APCR: An Adaptive Physical Channel Regulator for On-Chip Interconnects

Lei Wang, Poornachandran Kumar, Ki Hwan Yum, and Eun Jung Kim
Department of Computer Science and Engineering, Texas A&M University
College Station, TX, US
{wanglei, chandran, yum, ejkim}@cse.tamu.edu

ABSTRACT

Chip Multi-Processor (CMP) architectures have become mainstream for designing processors. With a large number of cores, Network-On-Chip (NOC) provides a scalable communication method for CMP architectures, where wires become abundant resources available inside the chip. NOC must be carefully designed to meet constraints of power and area, and provide ultra low latencies. In this paper, we propose an Adaptive Physical Channel Regulator (APCR) for NOC routers to exploit huge wiring resources. The flit size in an APCR router is less than the physical channel width (phit size) to provide finer granularity flow control. An APCR router allows flits from different packets or flows to share the same physical channel in a single cycle. The three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) intelligently allocate the output channel resources considering not only the availability of physical channels but the occupancy of input buffers. In an APCR router, each Virtual Channel can forward a dynamic number of flits every cycle depending on the run-time network status. Our simulation results using a detailed cycle-accurate simulator show that an APCR router improves the network throughput by over 100% in synthetic workloads, compared with a traditional design with the same buffer size. An APCR router can outperform a traditional router even if the buffer size is halved.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiprocessors—*Interconnection architectures*

Keywords

Channel Regulator, CMP, On-Chip Interconnects

1. INTRODUCTION

Moore's law has steadily increased on-chip transistor density and integrated dozens of components on a single die. Providing efficient communication in a single die is becoming a critical factor for high performance Chip Multi-Processors (CMPs) [23]. Traditional shared buses and dedicated wires do not meet the communication demands for future multi-core architectures. Moreover, the shrinking technology exacerbates the imbalance between transistors and wires in terms of delay, and power has embarked on a fervent search for efficient communication designs [12]. In this regime, Network-On-Chip (NOC) with packet-switching is a promising architecture that orchestrates chip-wide communications towards future many-core processors.

As feature size is shrinking, wires are becoming abundant resources available in NOC [3, 22].Since NOC can benefit from high wire density due to no limits on the number of pins and faster signaling rates, it is very critical in the NOC router design to find a way that fully utilizes the wire resources to provide high performance. There have been a handful studies exploiting abundant wire resources to explore different topologies with high degree channels such as Flattened Butterfly [15] and Express Cube [10]. Meanwhile, this wiring capability has led to networks with very wide transmission channels between routers, which facilitate low latency due to small serialization delay [3, 6]. However, as shown in Figure 1, if we simply increase the channel width and define the flit size to be the same as the channel width or phit size, we cannot achieve significant performance improvement for the same router buffer budget. It is interesting to observe that the performance with 512-bit channels is worse than the one with 256-bit channels. Due to the limited buffer area budget, increasing flit size proportionately decreases the virtual channel (VC) depth, which incurs more contention and, in turn, more performance degradation. To reduce the network contention, it is obvious to define a smaller flit size for finer granularity flow control. Therefore, to balance the trade-off between serialization latency and contention delay, the flit size should be carefully re-examined to exploit wide channels for the NOC router design.

Furthermore, diverse packet sizes found in the CMP communications also limit the usage of wide channels. The majority of on-chip communication emanates from cache traffic, such as cache coherence messages or L1/L2 cache blocks. A cache coherence message, like a request or an invalidation, consists of a small header and a memory address only, which are around 64 bits, while a packet that carries a L1/L2 cache block is as large as 512 bits (64 Bytes). Recently, Das et al. [8] suggest to define two kinds of flits (short and long) to maximize the usage of the physical channel. However, it is not trivial to provide two different flit sizes in a network due to the complicated credit management. Therefore, we attempt to solve this challenging problem to utilize wide channel resources in NOC without defining multiple flit sizes for different packet lengths.

In this paper, we propose an Adaptive Physical Channel Regulator (APCR) for NOC routers. Even though there may exist different packet lengths (short or long) in the same network, an APCR router only defines one constant flit size, which is typically smaller

PACT'12, September 19–23, 2012, Minneapolis, Minnesota, USA.

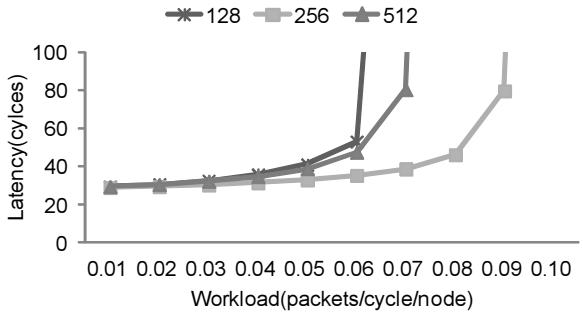


Figure 1: Average Packet Latencies with Three (128, 256 and 512 bits) Different Channel Widths and a Fixed Router Buffer Size in an (8×8) Mesh Network (Detailed simulation configuration can be found in Section 5.).

than the channel width. An APCR router allows VCs to transmit multiple small flits in one cycle. The number of flits being transmitted every cycle dynamically changes according to the network status. We present three regulation schemes for APCR: Monopolizing, Fair-sharing and Channel-stealing. Monopolizing allows only one packet to use the physical channel every cycle, while in Fair-sharing multiple packets can traverse at the same time. Channel-stealing is most flexible, where idle physical channels can be utilized by any ready packet. Routers become run-time configurable to different packet lengths with these three regulation schemes. It is similar to providing routers with diverse flit sizes to meet the requirements of different packet lengths. Long packets with multiple flits can transmit more than one flit in a cycle when no other packets want to use the same physical channel (This occurs in a low-workload network.). Short packets, normally containing one flit, use only a portion of the physical channel and leave other portions utilized by other packets, or the unused portions are fully shut down thus consuming only static power. Furthermore, an APCR allocates the channel resources using the occupancy of input buffers, thoroughly utilizing the wide channel and providing high network throughput. Our simulation results show that an APCR router outperforms a traditional router with double the buffer size. When buffer sizes are identical, an APCR router achieves more than 100% throughput improvement in synthetic workloads without negative effect on zero-load latency. Moreover, it is observed that an APCR router improves the network performance regardless of the underlying topologies. Monopolizing achieves 30% improvement in both Flattened Butterfly and Express Cube while Channel-stealing gets 40%, which demonstrates the synergy effect of the two approaches (wide transmission channels and high degree channels) exploiting abundant wire resources.

