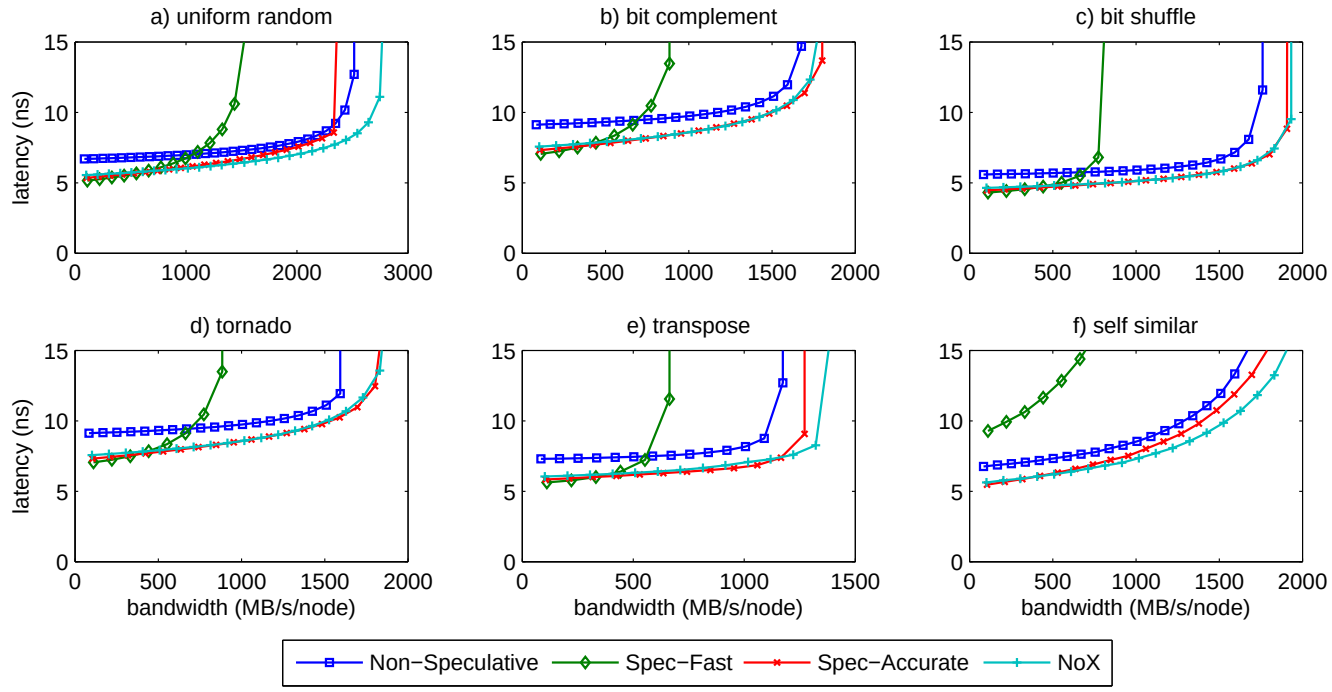

Figure 8: Synthetic traffic latency

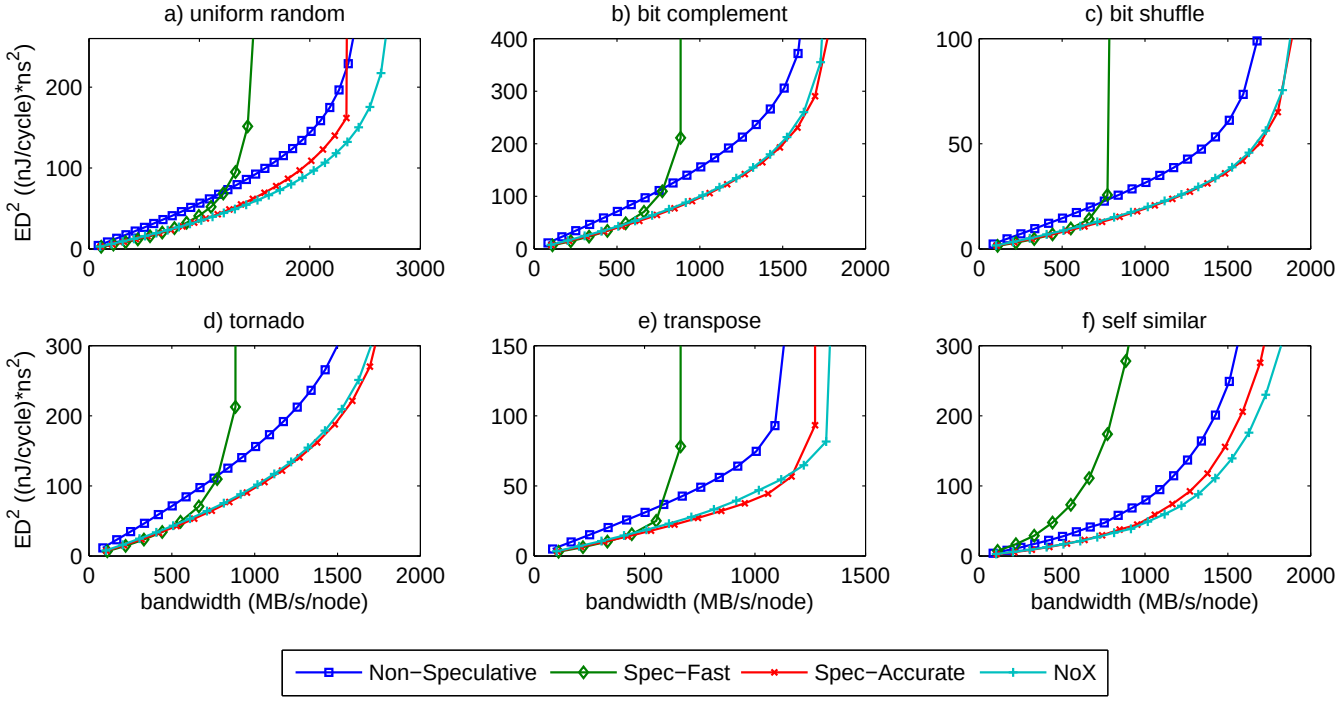

Figure 9: Synthetic traffic energy-delay^2 performance

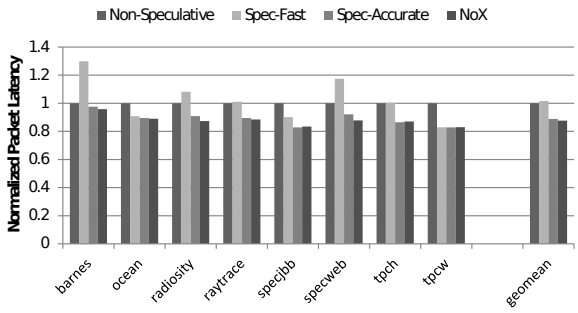

Figure 10: Application average packet latency

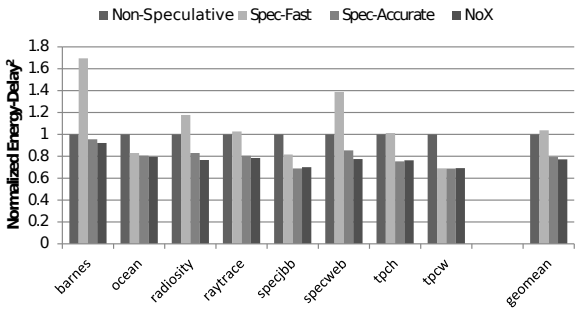

Figure 11: Application energy-delay^2 performance

perform the *Spec-Fast* and non-speculative architectures in terms of latency, allowing network feedback would result in higher contention favoring the NoX router.

Figure 11 shows average packet energy-delay^2 performance for each network on the application traffic traces. The NoX architecture extends its lead due to its latency and energy efficiency advantage over the comparable *Spec-Accurate* network architecture. On average the NoX architecture, outperforms the non-speculative, *Spec-Fast*, and *Spec-Accurate* by 29.5%, 34.4%, and 2.7% respectively on an energy-delay^2 basis.

5.3 Power Analysis

Figure 12 shows total dynamic power for a 64-node network under a 2 GB/s/node single-flit uniform traffic load. The *Spec-Fast* architecture is not shown due to its low saturation bandwidth. The non-speculative architecture consumes the least energy due to its minimal hardware and lack of speculation, although it experiences very high packet latencies. Comparing the *Spec-Accurate* and NoX architectures illustrates their power tradeoffs. The *Spec-Accurate* architecture consumes 4.6% more link energy, but 2.4% less switch energy than the NoX architecture. As link power dominates the power profile, consuming approximately 74% of all router power, the *Spec-Accurate* architecture draws 2.5% more overall power than the NoX architecture. Energy costs associated with packet decoding in the NoX architecture are also found to be minimal. It should be noted that although power numbers are comparable due to approximate equal amounts of work being performed, average packet latencies vary significantly under this workload.

