Appears in Proceedings of the 23rd International Symposium on High-Performance Computer Architecture (HPCA)⋆

Near-Ideal Networks-on-Chip for Servers

Pejman Lotfi-Kamran[§], Mehdi Modarressi[†§], and Hamid Sarbazi-Azad[‡§]
[§]*School of Computer Science, Institute for Research in Fundamental Sciences (IPM)*
[†]*School of Electrical and Computer Engineering, College of Engineering, University of Tehran*
[‡]*Department of Computer Engineering, Sharif University of Technology*

***Abstract*—Server workloads benefit from execution on many-core processors due to their massive request-level parallelism. A key characteristic of server workloads is the large instruction footprints. While a shared last-level cache (LLC) captures the footprints, it necessitates a low-latency network-on-chip (NOC) to minimize the core stall time on accesses serviced by the LLC. As strict quality-of-service requirements preclude the use of lean cores in server processors, we observe that even state-of-the-art single-cycle multi-hop NOCs are far from ideal because they impose significant NOC-induced delays on the LLC access latency, and diminish performance.**

Most of the NOC delay is due to per-hop resource allocation. In this paper, we take advantage of proactive resource allocation (PRA) to eliminate per-hop resource allocation time in single-cycle multi-hop networks to reach a near-ideal network for servers. PRA is undertaken during (1) the time interval in which it is known that LLC has the requested data, but the data is not yet ready, and (2) the time interval in which a packet is stalled in a router because the required resources are dedicated to another packet. Through detailed evaluation targeting a 64-core processor and a set of server workloads, we show that our proposal improves system performance by 12% over the state-of-the-art single-cycle multi-hop mesh NOC.

***Keywords*-Latency; network-on-chip; resource allocation; server**

I. Introduction

Server workloads are sensitive to last-level cache (LLC) access latency because of their large instruction footprint [1], [2]. Prior research shows that server workloads lose as much as half of the potential performance due to long latency LLC hits [3]. A noticeable fraction of LLC access latency is due to on-chip communications [1] — a request for a piece of data or an instruction should be sent to a destined LLC slice and the response should be sent back to the requesting core.

A common network-on-chip (NOC) in today's many-core processors is a two-dimensional mesh. It has been shown that a mesh-based fabric leads to poor performance on server workloads [1], [4]. The performance in mesh-based designs suffers as a result of a large average hop count, each hop involving a router traversal.

To reduce NOC latency, researchers have proposed single-cycle multi-hop networks [5], [6]. Such networks benefit from the fact that wires are relatively fast, and as such, in a single clock cycle, a packet can pass over more than one hop. Reducing the number of hops, a single-cycle multi-hop network improves performance over a mesh-based design by accelerating accesses to the LLC.

However, server workloads have strict quality-of-service requirements, so they require relatively fat cores with high clock frequency [7]. Large cores increase the link length between two adjacent hops, and high clock frequency reduces the time budget for link traversal. Together, these two factors limit the effectiveness of single-cycle multi-hop networks in reducing the number of hops for server processors. Consequently, even single-cycle multi-hop networks impose significant router delay on the LLC access latency.

Most of the per-hop delay is due to resource allocation. On arriving at a router, a packet's flit first needs to allocate the required resources and then use the allocated resources to go to the next hop. To reduce per-hop delay, prior work proposed allocating the resources to a flit a few cycles before its arrival at a router [8]. Using this concept, we propose proactive resource allocation (PRA) to eliminate the resource allocation delay of a single-cycle multi-hop network.

In contrast to prior work [8] that pre-allocates resources on a per-flit basis, PRA pre-allocates resources for the whole packet to avoid flit reordering in a single-cycle multi-hop network [5]. With PRA, packets may pass up to a few hops (e.g., two) in a single cycle. PRA is effective because it takes advantage of two opportunities to allocate resources to packets ahead of time on the way downstream to the destination: (1) the period between the end of tag and data lookup in the LLC [9], and (2) the in-network blocking period in which requested resources are not free.

When an LLC slice receives a request for a piece of data or an instruction, if the request turns into a hit in the LLC, there will be a response to the requesting core. Last-level caches of most processors benefit from a serial tag and data lookup to reduce energy usage [10], [11]. For such LLCs, the whole data lookup time is available for PRA. In cases when LLC uses a parallel tag and data lookup, data lookup takes longer than the tag lookup, as the data array is much larger, and the time between the end of the tag and data lookup is available for proactive resource allocation.

Moreover, if a packet (either a request or a response) is waiting in a router because the output port is busy forwarding a multi-flit packet, PRA benefits from the waiting time by proactively allocating the required resources for the waiting packet on the way downstream to the destination.

We make the observation that if the downstream router has enough buffers to store the in-transfer packet that holds the requested resources, we can determine exactly when the transmission of the in-transfer packet will end and consequently when the waiting packet can be transmitted.

In this paper, we make the following contributions:

- To the best of our knowledge, this is the first work that shows that single-cycle multi-hop networks are far from ideal for server processors.
- We show that pre-allocating resources to packets (and not individual flits) in a single-cycle multi-hop fashion within the two suggested time intervals results in a near-ideal network for servers, which is within 4% of the performance of an ideal network.
- To the best of our knowledge, this is the first time that packet stall time in a router is used for pre-allocating resources in the network.
- We use a full-system simulation infrastructure to evaluate PRA in the context of a 64-core server processor on a set of server workloads. Our results show that PRA offers 12% higher performance as compared to a single-cycle multi-hop network.

II. Background

In this section, we examine features of server workloads, and then describe trends in many-core processors. Last, we survey on-die interconnect schemes and describe their implications for performance in the context of many-core server processors.

A. Server Workloads

Server workloads have several features that are common across a wide range of applications, such as web search and media streaming [2], [3], [7]. Three of the common features are (1) request independence, (2) quality-of-service (QoS) requirements, and (3) sensitivity to LLC access latency. We examine these three common features in the following sections.

Request Independence: Users' requests that are processed by server workloads are mostly mutually independent. The independence of requests makes server workloads a good candidate for execution on many-core processors.

Quality-of-Service (QoS) Requirements: Many server workloads (e.g., web search and media streaming) have latency requirements as part of their service-level agreement. Moreover, server workloads increasingly invoke computationally intensive and performance-critical kernels [2], [7]. As lean cores (tiny cores with low frequency, e.g., [12]) may jeopardize application quality-of-service and latency constraints, they are not commonly used in server processors.

Sensitivity to LLC Access Latency: Server workloads have large instruction footprints beyond what can be captured in L1-I caches [1], [2]. Consequently, last-level caches hold the instruction footprints. As a result, server workloads are sensitive to LLC access latency.