The rest of this paper is organized as follows. We briefly summarize the related work in Section 2. We present the three channel regulation schemes in Section 3. Section 4 gives details on the microarchitecture of an APCR router. In Section 5, we describe the evaluation methodology and summarize the simulation results. Finally, we draw conclusions in Section 6.

2. RELATED WORK

Several NOC topologies have been proposed to explore abundant channel resources. MeshX2 [3] increases the wire and area utilization by introducing a second parallel network. Each subnetwork of the MeshX2 transports a disjoint subset of packets, one subnetwork for long packets and the other for short packets. However, partitioning the networks based on the packet lengths performs poorly when such a partitioning fails to maximize the available wire resources usage. Flattened butterfly [15] shows the benefit of having multiple parallel channels, and each parallel channel connects to a different switch in the same row or column. Hence, the number of ports in a switch will be increased. Since some of the parallel channels are multi-hop channels, the wires have to cross the chip, increasing the complexity of the floorplan. Express Cube topologies [10] get rid of the multiple parallel channels among routers. Instead, Multidrop Express Channels (MECS), a one-to-many communication fabric, are introduced to connect each router. It is our understanding that this design relies heavily on the intelligent repeaters in the MECS, and the design complexity of the intelligent repeaters can be high, because the intelligent repeaters should control the point-to-multipoint unidirectional links at run-time.

There are also some designs that share the channel among different flits in a single cycle. Das et al. [8] look into multiple flits sharing a channel. In their network, there are two kinds of flits, short and long. The sharing condition is simple: if two short flits are routed to the same output port, they can simultaneously traverse the crossbar and output channel. For long flits, no sharing is applied. The flow control can be challenging to support two flit sizes in a network, since it makes the credit management complicated. Leroy et al. [17] introduce Spatial Division Multiplexing (SDM) into the NOC design by providing a fully free reconfigurable wire level data path. The results show that SDM is a more interesting approach than Time Division Multiplexing (TDM), due to the high complexity and power needed by buffers to store the TDM configuration for each clock cycle. In [25, 26], abundant wire resources are explored, where the main goal is to provide more path diversity.

3. ADAPTIVE PHYSICAL CHANNEL REGULATION SCHEMES

In this section, we primarily elucidate three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) used in an APCR Router, while Section 4 delves into its detailed microarchitecture.

3.1 Monopolizing

Monopolizing allows only one VC to use the total bandwidth of the physical channel every cycle. In a generic router design, the flit size is usually the same as the phit size, thus a VC can fully use the whole bandwidth of the physical channel. However, in the APCR router the flit size is smaller than the phit size, so potentially multiple flits can be delivered in the same channel concurrently. Monopolizing allows a VC to transmit a different number of flits in one cycle. There is one restriction on this situation: a VC is only allowed to transmit the flits that belong to the same packet to prevent different packets in the same VC from interleaving. The maximum number of transmitted flits depends on how many flits stored in a VC belong to the same packet, not on how many flits the physical channel can process. In Figure 2 the phit size is four times the flit size, which implies the physical channel can transmit at most four flits in one cycle. Meanwhile, the VC depth is also four, and each packet consists of four filts (one head flit, two middle flits and one tail flit). Assume that VC_0 wins the arbitration of the output channel, and it contains four flits as follows: one middle flit at the head of VC_0 followed by a tail flit belonging to the same packet, then a head flit and a middle flit belonging to a different packet. In this case, although the physical channel can transmit four flits at a time, only the first two flits are allowed to be transmitted. The head and middle flits of the next packet should wait until the next cycle

because they belong to a different packet that may have a different routing direction.

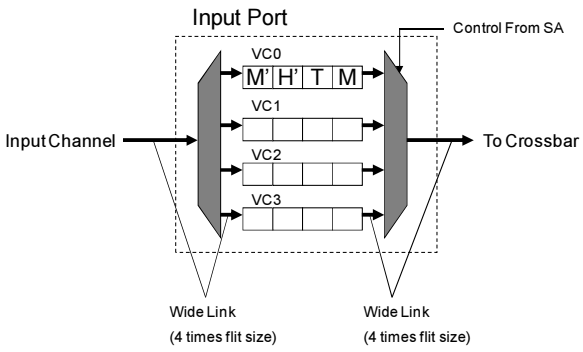


Figure 2: Monopolizing: Flits M and T belong to a packet, and flits H' and M' are from another packet. VC_0 wins the arbitration of the output channel. However, only M and T can be transmitted in the same cycle.

Compared with the generic router design, a smaller flit size can provide finer granularity of traffic control. The best case occurs when VC_0 always buffers four flits from the same packet, since VC_0 can transmit four flits every cycle that will occupy the whole bandwidth of the physical channel. This situation is similar to defining the flit size equal to the phit size. If we treat the term "flit" as the unit that a VC sends every cycle, an APCR router makes the "flit" size run-time configurable according to the status of the network, and routers use different "flit" sizes at the same time. However, Monopolizing can potentially waste the channel bandwidth, as shown in Figure 3. We assume that each VC has at least one flit and within each VC the flits to be sent are going to the same direction. If VC_2 wins the output channel, according to Monopolizing, only one quarter of the physical channel is utilized even though other VCs have flits waiting to be sent through the same physical channel.

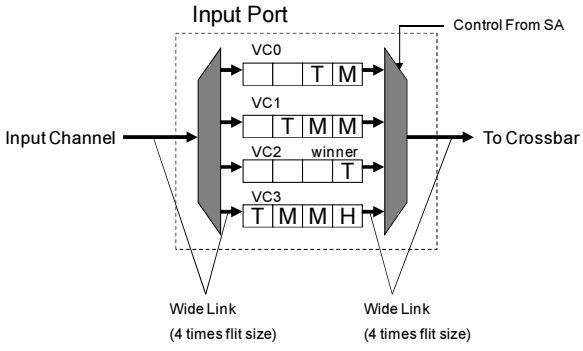


Figure 3: Monopolizing Wasting the Channel Bandwidth: VC_2 wins the arbitration of the physical channel. However, VC_2 has only one flit to be sent, thus wastes three quarters of the channel bandwidth.

3.2 Fair-sharing

Considering the inefficient channel utilization described above, we propose Fair-sharing, where VCs fairly share the physical channel resources. In Fair-sharing, a wide physical channel needs to be divided into several small parts, called sub-channels. We reserve a separate sub-channel for each VC in an input port. If a physical channel is divided into four sub-channels and each input port has four VCs, each VC of the same input port can have one sub-channel of its own (VC_0 uses sub-channel$_0$, VC_1 uses sub-channel$_1$, and so forth.). In addition, VCs from different input ports can share a sub-channel. For example, all VC_0s of different input ports share sub-channel$_0$. To simplify the hardware design, the ID of a VC is bound with the ID of a sub-channel. Since the width of a sub-channel is a flit, multiple VCs can send flits concurrently even though they belong to the same input port. If the number of sub-channels is bigger than the number of VCs in an input port, each VC can be assigned more than one sub-channel.