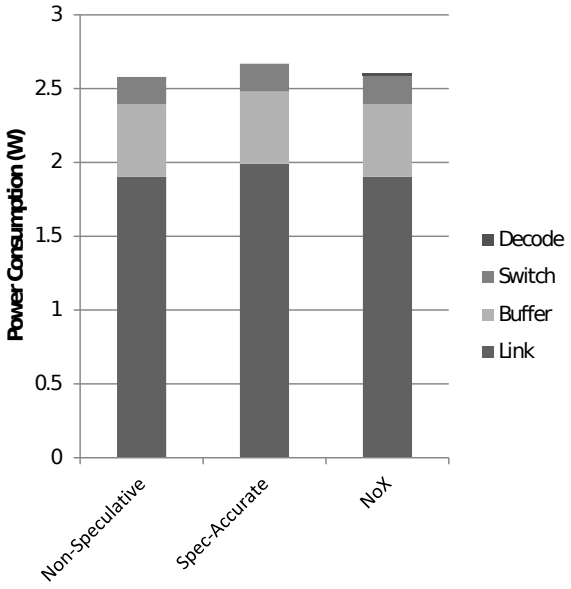

Figure 12: Total network dynamic power consumption for 2GB/s/node single-flit uniform random traffic

Architecture	Clock Period
Non-Speculative	0.92 ns
Spec-Fast	0.69 ns
Spec-Accurate	0.72 ns
NoX	0.76 ns

Table 2: Router Clock Periods

6. ROUTER IMPLEMENTATION

Detailed earlier in Section 4, substantial effort was placed into accurately modeling router energy, delay, and area overheads through a combination of analytical models, manual floorplanning, and synthesis. In this section we document the obtained clock periods and area overheads incurred by each router.

6.1 Frequency Results

Obtained clock period results for each router is presented in Table 2. All router latencies include a 248ps SRAM delay and 98ps link latency for the 2mm interconnection channel between adjacent tiles. As seen by the clock period difference between the *Spec-Accurate* and NoX architectures, decoding logic in the NoX architecture incurs approximatly 40ps of overhead. Relative to the non-speculative architecture, the *Spec-Fast*, *Spec-Accurate*, and NoX architectures are 33.3%, 27.8%, and 21.1% faster on a clock period basis.

6.2 Area Floorplanning

Area floorplans for both the conventional and NoX router architectures are shown in Figure 13. The general floorplan layout was adapted from [1]. Input SRAM buffers are stacked horizontally, assuming bit interleaving. SRAM areas were determined from memory compiler generation. The NoX architecture incurs 28.2μm additional horizontal length

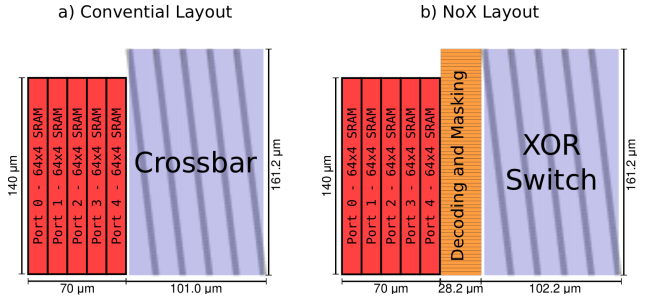

Figure 13: Router Floorplaning

due to the necessary decoding and masking hardware. For both floorplans, crossbar switch height is determined by our standard cell height of 2.52μm whereas crossbar width is determined by wire spacing. Allocation, abort, and route computation logic is not shown as it takes marginal area and does not impact total router area, as it fits comfortably within the upper left of the router floorplan. Overall, the total NoX router tile incurs a 17.2% area penalty for the inclusion of decoding and masking logic. We find this an acceptable overhead given its energy and performance benefits over comparable routers.

7. RELATED WORK

Extensive numbers of router microarchitectures and techniques exist within the field of NoCs that strive to eliminate arbitration overheads. The methods for arbitration elimination vary widely, including speculative techniques, non-speculative techniques, and alternative network topologies.

Over the years, speculative arbitration techniques have grown more aggressive. Peh and Dally [23] founded the use of speculation in on chip virtual channel routers for the purposes of overlapping virtual channel arbitration and switch arbitration latencies. Mullins et al. [21, 22], extended the use of speculation to enable speculative switch traversal. Kumar et al. [12] introduced a heavily optimized single cycle router capable of achieving single cycle latency in the absence of contention through the use of an optimized control datapath. Finally, the most aggressive speculative routers to date [30, 16], propose predictively routing packets out predetermined outputs without waiting for route computation, thus potentially routing packets out incorrect output ports.

Amongst non-speculative techniques, alternative topologies and router bypassing are quite common. Express channels [4] and their virtual equivalents [13], eliminate arbitration overheads by enabling router bypassing for multi-hop packets. Additionally, multiple alternative topologies attempt to capitalize on low latency, on chip wires for arbitration reduction [1, 10].

8. CONCLUSION

In this paper we propose the NoX router, a novel single cycle router architecture designed for energy efficient low latency operation. Through the use of a new XOR-based switch and input gating, switch arbitration latency can be hidden while productively routing packets, even under contention. With the development of detailed energy and delay models, the NoX architecture has been shown to improve average packet energy-delay^2 product over traditional router architectures by 29.5%, 34.4%, and 2.7% respectively as well as improving throughput by up to 9.9% on synthetic traffic patters over all evaluated speculative and non-speculative routers. In future work, we look to evaluate the NoX architecture on alternative, higher radix, topologies [1] which may derive more benefit given their higher arbitration latencies, their longer channels, and the fixed cost of the NoX decoding hardware.

9. ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation award CCF-1116450.

10. REFERENCES

[1] J. Balfour and W. J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 187–198, New York, NY, USA, 2006. ACM.

[2] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C.-C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 - processor: A 64-core soc with mesh interconnect. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 88 –598, feb. 2008.

[3] S. Borkar. Networks for multi-core chips: A contrarian view. In *Special Session at ISPLED*, 2007.

[4] W. J. Dally and B. Towles. Route packets, not wires: On-chip inteconnection networks. In *Proceedings of the 38th annual Design Automation Conference*, DAC '01, pages 684–689, New York, NY, USA, 2001. ACM.

[5] M. Galles. Scalable pipelined interconnect for distributed endpoint routing: the sgi spider chip. In *IEEE Hot Interconnects*, 1996.

[6] P. Gratz, B. Grot, and S. W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *International Symposium on High-Performance Computer Architecture*, pages 203–214, 2008.

[7] R. Ho, K. Mai, and M. Horowitz. Managing wire scaling: a circuit perspective. In *Interconnect Technology Conference, 2003. Proceedings of the IEEE 2003 International*, pages 177 – 179, june 2003.

[8] 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.

[9] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *Solid-State Circuits, IEEE Journal of*, 46(1):173 –183, jan. 2011.

[10] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. *Microarchitecture, IEEE/ACM International Symposium on*, 0:172–182, 2007.

[11] G. Kramer. On generating self-similar traffic using pseudo-pareto distribution,. *http://wwwcsif.cs.ucdavis.edu/kramer/papers/self sim.pdf*, 2011.

[12] A. Kumar, P. Kundu, A. P. Singh, L. shiuan Peh, and N. K. Jha. A 4.6tbits/s 3.6ghz single-cycle noc router with a novel switch allocator. In *in 65nm CMOS,âĂİ ICCD-2007*, 2007.

[13] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha. Express virtual channels: towards the ideal interconnection fabric. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 150–161, New York, NY, USA, 2007. ACM.

[14] M. M. Lee, J. Kim, D. Abts, M. Marty, and J. W. Lee. Probabilistic distance-based arbitration: Providing equality of service for many-core cmps. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 509–519, Washington, DC, USA, 2010. IEEE Computer Society.

[15] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *Networking, IEEE/ACM Transactions on*, 2(1):1 –15, feb 1994.

[16] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction router: A low-latency on-chip router architecture with multiple predictors. *Computers, IEEE Transactions on*, 60(6):783 –799, june 2011.

[17] G. Michelogiannakis, J. Balfour, and W. Dally. Elastic-buffer flow control for on-chip networks. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 151 –162, feb. 2009.

[18] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis. Evaluating bufferless flow control for on-chip networks. In *Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '10, pages 9–16, Washington, DC, USA, 2010. IEEE Computer Society.

[19] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 196–207, New York, NY, USA, 2009. ACM.

[20] M. L. Mui, K. Banerjee, and A. Mehrotra. A global interconnect optimization scheme for nanometer scale vlsi with implications for latency, bandwidth, and power dissipation. *Electron Devices, IEEE Transactions on*, 51(2):195 – 203, feb. 2004.

[21] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st annual international symposium on Computer architecture*, ISCA '04, pages 188–, Washington, DC, USA, 2004. IEEE Computer Society.

[22] R. Mullins, A. West, and S. Moore. The design and implementation of a low-latency on-chip network. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, ASP-DAC '06, pages 164–169, Piscataway, NJ, USA, 2006. IEEE Press.

[23] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, HPCA '01, pages 255–, Washington, DC, USA, 2001. IEEE Computer Society.

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

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

[26] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(1):29 –41, jan. 2008.

[27] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal. On-chip interconnection architecture of the tile processor. *Micro, IEEE*, 27(5):15 –31, sept.-oct. 2007.

[28] S. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the International Symposium on Computer Architecture*, June 1995.

[29] Y. J. Yoon, N. Concer, M. Petracca, and L. Carloni. Virtual channels vs. multiple physical networks: a comparative analysis. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 162–165, New York, NY, USA, 2010. ACM.

[30] T. Yoshinaga, S. Kamakura, and M. Koibuchi. Predictive switching in 2-d torus routers. In *Innovative Architecture for Future Generation High Performance Processors and Systems, 2006. IWIA '06. International Workshop on*, pages 65 –72, jan. 2006.

[31] G. L. Yuan, A. Bakhoda, and T. M. Aamodt. Complexity effective memory access scheduling for many-core accelerator architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 34–44, New York, NY, USA, 2009. ACM.