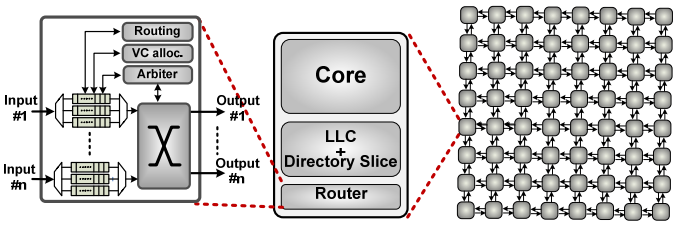

Figure 1. Elements of tiled server processors.

B. Server Processors

The observations captured in the previous section are reflected in several contemporary server processors. One such design is the Intel Xeon E5 series processors. Depending on the model, the E5 series features up to 12 cores, a banked LLC with 6-30 MB of storage capacity, and a ring interconnect for connecting cores and cache banks. While appropriate for a modest number of cores, the ring interconnect stands as a major obstacle for scaling up the core count, as its delay has linear dependence on the number of interconnected components.

To overcome the scalability limitations of ring-based designs, emerging many-core processors, such as Intel Knights Landing [13], use a tiled organization. Figure 1 shows an overview of a generic tiled processor. Each tile consists of a core, one bank of the distributed last-level cache, directory slice, and a router. The tiles are linked via a routed, packet-based, multi-hop interconnect in a mesh topology.

C. NOC Architecture

Even mesh-based designs expose the core-LLC communications to significant network delays, and diminish performance. Each hop in a mesh network involves a router traversal, which adds delay due to the need to access the packet buffers, arbitrate for resources, and navigate the switch. These delays diminish the performance of a mesh-based tiled processor on server workloads [4].

To overcome the performance drawbacks of mesh-based interconnects, researchers developed a single-cycle multi-hop network named SMART [5] for on-die communications. SMART takes advantage of a dedicated multi-drop network to set up multi-hop paths. When a header flit wins the arbitration, instead of sending the flit to the link in the next cycle, SMART attempts to establish a multi-hop path using the multi-drop network, which takes one cycle. In the next cycle, the header flit goes to the link, potentially passing over multiple hops (e.g., eight hops) before getting latched in the input buffer of a router. SMART reduces the contributions of routers to the end-to-end delay at the expense of an additional clock cycle delay to set up a multi-hop path.

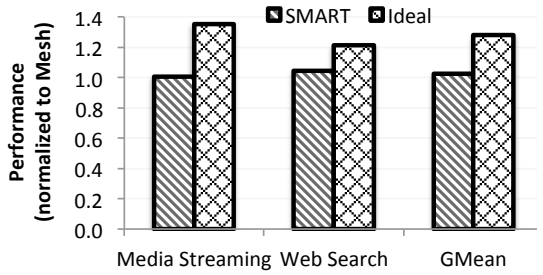

Figure 2. Performance of SMART and ideal NOCs, normalized to mesh.

Unfortunately, SMART does not offer a significant performance boost in the context of server processors. As cores in server processors are relatively fat with high clock frequency, a single-cycle multi-hop NOC can send a packet over just a few hops in a single cycle (e.g., two hops). Given the extra cycle needed to set up a multi-hop path and the probability that not all links are idle at the same time, the net effect of SMART in server processors is negligible.

Figure 2 compares the performance of the SMART network to that of an ideal network with zero router latency (link delay and contention are accounted for) for two representative server workloads. Both performance numbers are normalized to that of a mesh network. Packets may pass over two hops in a single cycle in both SMART and ideal networks. The details of the methodology can be found in Section IV. The results show that the performance of SMART is almost the same as that of the mesh interconnect. Moreover, a hypothetical network that does not impose router delay on the end-to-end packet latency results in an average 28% performance improvement for Media Streaming and Web Search workloads as compared to the mesh.

In summary, to get good performance from server workloads, we need to minimize the contributions of routers to the on-die communication delay in a single-cycle multi-hop network.

III. Our Proposal

This work aims to eliminate resource allocation time from the end-to-end packet transmission latency. For this purpose, on the way downstream to the destination, we proactively allocate resources to packets before they demand the resources. To proactively allocate resources, we need to know what resources are needed for transmission of a packet and in which timeslots. The former can be determined by the destination of a packet, as we know the whole path to the destination. The latter can be calculated if one knows when the packet starts passing through the network. Knowing the starting time, it is easy to calculate when the packet enters and exits each hop, given the assumption that proactive resource allocation is successful in prior hops.

We need to know the required information (i.e., starting time and destination) a few cycles before a packet starts traveling in the network. We observe that under two frequent events, the required information is known a few cycles before the actual packet transmission: (1) upon an LLC hit, and (2) when a packet is stalled because the requested output port is busy sending a multi-flit packet.

Depending on the available time, the distance between the node that initiates PRA and the destination, and the status of the required resources, PRA allocates part or even all of the required resources to the destination. For part of the path to the destination where resources are proactively allocated, the packet just uses the resources, which greatly speeds up the transfer, and for other parts, the packet first allocates the resources, as in a standard network, before using them.

Proactive resource allocation is a general idea and can be implemented on any NOC. As a case study, we provide details for the implementation of PRA on a standard mesh network.

A. PRA on a Mesh Network

In server processors, networks need to have three message classes—*request*, *response*, and *coherence*—to avoid protocol deadlock [14]. L1 caches are effective at filtering access to the network in server processors, so the traffic in the network is not heavy [4], [15].[1] Consequently, each message class usually consists of a single virtual channel (VC). Moreover, as the coherence traffic is negligible [4], [16], [17], the request and response traffic determines the performance.

The mesh network, with slight modifications to support PRA, is used for packet transmission. We refer to the mesh network as the *data network*. We augment the data network with a narrow bufferless *control network* to proactively allocate resources in the data network. In the following sections, we first explain the modifications needed in the data network for PRA and then discuss the control network in detail.

B. Data Network

Without PRA, the data network (i.e., the standard mesh network) does not support single-cycle multi-hop traversal. A header flit first goes to the VC and crossbar allocation, and if the required resources are allocated to it, goes to the crossbar and link in the following cycle. The rest of the flits follow the head flit in subsequent cycles. With PRA, however, resources are pre-allocated to packets so that packets can benefit from single-cycle multi-hop traversal. We assign resources to packets on a cycle-by-cycle basis. When a cycle in a router is assigned to a packet, all the necessary resources in the router are assigned to that packet. Moreover, for multi-flit packets, PRA either allocates the necessary resources for the transmission of all flits or fails.

[1]While the traffic is moderate in server workloads, most of the NOC traffic is due to instruction misses, and so the latency of the network significantly affects performance [2].

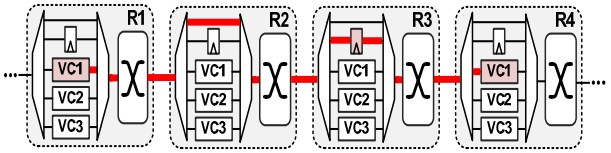

Figure 3. A 2-cycle proactively allocated path from R1 to R4.