Compared with Monopolizing, Fair-sharing utilizes the wide channel more efficiently, because the chance of sending multiple flits from multiple VCs is normally higher than sending multiple flits from a single VC, especially at high workload. Fair-sharing still has its own limitations, as illustrated in Figure 4, where VC_0 has two flits, VC_1 has three, but VC_2 and VC_3 are empty. According to Fair-sharing, VC_0 and VC_1 will use sub-channel$_0$ and sub-channel$_1$ to send one flit each. Since VC_2 and VC_3 have no flits, sub-channel$_2$ and sub-channel$_3$ are empty. Even though VC_0 and VC_1 have more flits to be sent, they cannot use sub-channel$_2$ and sub-channel$_3$ because they are not bound to sub-channel$_2$ and sub-channel$_3$.

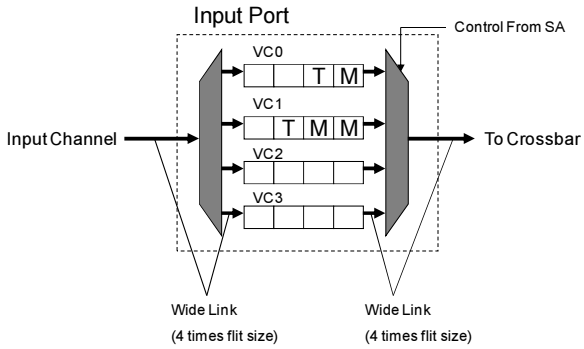


Figure 4: Fair-sharing: VC_0 and VC_1 can only send one flit each using their own sub-channels. VC_2 and VC_3 waste their sub-channels because they do not have any flits. These wasted sub-channels cannot be used by VC_0 and VC_1.

3.3 Channel-stealing

To improve the utilization of wide channels further, we propose Channel-stealing, which is built upon Fair-sharing. If a VC finally has no flit to be sent, its sub-channel can be stolen by other VCs. Here the stealing can occur in two different ways. One is stealing from VCs belonging to the same input port, and the other is stealing from VCs of different input ports that have no flits to be sent in the same output direction. Channel-stealing can explore the channel resources thoroughly. It optimizes the arbitration of output channels by using the buffer occupancy information from each VC and finally increases the network throughput. In Figure 4, since

VC_2 and VC_3 have no flits to be sent, VC_0 and VC_1 can steal the sub-channels assigned to VC_2 and VC_3, and send more than one flit. There are two options: Both VC_0 and VC_1 send two flits each or VC_0 sends one flit while VC_1 sends three flits. Choosing from the two options depends on the scheduling policy. In this study, we adopt Round-Robin as our scheduling policy, which fairly allocates the extra free sub-channels to other VCs with more flits.

4. APCR ROUTER MICROARCHITECTURE DESIGN

To support three channel regulation schemes in Section 3, a generic NOC router microarchitecture has to be enhanced. In this section, we first briefly explain a generic NOC router microarchitecture and the functionality of each pipeline stage. Then we propose the APCR router microarchitecture design and analyze each main component.

4.1 A Generic NOC Router

Figure 5 (a) shows a generic NOC router architecture [7] for a 2D mesh network. In most implementations, there are 5 ports in the router: four from/to the four cardinal directions (NORTH, EAST, SOUTH and WEST) and one from/to local Processing Element (PE). The main building blocks for a generic NOC router are input buffer, route computation logic, VC allocator, switch allocator, and crossbar. To achieve high performance, routers process packets with four pipeline stages, which are routing computation (RC), VC allocation (VA), switch allocation (SA), and switch traversal (ST). First, the RC stage directs a packet to a proper output port by looking up its destination address. Next, the VA stage allocates one available VC of the downstream router determined by RC. The SA stage arbitrates input and output ports of the crossbar, and successfully granted flits traverse the crossbar during the ST stage. Due to the stringent area budget of a chip, routers use flit level buffering in a wormhole-switching network as opposed to packet level buffering. Additionally, the buffer is managed with credit-based flow control, where downstream routers provide back-pressure to upstream routers to prevent buffer overflow. Considering that only the head flit needs routing computation and middle flits always have to stall at the RC stage, low-latency router designs parallelize RC, VA and SA using lookahead routing [9] and speculative switch allocation [24]. The functionality of lookahead routing is the same as the normal RC stage, calculating the output ports. However, instead of calculating routing information for the current router, lookahead routing does the same for the downstream router and stores the routing information in the head flit. In this way the RC and VA stages can be overlapped because the VC allocator does not need to wait for the output of the RC logic. Speculative switch allocation [24] predicts the winner of the VA stage and performs SA based on the prediction. If the packet fails to allocate a VC, the pipeline stalls and both the VA and SA stages will be repeated in the next cycle. These two modifications lead to two-stage and even single-stage [21] routers, which parallelize the various stages in the router. In this paper, we use a two-stage router as the baseline router.

4.2 APCR Router Microarchitecture

Figure 5 (b) shows the microarchitecture of an APCR router, where the differences from a generic router are shaded. In an APCR router, the VC allocator is the same as a generic router for the three regulation schemes. Also no modification will be needed to the crossbar. Even with wide channels, the number of input and output ports will be the same (5×5 crossbar for a 2D mesh topology). The APCR component works together with switch allocator and input buffers. First, based on the characteristics of the regulation schemes selected, APCR decides the number of flits each VC can send. For example, Monopolizing allows a VC to send multiple flits but selects only one VC from each input port, while in Fair-sharing a VC can only send one flit but no competition exists among VCs of the same input port. Second, APCR controls the input buffer operations according to the allocation results. In the following we describe the design of each main component in an APCR router.

4.2.1 Buffer Management

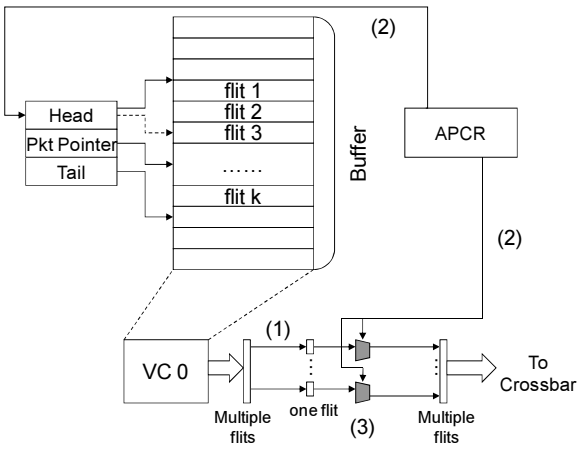
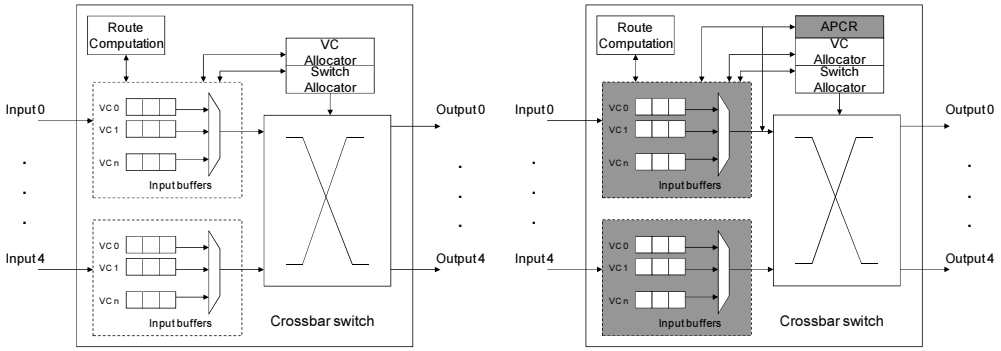


Figure 7: Three Steps of Buffer Management: (1) VC_0 reads multiple flits out (the number of flits depends on the relationship between the flit size and phit size.). (2) APCR sets up the head of each input buffer and the output. (3) The guaranteed flits are sent through the crossbar. The remaining flits will be dropped and read out again in the next cycle.

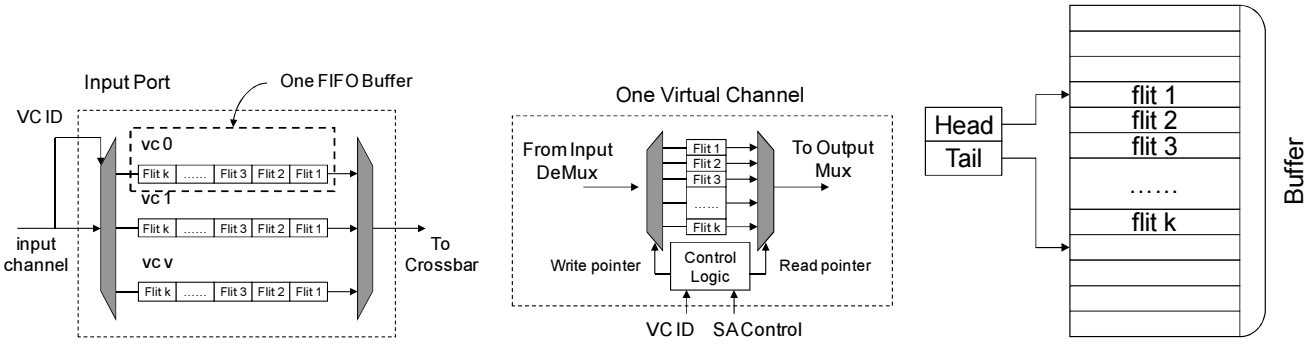
Due to the limitations of power and area, NOC routers rely on simple buffer structures. As shown in Figure 6 (a), each input port has v VCs, each of which has dedicated k-flit FIFO buffer. Recent NOC routers have small input buffers to minimize their overhead. The necessity for low latency buffer operation mandates the use of a parallel FIFO as illustrated in Figure 6 (b). As opposed to a serial FIFO implementation [30], the parallel FIFO eliminates the need for a flit to traverse all slots in a pipelined manner before exiting the buffer. This fine-grained control uses read and write pointers to maintain the FIFO order. Figure 6 (c) shows a data structure to represent parallel FIFO buffers. When a read operation is performed, the flit pointed by the head is sent out and the head pointer is automatically moved to the next position. Similarly, when a write operation comes, the flit is written to the position pointed by the tail and the tail pointer is increased by one. When the head is equal to the tail after a read, the buffer is empty. When the tail is equal to the head after a write, the buffer becomes full. We use this parallel FIFO implementation for our study.

The three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) require different buffer structures. While the input buffer for Fair-sharing is the same as that of a generic router since every cycle only one flit can be sent from a VC, Monopolizing and Channel-stealing need a different buffer structure. Since Monopolizing and Channel-stealing allow a VC to send multiple



(a) A Generic NOC Router Architecture (b) An APCR Router Architecture

Figure 5: Generic Router and APCR Router Architectures.



(a) Input Buffer Organization (b) VC Design with Parallel FIFO Implementation (c) Data Structure for Parallel FIFO Buffers

Figure 6: Generic NOC Router Input Buffer Architecture.

flits to the output channel per cycle, the input buffer should support multiple-flit read or write operations. Instead of the multiple read or write ports buffer design [27] that incurs significant overhead, the proposed buffer design reads data from each VC as wide as the output channel bandwidth. Note that all the data (e.g. 4 flits) read from a VC may not be sent to the output channel in a cycle. Then those flits that were read out but not sent should be dropped. To support the functionality of reading and dropping flits, the data structure shown in Figure 6 (c) should be modified, since it is necessary to add a packet pointer which points to the end position of a packet. Also the input buffer should be notified how many flits are effectively sent out. After each read operation, the head can be moved to the correct position. This information is gathered from the APCR component after the SA stage. Meanwhile, the channel allocation information should be transmitted to the downstream router through control links, since flits in different parts of the channel may enter different VCs. It does not incur any extra router delays, because the transmission can overlap with the ST and Link Traverse (LT) stages.