Figure 3 shows a proactively allocated path between R1 and R4. Assuming two hops can be passed in a single cycle, the 3-hop path consists of two single-cycle traversals of length 2 and 1. The second part of the path is shorter because either R4 is the final packet destination or the required resources are not idle at R4. In router R1 and at time *t*, a packet is read from VC_1's buffer, and is passed through the crossbar and R1–R2 link. In router R2 and at the same cycle, the mux and demux are set so that whatever comes out of the R1–R2 link goes to the crossbar and R2–R3 link. So the packet bypasses R2 and goes directly toward R3, where the demux is set so that the packet goes to the latch. The latches are used for temporary storage of flits after passing over n hops, where n is the number of hops that a flit can pass over in a single cycle (while PRA only supports two-hops-per-cycle traversal, the data network is general). In the following cycle (i.e., *t+1*), the packet similarly goes to R4 and is buffered in VC_1. As resource allocation is performed proactively, the whole transfer time from R1 to R4 is two cycles.

Figure 4 shows components of a router in the data network. Components that are unique to or modified by PRA are shaded in gray. One of the modified components is the input unit. In addition to standard VCs, we need to add two extra VCs to the input units for operation of PRA. One VC is a bypass link that lets packets bypass the VC buffers and go directly to the crossbar. The other added VC is a latch that is used as a temporary 1-cycle storage within a proactively allocated path.

Moreover, for each output port there is a set of bit vectors that store the allocation status of the output port for several timeslots starting from the next cycle. For each cycle, the bit vectors indicate whether resource allocation is done for that cycle (*Valid* vector in Figure 4). If the resource allocation is done, the bit vectors also indicate the input port that the packet comes from (*Input Select*), the exact VC within the port (*Local VC Select*), and the exact VC of the downstream router to which the packet should go (*Downstream VC Select*). The content of the bit vectors is shifted to the left each cycle and one free timeslot becomes available in the last cycle. The control network sets the bit vectors, as explained in Section III-C.

The arbiter is one of the components that are slightly modified to support proactive resource allocation. The arbiter consists of a local arbiter, as in a standard mesh network, and a PRA arbiter. If the bit vectors indicate that no proactive

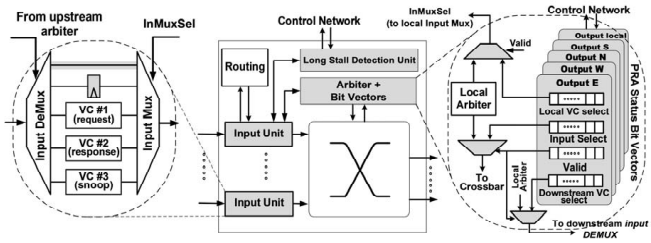

Figure 4. Mesh+PRA router in the data network.

resource allocation is recorded for a given timeslot, the local arbiter will be in charge; otherwise, the PRA arbiter decides what will happen in the cycle. Figure 4 shows how local and PRA arbiters are connected together. The PRA arbiter uses the bit vectors to decide what to do in each cycle. If there is no recorded resource allocation for the next cycle, it does nothing. Otherwise, based on the bit vectors, it sets the *select* of the mux to connect the right VC to the crossbar, sets the control signal of the crossbar to send the packet to the output port, and sets the *select* of the demux of the downstream router to guide the packet to the right VC.

Finally, the data network includes a *Long Stall Detection (LSD)* unit that checks for a stalled packet waiting for the end of transmission of a multi-flit packet. If there are enough buffers in the downstream router for the multi-flit packet, this unit injects a control packet into the control network to proactively allocate resources for the stalled packet.

C. Control Network

The control network is a narrow, bufferless NOC that is used for proactive resource allocation in the data network. A control packet pre-allocates resources in the data network to accelerate packet transmission. If a control packet cannot pre-allocate resources in a router, it simply gets dropped. A control network consists of a mesh of single-cycle multi-drop segments, as shown in Figure 5. The figure highlights the internal structure and connections of a control network's router in the X dimension. The router has the same connections in the Y dimension.

Each router is connected to the next two routers in each direction using a multi-drop segment. Turns are not allowed in multi-drop segments as a way of minimizing the overhead. With multi-drop segments, when a router sends a control packet, two subsequent routers receive the packet. We use 2-hop multi-drop segments to enable control packets to pass over two hops in two cycles: one cycle for packet processing and one for packet transmission.

A router has two multi-drop inputs per direction. For each direction, there are three latches: two corresponding to the two multi-drop inputs and one associated to the LSD, as we will discuss shortly. The three latches are statically prioritized such that the closest multi-drop segment has the

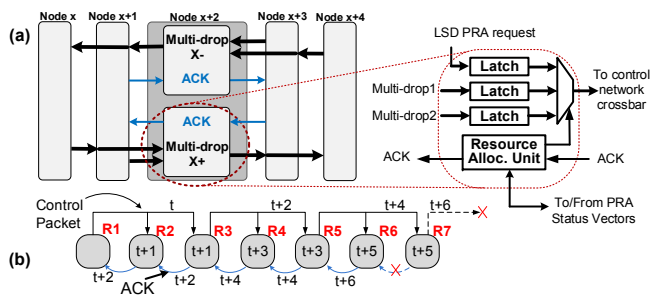

Figure 5. Control network: (a) links and signals and (b) signaling for sending a control packet from R1 to R7.

highest and the LSD has the lowest priority. PRA prioritizes the closest multi-drop segment over the farthest one to enable the second node of a 2-hop multi-drop to locally determine whether the first node can proactively allocate the required resources or not, as we explain in this section. If a router receives more than one control packet in a direction in a single cycle, the lower priority packets are dropped.

Control packets, which are one flit long, consist of the destination address, the lag between the control and the data packet (number of cycles), the size of the data packet (long or short), the message class (VC number), and look-ahead routing information. The destination is either the source field of the request packet or the destination field of the stalled packet, and the lag is the number of cycles between the control and its corresponding data packet.

On arriving at the LLC, a request packet is queued within the LLC and waits for its lookup time. If the tag lookup indicates a hit, the LLC controller will notify the network interface (NI). The NI creates a control packet and places it in the local latch of the control network if the latch is empty. Otherwise, the control packet will be dropped.

Moreover, the long stall detection (LSD) unit checks to see if a packet is waiting for a multi-flit packet, and if there are enough buffers for transmission of the multi-flit packet. If the conditions hold, the LSD unit generates a control packet and injects it into the control network to proactively allocate resources for the waiting packet.