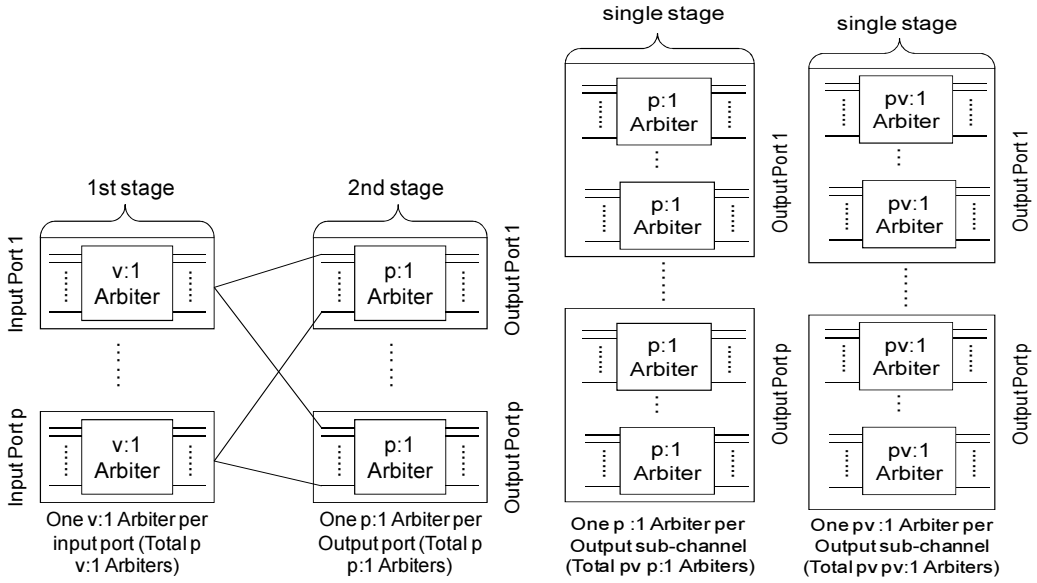
Figure 7 describes the whole procedure of the buffer management in an APCR router that supports Monopolizing and Channel-stealing. VC_0 buffers multiple flits that belong to the same packet. At step (1) multiple flits will be read out since the buffer output width is equal to the router output channel width, a multiple of the flit size. After the SA stage, the APCR component has the information of how many flits each VC can eventually send. At step (2) the allocation information will be used to set up the head pointer in each input buffer and the control signals of each output. At step (3) the guaranteed flits will be sent to downstream routers through the crossbar and the remaining read-out flits are dropped. These dropped flits will be read out from the input buffers again in the next cycle. The buffer write adopts the same method as the buffer read. Obviously, these redundant operations will incur more energy consumption. Compared with Fair-sharing, Monopolizing and Channel-stealing consume average 13% more energy for delivering one packet[1].

4.2.2 VC and Switch Allocation

The VC allocation in an APCR router is the same as that in a generic router. However, the switch allocation is different, which brings the main overhead. Figure 8 (a) shows the two-stage arbitration used in a generic router[2]. The first stage is to select one VC from each input port. It needs p $(v : 1)$ arbiters. The second stage is for output ports, selecting one valid request from p input ports which have the same output direction. Hence, each output port needs a $(p : 1)$ arbiter and the total number of the second stage arbiters adds up to p. Monopolizing has the same SA structure as a generic router, which does not incur any SA overhead.

[1]We use Orion 2.0 [13] for energy and area estimation in 45nm technology.

[2]We assume the router has p input ports and each input port has v virtual channels.



(a) Generic Case (b) Fair-sharing (c) Channel-stealing

Figure 8: Switch Allocation.

Switch allocation for Fair-sharing is different as shown in Figure 8 (b). In Fair-sharing, each VC of the same input port has its own reserved sub-channel, which guarantees one flit bandwidth. There is no competition among the VCs of the same input port. They share the wide channel between the input buffer and the crossbar, each occupying one flit bandwidth. This removes the first stage arbitration of a generic router. However, each output sub-channel needs an arbiter to decide the current winner because it receives requests from VCs of different input ports. Considering VCs are bound with output sub-channels in Fair-sharing[3], the number of inputs of an output sub-channel arbiter is the same as the number of input ports, p. Assuming that the number of sub-channels of an output port is equivalent to the number of VCs of an input port (v), pv $(p : 1)$ arbiters should be provided. Channel-stealing needs even more complex SA structure because it provides the most flexible channel usage. Similar to Fair-sharing, only a single stage arbitration is required, as shown in Figure 8 (c). However, Channel-stealing does not bind VCs with sub-channels, which means a VC can request any sub-channel in the same output direction. Hence, each output sub-channel needs a $(pv : 1)$ arbiter and the whole SA requires pv $(pv : 1)$ arbiters.

Figure 9 shows the normalized area for the three schemes. We can see that an APCR router incurs less than 1% area overhead. The difference mainly comes from the SA arbiters. Monopolizing and Channel-stealing require extra control logic in buffer management, which does not incur much area overhead. Compared with other components, the area used by the arbiters and control logic is negligible. This explains why we cannot see much area difference when we consider the total router area.

4.2.3 Credit-Based Flow Control

With credit-based flow control, the upstream router keeps a record of the number of free buffers in each VC of downstream routers. Every cycle, when the upstream router forwards a flit, it will consume one credit until the credit counter becomes zero, which means the VC of the corresponding downstream router is full. In the opposite way, when the downstream forwards a flit and frees the associated buffer, it sends a credit to the upstream, causing the credit counter to increase by one. In the generic NOC design, the flit size is the same as the phit size, which means every cycle the current router can forward at most one flit to its downstream router in each direction. This triggers a credit to be sent to the corresponding upstream router. However, in an APCR router the flit size is no more equivalent to the phit size (normally smaller), which causes

[3] VC_0 from any input port can only use sub-channel$_0$ of all the output ports.

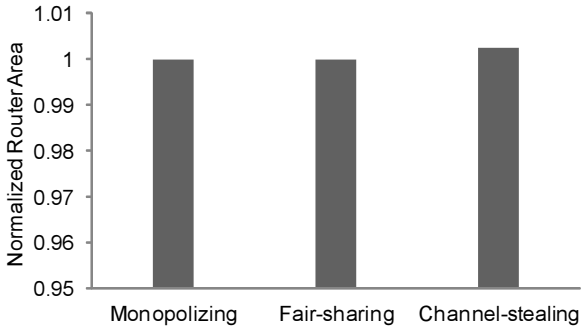


Figure 9: Total Router Area Normalized to a Generic Router.

Table 1: Time Analysis of the SA Stage Delay from HSPICE Simulations Using 45nm Technology.

Generic	Monopolizing	Fair-sharing	Channel-stealing
247 ps	247 ps	122 ps	333 ps

Table 2: Basic Network Configuration and Variations.

Characteristic	Basic	Variations
Topology	8×8 2D Mesh	MeshX2, MeshX4, FBFLY, MECS
Routing	XY	–
Router Arch	Two-stage Speculative	APCR Router
Per-hop Latency	3 cycles: 2 cycle in a router, 1 cycle to cross a link	–
Virtual Channels/Port	4	–
Phit Size (Channel Width)	512 bits	–
Flit Size	512 bits	128 bits
Packet Length (flits)	one (control packet), five (data packet)	–
Traffic Pattern	Uniform Random, Bit Complement, Transpose	SPEComp, SPECjbb, SPLASH-2 and PARSEC
Simulation Warm-up Cycles	10,000	–
Total Simulation Cycles	200,000	10,000,000 (Benchmarks)

Table 3: CMP System Parameters.

Clock Frequency	4 GHz
Core Count	32
L1 I and D cache	1-way and 4-way, 32KB, 1 cycle
L2 cache	16-way, 16MB, 512KB per bank, 32 banks, 20 cycles
L1/L2 cache block	64B
Memory Latency	300 cycles

the number of credits sent in each direction to be greater than one. To solve this problem, more wires are needed between two neighboring routers to transmit the credit information. Considering the depth of a VC is not big, the overhead of credit wires can be trivial.

4.2.4 APCR's Effect on the Router Pipeline

NOC router pipeline stage delays are quite imbalanced unlike the processor pipeline [24].Normally, the VA stage is the bottleneck [20]. The VA stage in an APCR router is the same as in a generic NOC router, but the SA stage is different. In the following time analysis, we mainly focus on the SA stage. Table 1 shows the SA stage delays for a generic router and three different regulation schemes. These results are obtained from HSPICE simulations using 45nm technology. Since Monopolizing has the same SA as a generic router, they have the same SA delay. Fair-sharing has the lowest delay, because of two main reasons: first, Fair-sharing only needs a single stage arbitration; second, binding VCs with output sub-channels makes the number of inputs of an output port arbiter small. Channel-stealing has the highest SA stage delay because of its complicated arbitration. However, when we compare these delays with the VA stage delay (328ps), the SA stage delays of all the schemes are still affordable. Additionally, according to the time stealing technology proposed by Das et al. [20], a slower stage in the router gains evaluation time by stealing it from successive or previous router pipeline stages. Considering the relative lower delay of the ST stage, the SA stage can steal time from the ST stage by delaying the triggering edge of the clock to all the subsequent latches. By adopting the time stealing technology, the APCR component is expected to have marginal effects on the router clock cycle time.

5. EXPERIMENTAL EVALUATION

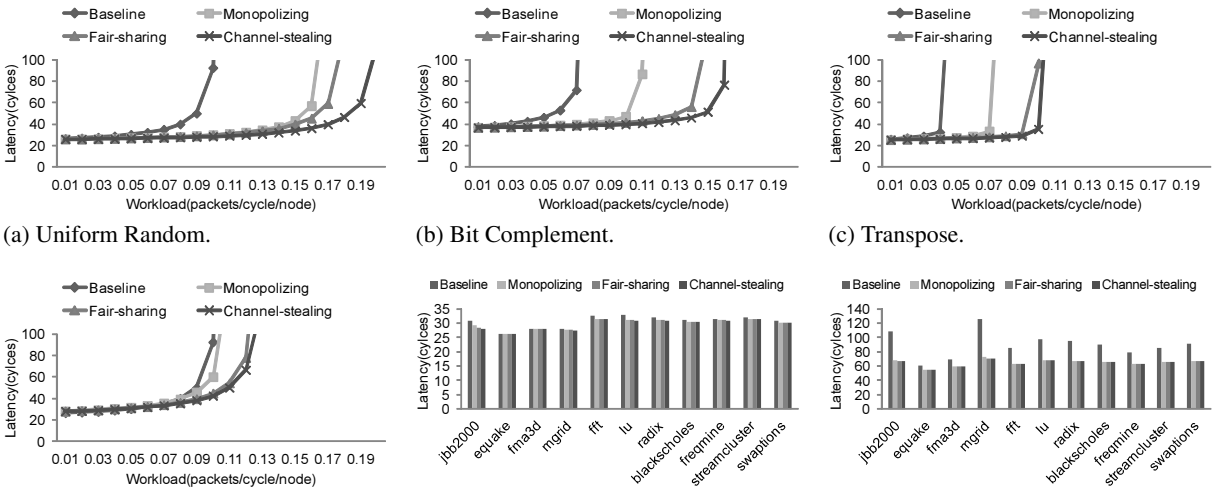
In this section, we present a detailed evaluation of the proposed design.

5.1 Methodology

We evaluate the performance of our proposed schemes with a cycle-accurate network simulator that models all router pipeline delays and wire latencies. To achieve a fair comparison, we assume the same hardware budgets for the baseline router and the APCR router, including buffer and wire resources. We model a link as 512 parallel wires, which takes advantage of abundant metal resources provided by future multi-layer interconnects. In the baseline router, the flit size is the same as the link width (512 bits), while an APCR router uses a 128-bit flit. We simulate two kinds of packets in the network: control packets (short packets) and data packets (long packets). Note that each packet, whether it is short or long, has the same number of bits in both the baseline and the APCR router designs. With different filt sizes, the packet size of the baseline router is different from that of the APCR router. For example, a one-flit short packet and a five-flit long packet in the APCR design correspond to a one-flit short packet and a two-flit long packet in the baseline design. We choose an 8×8 2DMesh topology and XY routing algorithm, and configure four VCs for each port. Table 2 summarizes the network simulator configuration.

The workloads for our evaluation consist of synthetic workloads and real applications. Three different synthetic traffic patterns, Uniform Random (UR), Bit Complement (BC) and Transpose (TP), are used in our evaluation. In synthetic workloads, the percentage of short packets is 60%[4]. The packet generation rate for each node is

[4]The percentage is taken from SIMICS with GEMS [19] extension [5].



(a) Uniform Random.

(b) Bit Complement.

(c) Transpose.

(d) Uniform Random with the buffer size of the APCR Router halved.

(e) Real Applications.

(f) Average Latencies in the Congestion Period.

Figure 10: Packet Latencies with Three Synthetic Traffic Patterns and Real Applications.

constant. The real application workloads considered in this paper are three programs (fft, lu, radix) from SPLASH-2 [29], four benchmarks (blackscholes, streamcluster, swaptions and freqmine) from the PARSEC suite [4], three programs (equake,fma3d and mgrid) from SPEComp2001 [2], and SPECjbb2000 [1].

To extract traces from real applications, we use Simics [18], a full-system simulator, configured for UltraSPARCIII+ multiprocessors running Solaris 9 operating system. We develop a customized timing-model interface modeling out-of-order cores with 4 MSHRs per each processing core to implement a self-throttling CMP network [16]. Our CMP configuration has 32 out-of-order processors and 32 L2 cache banks in a single chip, modeling static non-uniform cache architecture (S-NUCA) [14]. Processing cores and L2 cache banks are connected through the on-chip interconnection network with 2D Mesh topology. All the cache related delays and area parameters are determined by CACTI [28]. Table 3 shows the main parameters of our CMP system.

5.2 Performance Results

We first evaluate the average packet latencies for the three schemes and the baseline router design. Since the flit sizes in the APCR design and the baseline design are different, we use *packet per node per cycle* as the metric of workloads to ensure a fair comparison.