On receiving a control packet in a given direction, the packet is passed through the route computation unit and resource allocation unit in parallel. If there is more than one control packet in a direction, one is statically chosen and the rest are dropped. The resource allocation unit determines whether the requested timeslots on the requested output port and downstream VC buffers can be granted. Note that the control network always allocates buffers for a full packet. If the packet needs to pass the bypass VC or the latch of Figure 4, the decision is updated later, as we discuss. If the request cannot be granted, the control packet will be dropped. Otherwise, the granted timeslot(s) will be recorded in the bit vectors and the required buffer space in the downstream router is allocated properly for the full packet.

If the two routers in a multi-drop can allocate the required resources, the second router forwards the control packet to the subsequent multi-drop segment, provided that the control packet's lag is greater than zero. As it takes two cycles for the control packet to pass over a multi-drop segment while it takes only one cycle for the corresponding data packet to pass over the pre-allocated multi-hop path, routers decrement the lag to account for this difference and drop control packets when the lag becomes zero. With the lag being zero, the data packet has reached the control packet and no further pre-allocation is possible.

As a control router does not know if proactive resource allocation will succeed in the downstream router, it should reserve a downstream buffer for the packet in the data network. Each control router that successfully pre-allocates the required resources passes an ACK signal back to the upstream router. The ACK signal notifies the upstream router that it is not the last node of a pre-allocated path. Upon receiving the ACK signal, the control router frees up the allocated buffer space and changes the *Down Stream VC Select* (see Figure 4) for the data packet to pass through the latch or bypass VC that are included in the input unit of the data network. If the router is the second router in the multi-drop segment, the latch VC will be selected (Router 3 in Figure 5(b)); otherwise the bypass link (Router 2 in Figure 5(b)) is selected.

While only the second node in a multi-drop segment is responsible for transmitting the control packet to the next multi-drop segment, the transmission should happen only if the two nodes in the multi-drop segment can pre-allocate the required resources. Fortunately, the second node knows whether the first node can allocate the required resources or not, because the second node knows the status of its input port and buffer, which are the output port and downstream buffer of the first node, and PRA gives higher priority to the closest multi-drop segment, as mentioned earlier.

Finally, PRA needs to avoid the possibility of two multi-flit packets, one with normal and one with proactive resource allocation, getting interleaved in buffers of a standard VC. When a router allocates timeslots to a multi-flit packet on an output port, it sets a special flag corresponding to the message class. While this flag is set, no multi-flit packet can use the message class, but single-flit packets can still use the message class. The flag is cleared when either the multi-flit packet passes over the output port or the downstream router, through the ACK signal, informs the router that the multi-flit packet will not be using the VC buffer.

IV. Methodology

Table I summarizes the key elements of our methodology, and the following sections detail the evaluated designs, technology parameters, workloads, and simulator.

Table I
EVALUATION PARAMETERS.

Parameter	Value
Technology	32 nm, 0.9 V, 2 GHz
Processor features	64 cores, 8 MB NUCA LLC, Four DDR3-1600 memory channels
Core	ARM Cortex-A15-like: 3-way out-of-order, 64-entry ROB, 16-entry LSQ, 2.9 mm^2, 1.05 W
Cache	per MB: 3.2 mm^2, 500 mW
NOC Organizations:	
Mesh	Router: 5 ports, 3 VCs/port, 5 flits/VC, 1-stage (speculative) pipeline. Link: 1 cycle
SMART	Router: 5 ports, 3 VCs/port, 5 flits/VC, 2-stage pipeline. Link: up to 2 tiles per cycle
Mesh+PRA	Data network: 5 ports/router, 3 VCs/port Packets with PRA support: Bypassing pipeline stages, Link: 2 tiles per cycle Others: 1-stage speculative pipeline. Link: 1 tile per cycle Control network: 4-output and 13-input ports/router, 1-stage bufferless pipeline. Link: 2 tiles per cycle
Ideal	Router: 5 ports, 3 VCs/port, 5 flits/VC, Bypassing pipeline stages. Link: 2 tiles per cycle

A. Processor Parameters

Our target is a 64-core processor based on the Scale-Out Processor design methodology [18], [19], which seeks to maximize throughput per die area. The chip features a modestly sized last-level cache to capture the instruction footprint and shared OS data, and dedicates the rest of the die area to the cores to maximize throughput. The architectural features are listed in Table I.

We consider four system organizations, as follows:

Mesh: Our baseline is a mesh-based tiled processor, as shown in Figure 1. The 64 tiles are organized as an 8-by-8 grid, with each tile containing a core, a slice of the LLC, and a directory node. A mesh hop consists of a single-cycle crossbar and link traversal followed by a one-stage router pipeline for a total of two cycles per hop at zero load. The router performs routing, VC allocation, and speculative crossbar (XB) allocation in the first cycle, followed by XB and link traversal in the next cycle. Each router port has three VCs to guarantee deadlock freedom across three message classes: request, coherence, and response. Each VC is five flits deep, which is the minimum needed to cover the round-trip credit time.

SMART: The SMART-based processor has the same tiled organization as the mesh baseline, but enjoys single-cycle multi-hop traversal. A SMART hop consists of a two-stage router pipeline followed by a single-cycle (potentially) multi-tile link traversal for a total of three cycles per hop at zero load. The router performs routing, VC allocation, and speculative crossbar (XB) allocation in the first cycle, a multi-tile link allocation in the second cycle, and finally XB and link traversal. Each router port has three VCs to guarantee deadlock freedom across three message classes. Each VC is five flits deep.

Mesh+PRA: The proposed proactive resource allocation (PRA) is implemented on top of the baseline mesh. LLC waiting time and in-network blocking time are used to proactively allocate resources using a dedicated bufferless control network with 15-bit-wide links. At the control-network, a hop consists of a single-stage router pipeline followed by a single-cycle two-tile multi-drop link traversal. At the data network, a Mesh+PRA hop consists of a single-stage router pipeline followed by a single-cycle single-tile traversal for a total of two cycles per hop at zero load without PRA (i.e., baseline mesh). However, when proactive resource allocation takes place, a packet passes over up to two tiles (crossbars and links) in a single cycle. Each router port has three VCs to guarantee deadlock freedom across three message classes. Each VC is five flits deep.

Ideal: The ideal interconnection network is a hypothetical network-on-chip with router delay of zero cycles. For the ideal network-on-chip, only wire delays are considered. A header flit can pass over up to two hops in a single cycle if the required crossbars and links are free. Body flits follow the header flit in subsequent cycles. While router delay is zero, packets may get blocked in a router due to contention (e.g., two packets competing for the same output port). Each router port has three VCs for request, coherence, and response packets. Each VC is five flits deep.

B. Technology Parameters

We use publicly available tools and data to estimate the area and energy of the various network organizations. Our study targets a 32 nm technology node with an on-die voltage of 0.9 V and a 2 GHz operating frequency.

We use custom wire models, derived from a combination of sources [20], [21], to model links and router switch fabrics. For links, we model semi-global wires with a pitch of 200 nm and power-delay-optimized repeaters. For SMART and Mesh+PRA, we choose the number of repeaters to get a link latency of 85 ps/mm. Given the delay of wires and the aspect ratio of the tiles, two tiles can be traversed in a single clock cycle. On random data, links dissipate 50 fJ/bit/mm, with repeaters responsible for 19% of link energy. For area estimates, we assume that link wires are routed over logic or SRAM and do not contribute to network area; however, repeater area is accounted for in the evaluation.

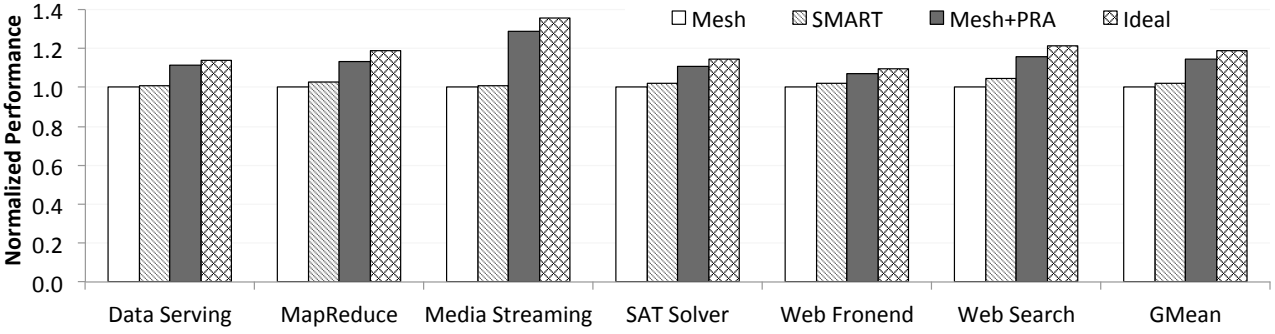

Figure 6. System performance, normalized to a mesh-based design.

Our buffer models are taken from DSENT [22]. We model flip-flop based buffers as all NOCs have relatively few buffers. Cache area, energy, and delay parameters are derived via CACTI 6.5 [23]. A 1 MB slice of the LLC has an area of 3.2 mm^2 and dissipates 500 mW of power (mostly leakage). The tag and data lookups take 1 and 4 cycles, respectively.

Finally, parameters for the ARM Cortex-A15 core are borrowed from Microprocessor Report [24] and scaled down from the 40 nm technology node to the 32 nm target. Core area, including L1 caches, is estimated at 2.9 mm^2. Core power is 1.05 W at 2 GHz. Core features include 3-way decode/issue/commit, 64-entry ROB, and 16-entry LSQ.

C. Workloads

We use server workloads from CloudSuite [25]. The workloads include Data Serving, MapReduce, Media Streaming, SAT Solver, Web Frontend, and Web Search. Two of the workloads—SAT Solver and MapReduce—are batch, while the rest are latency-sensitive and tuned to meet the response time objectives. Prior work [2] has shown that these workloads have characteristics representative of the broad class of server workloads.

D. Simulation Infrastructure

We estimate the performance of various processor designs using the Flexus full-system simulation [26]. Flexus extends the Virtutech Simics functional simulator with timing models of cores, caches, on-chip protocol controllers, and interconnect. Flexus models the SPARC v9 ISA and is able to run unmodified operating systems and applications. Flexus uses the BookSim 2.0 network simulator [27] for modeling the on-chip network.

We use the SimFlex multiprocessor sampling methodology [28]. Our samples are drawn over an interval of 10 seconds (except Media Streaming samples, which are drawn over 30 seconds) of simulated time. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and run 100 K cycles to achieve a steady state of detailed cycle-accurate simulation before collecting measurements for the subsequent 50 K cycles. We use the ratio of the number of application instructions to the total number of cycles (including the cycles spent executing operating system code) to measure performance; this metric has been shown to accurately reflect overall system throughput of multiprocessors [28]. Performance measurements are computed with 95% confidence and an error of less than 4%.

V. Evaluation

We first examine system performance and area efficiency of the Mesh, SMART, and Mesh+PRA designs, given a 128-bit link bandwidth. We then present an area-normalized performance comparison, followed by a discussion of power trends.

A. System Performance

Figure 6 shows full system performance, normalized to the mesh, for various NOC organizations. Mesh and SMART offer almost the same performance. SMART enables packets to go over two hops in a single cycle but, unlike Mesh, requires an extra cycle to set up the multi-hop path. As Figure 6 shows, the net effect is negligible.

The proposed Mesh+PRA offers the highest performance when compared to realistic networks. Compared to Mesh, Mesh+PRA improves performance by 7–29%, with a geomean of 14%. On average, the proposed Mesh+PRA improves performance over SMART by 12%. The highest performance gain is registered on the Media Streaming workload, which is characterized by very low instruction-level parallelism (ILP) and memory-level parallelism (MLP), making it particularly sensitive to the LLC access latency.

As a point of reference, we also include the performance of an ideal network with zero router latency. Across all benchmarks, the proposed Mesh+PRA closely follows the performance of the ideal network. On average, Mesh+PRA is only 4% behind the performance of the ideal network with zero router latency.

B. Why is PRA Effective?

To demonstrate why PRA is capable of reducing the on-chip network's delay, and consequently, improving system

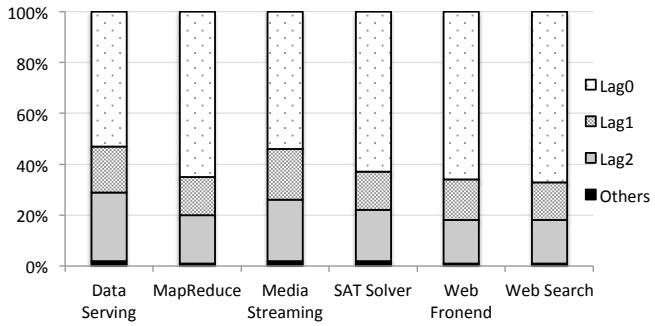

Figure 7. Distribution of control packets' lags when they are dropped. The maximum lag in our setup is four.