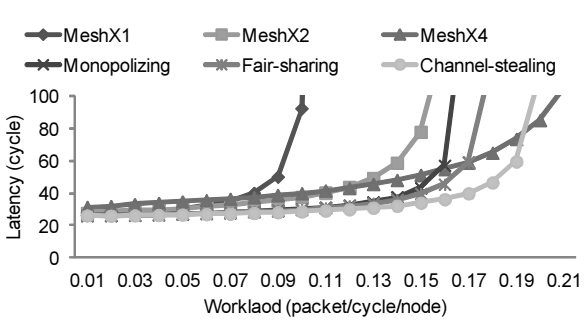
Standard Synthetic Workloads: Figure 10 (a), (b) and (c) show the simulation results using three synthetic workloads. The results are consistent with our expectations. The trends observed in all the three traffic patterns are the same. When the packet injection rate is low, the performance of the four schemes only has minor differences. However, with high injection rates, an APCR router outperforms the baseline router. For example, when the injection rate is 0.1, Monopolizing improves the performance by 67% for the Uniform Random traffic. Among the three regulation schemes, Channel-stealing is the best, since it utilizes the output channels most efficiently. If there are flits ready in any VC and the downstream routers have enough credits, the output channel can always be utilized, since no channel usage restrictions exist in Channel-stealing. In the baseline design, if a flit is transmitted through the channel, the wide channel is also fully used, since the flit size is the same as the channel width. However, a big flit size can affect the VC depth if we keep the router buffer size constant. When the packet injection rate is high, with shallow VC depth, contention occurs and the network can easily get saturated. An APCR router makes more packets share the same channel resources, which potentially relieves the network congestion. The three schemes perform much better than the baseline when the packet injection rate is more than 0.1. Also when the network congestion is relieved, the network throughput is improved. It is observed that Channel-stealing increases the throughput by more than 100% for all three traffic patterns compared with the baseline design. Figure 10 (d) shows the performance result with the buffer size of the APCR router halved. We can see that the APCR router can still outperform the baseline router.

Real Applications: Figure 10 (e) shows the average packet latency with real applications. Channel-stealing delivers the best performance while the baseline is the worst. Since the packet injection rate of each node in these real applications is very low (below 0.01), the latency improvement of the three schemes over the baseline is not obvious (jbb2000 gets 9.3% improvement while fft gets 4.3% and lu gets 6.6%.). However, when we analyze the communication latency in the congestion period[5] for these real programs, as shown in Figure 10 (f), an APCR router can improve the communication latency by 25% in most of the benchmarks.

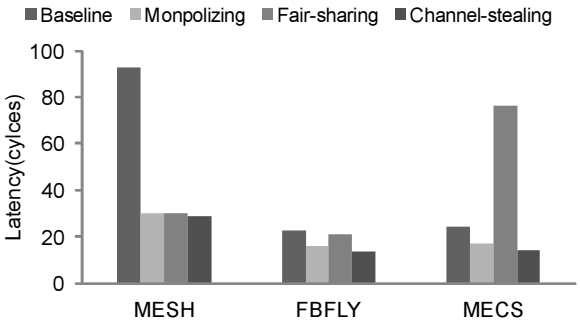
5.3 Topology Studies

In this section, we evaluate the performance of an APCR router in different topologies. Note that MeshX, such as MeshX2 and MeshX4, explores the channel resources by introducing multiple parallel networks. First, we compare the performance of an APCR router with that of MeshX using UR, as shown in Figure 11 (a). We assume that each design has the same buffer budget and bisection bandwidth, and that the flit size is always equal to the phit size in MeshX. It is observed that MeshX1, MeshX2 and MeshX4 have higher latency than the three APCR schemes before the network saturation point. Although MeshX divides the workload into mul-

[5]We define the congestion period as the time period when the average ratio of buffer occupancy in injection ports is above a threshold, which is set to 75% in our study.

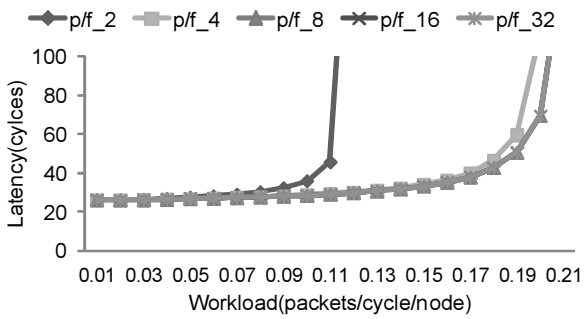


(a) Performance of MeshX and an APCR Router.

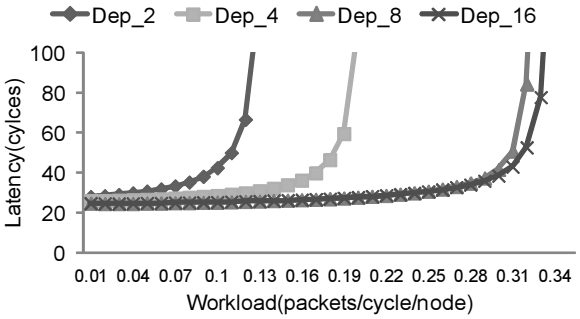


(b) Performance of an APCR Router in Various Network Topologies.

Figure 11: Topology Studies.



(a) Impact of the Number of Sub-channels on Packet Latency.



(b) Impact of the VC Depth on Packet Latency.

Figure 12: Sensitivity Studies with Different Network Design Points.

tiple parallel networks, these networks share the same amount of buffer space, which limits the VC depth of each port, making the network congested easier. MeshX4 produces the highest network throughput, benefiting from its multi-network design. However, MeshX4 increases the complexity of the crossbar, which incurs a big area overhead [11].

Next, we investigate the performance of an APCR router in recently proposed topologies, as shown in Figure 11 (b). We use a 2D Mesh, Flattened Butterfly (**FBFLY**) [15] and Multitask Express Channels (**MECS**) [10] with Uniform Random traffic[6], assuming each physical channel has the same bandwidth. We can see that an APCR router improves the network performance regardless of the underlying topologies. Monopolizing achieves 30% improvement in both FBFLY and MECS while Channel-stealing gets 40%. An exception comes from the MECS topology with the Fair-sharing scheme. Unlike FBFLY, MECS does not have any replicated channel, which implies all the multi-hop channels share a physical channel. If a physical channel is used as a two-hop channel, it cannot be used as a one-hop or three-hop channel in the same cycle. This reduces the probability of sharing physical channels from different packets, which hurts the performance of Fair-sharing.