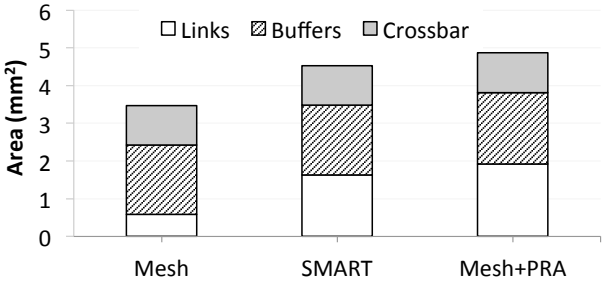

Figure 8. NOC area breakdown.

performance, it is essential to investigate how effectively control packets proactively allocate resources for the data packets traveling in the network. Figure 7 shows the distribution of control packets' lags when they are dropped. Because a control packet's lag is decremented in each multi-drop, the lower the lag becomes, the more resource allocation is done for the corresponding data packet (ideally the lag becomes zero before the control packet is dropped). Figure 7 shows that across all workloads, 53–67% of control packets have a lag of zero (ideal case) when they are dropped (the average across all workloads is 61%). Moreover, 15–20% of the control packets have a lag of one, and 17–27% have a lag of two. More than 98% of the control packets have a lag of 0–2 across all workloads (less than 2% of control packets have a lag of greater than two: the maximum lag in our setup is four). The results clearly show the effectiveness of control packets in pre-allocating resources to the data packets in the on-chip network.

Moreover, we measure the number of control packets that are injected into the control network. On average, we have 1.60 (SAT Solver) to 1.89 (Data Serving) control packets for a single data packet (either a request or a response). As there is more than one control packet per data packet and control packets are effective at proactive resource allocation (see Figure 7), a considerable number of a data packet's required resources is proactively allocated on the way downstream to the destination.

Finally, when resources are proactively allocated on an output port to be used later by a packet, the output port becomes unusable by multi-flit packets (short packets, i.e., requests, can still use the output port) until either the allocated resources are released or an ACK signal is received from the downstream router. This resource underutilization may have a negative impact on the effectiveness of the proposed resource allocation scheme. We measure the number of cycles that an output port cannot be used by a packet because the port is proactively allocated to another packet, and normalize it to the time the packet travels in the network. Across all workloads, a packet only spends 0.01% of the end-to-end latency waiting in the network because the resources are proactively allocated to other packets. The large number of control packets per data packet, the effectiveness of control packets in allocating resources, and the negligible impact of resource underutilization for server workloads explain the performance improvements observed in Figure 6.

C. NOC Area

Figure 8 breaks down the NOC area of the three organizations by links, buffers, and crossbars. Only repeaters are accounted for in link area, as wires are routed over tiles.

For SMART and Mesh+PRA, the area of the interconnect is 4.5 mm^2 and 4.9 mm^2, respectively. Compared to the Mesh, SMART and Mesh+PRA require 31% and 40% more area, respectively. Because interconnect has a small footprint (i.e., Mesh's footprint is 3.5 mm^2), the 1.0 mm^2 and 1.4 mm^2 area overheads of SMART and Mesh+PRA seem considerable, but as compared to the area of the whole chip (i.e., over 200 mm^2), they are relatively small.

D. Performance-Density Comparison

The area analysis in the previous section indicates different NOC area costs (and hence chip area) for the examined networks. To better understand how well the various designs use the chip silicon area, we assess the performance density (i.e., performance per square millimeter) of various processors. We only consider the area of cores, caches, and interconnect, disregarding the area of memory channels and IO devices.

Figure 9 summarizes the results of the study, with performance density of the four organizations normalized to that of the mesh (we idealistically assume the area of a mesh for the area of the ideal network). On realistic designs, Mesh+PRA offers the highest performance density, followed by SMART. The lowest performance density is registered for Mesh. While Mesh+PRA has the highest network area, due to its effectiveness in boosting performance and relatively low area overhead at the chip level, it is the most area-efficient organization. Mesh+PRA boosts performance density by 14% over Mesh, and 12% over SMART. Mesh+PRA is only 5% behind the performance density of the ideal network.

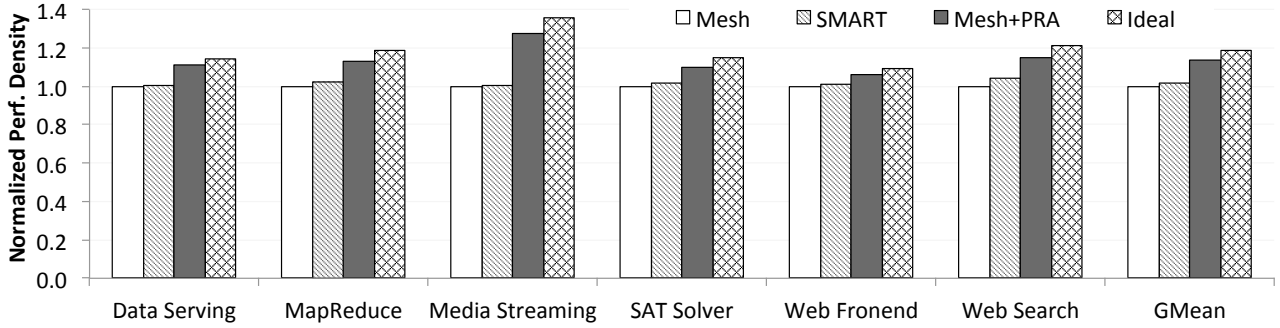

Figure 9. System performance per square millimeter (i.e., performance density), normalized to a mesh-based design.

E. Power Analysis

Our analysis shows that the NOC is not a significant consumer of power at the chip level (corroborating prior work [4], [29]). For all organizations, NOC power is below 2 W. In contrast, cores alone consume in excess of 60 W. Low ILP and MLP of server workloads [2] is the main reason for the low power consumption at the NOC level.

VI. Related Work

Various proposals have pointed out the need for low-latency on-chip communication mechanisms [30], [31]. Existing low-latency NOC designs often target reducing (1) hop counts, (2) blocking latency, or (3) per-hop latency.

Hop-count reduction. Packet hop-count reduction has long been a major target in many low-latency NOC designs. Prior work has focused on low-diameter topologies, including high-radix networks [32]–[34], reconfigurable networks [35], and mesh-based topologies equipped with extra irregular links that are inserted either randomly [36] or based on applications' traffic patterns [37]. Core-to-network mapping and customized topology generation [38] are also effective application-specific methods that reduce average hop count for a target application, when the application and its traffic pattern can be pre-characterized at design time.

Blocking-latency reduction. Adaptive routing is a technique to reduce blocking latency by directing packets to less congested paths. Among adaptive routing schemes, those methods that leverage both local and global congestion metrics [39]–[43] or are aware of the running applications' traffic behavior [44] often make more appropriate routing decisions. Moreover, arbitration plays an important role in managing the inevitable blocking latency in favor of total application performance and quality of service requirements. Slack-based arbitration [45] and prioritization schemes such as QoS-aware prioritization [46], application-aware prioritization [47], and message class-based prioritization [48] are effective in increasing applications' performance. Prior work also showed the effectiveness of predictive switch allocation [49], packet-chained allocation [50], run-time adaptive buffer sizing [51], smart VC allocation [52], packet compression [53], and heterogeneous router design [54], [55] in NOC latency reduction.