5.4 Sensitivity to Network Design Points

In this section, we present variations that provide insight into the performance of an APCR router in different environments. Observing that Channel-stealing delivers the best performance, we use Channel-stealing for our experiments in the sensitivity study. The key idea of an APCR router is that the flit size is smaller than the phit size. The effect of the ratio between the phit size and flit size on the performance is shown in Figure 12 (a) where "p/f_2" means the phit size is two times the flit size. In this situation, the output channel is divided into two sub-channels. We can see that from "p/f_2" to "p/f_4" the network throughput is almost doubled. However, as the value of "p/f" keeps on increasing, there is no significant improvement. The main reason is that the VC depth is fixed as four. Even though the router has many free sub-channels, such as 16 or 32, the maximum flits stored in a VC is four. Hence, VCs cannot provide enough flits to completely utilize the sub-channels.

To further study how the VC depth in the number of flits affects the performance of an APCR router, we simulate with different VC depths. Each output channel is divided into four sub-channels, which also means that the flit size is one quarter of the phit size. The number of VCs is also fixed as four. Figure 12 (b) summarizes our experiments. Firstly, we observe that increasing the VC depth improves the network throughput. For example, changing the VC depth from 2 to 4 causes the network throughput to be improved by more than 53%. From 4 to 8, we get around 60% improvement. However, if we further increase the VC depth from 8 to 16, the improvement is marginal. One reason for this can be deduced from the packet size. In our experiment, the maximum packet size is five flits. VCs, whose depth is 8 or 16, can always hold one whole packet. The number of credits provided by the downstream

[6]We observe that other traffic patterns have the same trend.

router can also be greater than five. However, due to the regulation schemes, flits belonging to different packets cannot be transmitted in the same cycle. Therefore the maximum number of flits a VC can send is no more than five. Another reason is from the number of sub-channels. Since there are only four sub-channels, the output channel can only process four flits in a cycle. Thus, even with a long VC depth, the network throughput cannot be improved further.

6. CONCLUSIONS

Abundant wires inside a chip infer that it is important to organize this huge wiring capability efficiently. In this paper, we propose an Adaptive Physical Channel Regulator (APCR) for NOC routers to efficiently allocate the huge wiring resources. By defining the small flit, an APCR router allows flits from different packets or flows to share the same output channel in a single cycle. The three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) intelligently allocate the output channel resources considering not only the availability of physical channels but the occupancy of input buffers. Among the three schemes, Channel-stealing provides the most flexible usage of wide channels and, thus, achieves the best performance. An APCR router allows VCs to transmit multiple flits every cycle. The number of flits being transmitted every cycle dynamically changes according to the network status. Our simulation results using a detailed cycle-accurate simulator show that an APCR router improves the network throughput by over 100% in synthetic workloads. An APCR router can still outperform the baseline router when the buffer size is halved. It is also observed that an APCR router improves the network performance regardless of the underlying topologies.

7. REFERENCES

[1] Specjbb 2000 Benchmark. http://www.spec.org/jbb2000/.

[2] Specomp 2001 Benchmark Suite. http://www.spec.org/omp/.

[3] J. D. Balfour and W. J. Dally. Design Tradeoffs for Tiled CMP On-Chip Networks. In *ICS*, pages 187–198, 2006.

[4] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *PACT*, pages 72–81, 2008.

[5] B. Cuesta, A. Robles, and J. Duato. An Effective Starvation Avoidance Mechanism to Enhance the Token Coherence Protocol. In *PDP*, pages 47–54, 2007.

[6] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proceedings of DAC*, pages 684–689, 2001.

[7] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[8] R. Das, S. Eachempati, A. K. Mishra, N. Vijaykrishnan, and C. R. Das. Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs. In *HPCA*, pages 175–186, 2009.

[9] M. Galles. Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER chip. In *Proceedings of Hot Interconnect 4*, pages 141–146, 1996.

[10] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express Cube Topologies for On-Chip Interconnects. In *HPCA*, pages 163–174, 2009.

[11] B. Grot, S. W. Keckler, and O. Mutlu. Topology-aware Quality-of-Service Support in Highly Integrated Chip Multiprocessors. In *WIOSCA*, 2010.

[12] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. In *Proceedings of the IEEE*, pages 490–504, 2001.

[13] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration. In *DATE*, pages 423–428, 2009.

[14] C. Kim, D. Burger, and S. W. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. In *ASPLOS*, pages 211–222, 2002.

[15] J. Kim, J. Balfour, and W. J. Dally. Flattened Butterfly Topology for On-Chip Networks. In *Proceedings of MICRO*, pages 172–182, 2007.

[16] D. Kroft. Lockup-Free Instruction Fetch/Prefetch Cache Organization. In *ISCA*, pages 81–88, 1981.

[17] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, and D. Verkest. Spatial Division Multiplexing: A Novel Approach for Guaranteed Throughput on NoCs. In *CODES+ISSS*, pages 81–86, 2005.

[18] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A Full System Simulation Platform. *IEEE Computer*, 35(2):50–58, 2002.

[19] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *SIGARCH Computer Architecture News*, 33(4):92–99, 2005.

[20] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das. A Case for Dynamic Frequency Tuning in On-Chip Networks. In *MICRO*, pages 292–303, 2009.

[21] R. D. Mullins, A. West, and S. W. Moore. Low-Latency Virtual-Channel Routers for On-Chip Networks. In *Proceedings of ISCA*, pages 188–197, 2004.

[22] R. D. Mullins, A. West, and S. W. Moore. The Design and Implementation of a Low-Latency On-Chip Network. In *ASP-DAC*, pages 164–169, 2006.

[23] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh. Research Challenges for On-Chip Interconnection Networks. *IEEE Micro*, 27(5):96–108, 2007.

[24] L.-S. Peh and W. J. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proceedings of HPCA*, pages 255–266, 2001.

[25] C. G. Requena, M. E. Gómez, P. López, and J. Duato. Exploiting Wiring Resources on Interconnection Network: Increasing Path Diversity. In *PDP*, pages 20–29, 2008.

[26] C. G. Requena, M. E. Gómez, P. J. L. Rodríguez, and J. Duato. An Efficient Switching Technique for NoCs with Reduced Buffer Requirements. In *ICPADS*, pages 713–720, 2008.

[27] Y. Tamir and G. L. Frazier. High-Performance Multi-Queue Buffers for VLSI Communication Switches. In *Proceedings of ISCA*, pages 343–354, 1988.

[28] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. CACTI 5.1. Technical report, HP Laboratories, 2008.

[29] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *ISCA*, pages 24–36, 1995.

[30] A. Yakovlev, A. Koelmans, and L. Lavagno. High-Level Modeling and Design of Asynchronous Interface Logic. *IEEE Design & Test of Computers*, 12(1):32–40, 1995.