Per-hop latency reduction. To decrease router latency, efforts seek to cut down or bypass the pipeline stages of routers. A single-stage router [56] utilizes extensive pre-computation techniques to forward packets in a single cycle under low traffic. Router bypassing, which is implemented in prior work (such as Express Virtual Channels [57], Token Flow Control [58], and Pseudo Circuits [59]), enables flits to travel one hop per cycle on pre-established paths.

In flit-reservation flow control [8], a control packet traverses the network ahead of data flits to reserve buffers and channel bandwidth in advance. Each control packet leads one or multiple flits of a packet. Unlike PRA, this method does not support single-cycle multi-hop traversal, and reserves resources for individual flits, which makes it difficult to support single-cycle multi-hop traversal (e.g., flits may be reordered [5]). Bufferless NOCs cut down router pipeline stages by always forwarding received packets to an output port in a single cycle [60], [61]. Most bufferless methods enable single-cycle packet forwarding, but at the price of deflecting or dropping the packet when the preferred output port is busy, hence increasing network latency under moderate traffic loads.

The design of efficient circuit-switched NOCs has been the focus of many proposals [62], [63]. Traversing dedicated paths, circuit-switched data need not go through buffering, routing, arbitration, and flow control once circuits are set up. However, this switching method often suffers from performance degradation due to long circuit setup delay and poor bandwidth utilization. The time-division multiplexing (TDM) scheme mitigates the low bandwidth utilization of circuit switching [63], but its complexity introduces difficulties using the circuits. We use the concept of the timeslot in allocating link bandwidth to packets, but (1) avoid the long setup time of circuit switching by overlapping resource allocation and packet waiting time, and (2) relax the complex timeslot allocation and alignment of TDM by storing packets locally in case of unsuccessful allocation. Proactive circuit-switching [62] is a recent effort to hide long

circuit setup time by having request packets reserve circuits for their anticipated response packets as they go toward the destination. Although pre-allocation reduces the latency for those packets that travel on circuits, it requires multiple NOC planes. In addition, early reservation of circuits results in underutilization of network bandwidth. PRA pre-allocates resources for the exact packet transmission time; hence bandwidth loss is minimized. Likewise, a circuit-switched memory access NOC [9], called CIMA, pre-establishes circuits for long response packets. Routers attached to caches set up circuits a few cycles before actual response data transmission. Unlike PRA, CIMA establishes circuits only for response packets, does not use in-network packet stall time to set up circuits, and does not benefit from multi-hop forwarding. NOC-Out benefits from high-radix flattened butterfly topology to reduce hop count and low latency simple routers to reduce per-hop latency [4].

VII. Conclusion

Server processors require a fast fabric for core-LLC communications in order to maximize performance. Due to strict quality-of-service requirements, lean cores are not usually used for execution of server applications. Consequently, even state-of-the-art single-cycle multi-hop on-chip networks impose significant delays on the core-LLC communications.

This work identifies resource allocation as the major obstacle to a fast on-chip network for server processors that use single-cycle multi-hop networks. To eliminate this obstacle, this work takes advantage of (1) LLC data lookup time, and (2) packet blocking time to proactively allocate resources to packets. Experimental evaluation indicates that our proposal improves system performance over the state-of-the-art single-cycle multi-hop network by 12%.

Acknowledgment

The authors would like to thank Mohammad Sadrosadati for his help on the wire-delay analysis, Abbas Mazloumi for his help implementing NOCs in Booksim, and anonymous reviewers for their valuable comments and suggestions.

References

[1] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2009, pp. 184–195.

[2] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," in *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Mar. 2012, pp. 37–48.

[3] N. Hardavellas, I. Pandis, R. Johnson, N. G. Mancheril, A. Ailamaki, and B. Falsafi, "Database Servers on Chip Multiprocessors: Limitations and Opportunities," in *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR)*, Jan. 2007, pp. 79–87.

[4] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "NOC-Out: Microarchitecting a Scale-Out Processor," in *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2012, pp. 177–187.

[5] T. Krishna, C.-H. O. Chen, W. C. Kwon, and L.-S. Peh, "Breaking the On-chip Latency Barrier Using SMART," in *Proceedings of the 19th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2013, pp. 378–389.

[6] C.-H. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L.-S. Peh, "SMART: A Single-cycle Reconfigurable NoC for SoC Applications," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, Mar. 2013, pp. 338–343.

[7] V. J. Reddi, B. C. Lee, T. Chilimbi, and K. Vaid, "Web Search Using Mobile Cores: Quantifying and Mitigating the Price of Efficiency," in *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2010, pp. 314–325.

[8] L.-S. Peh and W. J. Dally, "Flit-Reservation Flow Control," in *Proceedings of the 6th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Jan. 2000, pp. 73–84.

[9] P. Lotfi-Kamran, M. Modarressi, and H. Sarbazi-Azad, "An Efficient Hybrid-Switched Network-on-Chip for Chip Multiprocessors," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1656–1662, May 2016.

[10] D. Sanchez and C. Kozyrakis, "The ZCache: Decoupling Ways and Associativity," in *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2010, pp. 187–198.

[11] M. Ferdman, P. Lotfi-Kamran, K. Balet, and B. Falsafi, "Cuckoo Directory: A Scalable Directory for Many-core Systems," in *Proceedings of the 17th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2011, pp. 169–180.

[12] G. Gerosa, S. Curtis, M. D'Addeo, B. Jiang, B. Kuttanna, F. Merchant, B. Patel, M. Taufique, and H. Samarchi, "A Sub-1W to 2W Low-Power IA Processor for Mobile Internet Devices and Ultra-Mobile PCs in 45nm Hi-K Metal Gate CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2008, pp. 256–611.

[13] A. Sodani, "Knights Landing (KNL): 2nd Generation Intel Xeon Phi Processor," 2015. [Online]. Available: http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.25-Tuesday-Epub/HC27.25.70-Processors-Epub/HC27.25.710-Knights-Landing-Sodani-Intel.pdf

[14] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, 1st ed. Morgan Kaufmann Publishers Inc., 2003.

[15] R. Hesse, J. Nicholls, and N. E. Jerger, "Fine-Grained Bandwidth Adaptivity in Networks-on-Chip Using Bidirectional Channels," in *Proceedings of the 6th IEEE/ACM International Symposium on Networks-on-Chips (NOCS)*, May 2012, pp. 132–141.

[16] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary, "JETTY: Filtering Snoops for Reduced Energy Consumption in SMP Servers," in *Proceedings of the 7th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Jan. 2001, pp. 85–96.

[17] P. Lotfi-Kamran, M. Ferdman, D. Crisan, and B. Falsafi, "TurboTag: Lookup Filtering to Reduce Coherence Directory Power," in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, Aug. 2010, pp. 377–382.

[18] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-Out Processors," in *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2012, pp. 500–511.

[19] B. Grot, D. Hardy, P. Lotfi-Kamran, B. Falsafi, C. Nicopoulos, and Y. Sazeides, "Optimizing Data-Center TCO with Scale-Out Processors," *IEEE Micro*, vol. 32, no. 5, pp. 52–63, Sep. 2012.

[20] J. D. Balfour and W. J. Dally, "Design Tradeoffs for Tiled CMP On-Chip Networks," in *Proceedings of the 20th Annual ACM International Conference on Supercomputing (ICS)*, Jun. 2006, pp. 187–198.

[21] "International Technology Roadmap for Semiconductors (ITRS), 2011 Edition." [Online]. Available: http://www.itrs2.net/2011-itrs.html

[22] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *Proceedings of the 6th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, May 2012, pp. 201–210.

[23] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2007, pp. 3–14.

[24] J. Turley, "Cortex-A15 "Eagle" Flies the Coop," *Microprocessor Report*, vol. 24, no. 11, pp. 1–11, Nov. 2010.

[25] CloudSuite, http://cloudsuite.ch.

[26] Flexus, http://parsa.epfl.ch/simflex/flexus.html.

[27] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim, and W. J. Dally, "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2013, pp. 86–96.

[28] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe, "SimFlex: Statistical Sampling of Computer System Simulation," *IEEE Micro*, vol. 26, no. 4, pp. 18–31, July-August 2006.

[29] A. Kumar, P. Kundu, A. P. Singh, L.-S. Peh, and N. K. Jha, "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS," in *Proceedings of the 25th International Conference on Computer Design (ICCD)*, Oct. 2007, pp. 63–70.

[30] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2005, pp. 408–419.

[31] B. K. Daya, C.-H. O. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L.-S. Peh, "SCORPIO: A 36-core Research Chip Demonstrating Snoopy Coherence on a Scalable Mesh NoC with In-network Ordering," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA)*, Nov. 2014, pp. 25–36.

[32] J. Kim, J. Balfour, and W. Dally, "Flattened Butterfly Topology for On-Chip Networks," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2007, pp. 172–182.

[33] A. Jain, R. Parikh, and V. Bertacco, "High-Radix On-chip Networks with Low-Radix Routers," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2014, pp. 289–294.

[34] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express Cube Topologies for On-Chip Interconnects," in *Proceedings of the 15th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2009, pp. 163–174.

[35] M. M. Kim, J. D. Davis, M. Oskin, and T. Austin, "Polymorphic On-Chip Networks," in *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2008, pp. 101–112.

[36] H. Yang, J. Tripathi, N. E. Jerger, and D. Gibson, "Dodec: Random-Link, Low-Radix On-Chip Networks," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2014, pp. 496–508.

[37] U. Y. Ogras and R. Marculescu, ""It's a Small World After All": Noc Performance Optimization via Long-range Link Insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, Jul. 2006.

[38] M. Modarressi, A. Tavakkol, and H. Sarbazi-Azad, "Application-Aware Topology Reconfiguration for On-Chip Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 11, pp. 2010–2022, Nov. 2011.

[39] P. Gratz, B. Grot, and S. W. Keckler, "Regional Congestion Awareness for Load Balance in Networks-on-Chip," in *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2008, pp. 203–214.

[40] P. Lotfi-Kamran, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY - A Low Cost Congestion-Aware Routing Algorithm for Network-on-Chips," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 256–264, Jul. 2010.

[41] S. Ma, N. E. Jerger, and Z. Wang, "DBAR: An Efficient Routing Algorithm to Support Multiple Concurrent Applications in Networks-on-chip," in *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2011, pp. 413–424.

[42] B. Fu, Y. Han, J. Ma, H. Li, and X. Li, "An Abacus Turn Model for Time/Space-efficient Reconfigurable Routing," in *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2011, pp. 259–270.

[43] P. Lotfi-Kamran, "Per-Packet Global Congestion Estimation for Fast Packet Delivery in Networks-on-Chip," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3419–3439, Sep. 2015.

[44] M. A. Kinsy, M. H. Cho, T. Wen, E. Suh, M. van Dijk, and S. Devadas, "Application-aware Deadlock-free Oblivious Routing," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2009, pp. 208–219.

[45] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Argia: Exploiting Packet Latency Slack in On-chip Networks," in *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2010, pp. 106–116.

[46] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks," in *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2008, pp. 89–100.

[47] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware Prioritization Mechanisms for On-chip Networks," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2009, pp. 280–291.

[48] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, "The Power of Priority: NoC Based Distributed Cache Coherency," in *Proceedings of the 1st IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, May 2007, pp. 117–126.

[49] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, "Prediction Router: Yet Another Low Latency On-Chip Router Architecture," in *Proceedings of the 15th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2009, pp. 367–378.

[50] G. Michelogiannakis, N. Jiang, D. Becker, and W. J. Dally, "Packet Chaining: Efficient Single-cycle Allocation for On-chip Networks," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2011, pp. 83–94.

[51] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2006, pp. 333–346.

[52] Y. Xu, B. Zhao, Y. Zhang, and J. Yang, "Simple virtual channel allocation for high throughput and high frequency on-chip routers," in *Proceeding of the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Jan. 2010, pp. 1–11.

[53] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das, "Performance and power optimization through data compression in Network-on-Chip architectures," in *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2008, pp. 215–225.

[54] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A Case for Heterogeneous On-chip Interconnects for CMPs," in *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*, Nov. 2011, pp. 389–400.

[55] S. H. Seyyedaghaei Rezaei, A. Mazloumi, M. Modarressi, and P. Lotfi-Kamran, "Dynamic Resource Sharing for High-Performance 3-D Networks-on-Chip," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 5–8, Jan. 2016.

[56] 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)*, Nov. 2004, pp. 188–197.

[57] 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)*, Jun. 2007, pp. 150–161.

[58] A. Kumar, L.-S. Peh, and N. K. Jha, "Token Flow Control," in *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Nov. 2008, pp. 342–353.

[59] M. Ahn and E. J. Kim, "Pseudo-Circuit: Accelerating Communication for On-Chip Interconnection Networks," in *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2010, pp. 399–408.

[60] 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)*, Jun. 2009, pp. 196–207.

[61] M. Hayenga, N. E. Jerger, and M. Lipasti, "SCARAB: A Single Cycle Adaptive Routing and Bufferless Network," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2009, pp. 244–254.

[62] A. Abousamra, A. K. Jones, and R. Melhem, "Proactive Circuit Allocation in Multiplane NoCs," in *Proceedings of the 50th Annual Design Automation Conference (DAC)*, Jun. 2013, pp. 35:1–35:10.

[63] R. A. Stefan, A. Molnos, and K. Goossens, "dAElite: A TDM NoC Supporting QoS, Multicast, and Fast Connection Set-Up," *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 583–594, Mar. 2014.