ACCELERATING COMMUNICATION IN ON-CHIP INTERCONNECTION NETWORKS

A Dissertation

by

MIN SEON AHN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

ACCELERATING COMMUNICATION IN ON-CHIP INTERCONNECTION NETWORKS

A Dissertation

by

MIN SEON AHN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

ABSTRACT

Accelerating Communication in On-Chip Interconnection Networks. (May 2012)

Min Seon Ahn, B.S., Korea Advanced Institute of Science and Technology;

M.S., Korea Advanced Institute of Science and Technology

Chair of Advisory Committee: Dr. Eun Jung Kim

Due to the ever-shrinking feature size in CMOS process technology, it is expected that future chip multiprocessors (CMPs) will have hundreds or thousands of processing cores. To support a massively large number of cores, packet-switched on-chip interconnection networks have become a de facto communication paradigm in CMPs. However, the on-chip networks have their own weakness, such as limited on-chip resources, increasing communication latency, and insufficient communication bandwidth. High performance in on-chip interconnection networks must be achieved by challenging the weakness.

In this dissertation, three schemes are proposed to accelerate communication in on-chip interconnection networks within area and cost budgets to overcome the weakness. First, an early transition scheme for fully adaptive routing algorithms is proposed to improve network throughput. Within a limited number of resources, previously proposed fully adaptive routing algorithms have low utilization in escape channels. To increase utilization of escape channels, it transfers packets earlier before the normal channels are full. Second, a pseudo-circuit scheme is proposed to reduce network latency using communication temporal locality. Reducing per-hop router delay becomes more important for communication latency reduction in larger on-chip interconnection networks. To improve communication latency, the previous arbitration information is reused to bypass switch arbitration. For further acceleration, we also propose two aggressive schemes, pseudo-circuit speculation and buffer bypassing. Third, two handshake schemes are proposed to improve network throughput for nanophotonic interconnects. Nanophotonic interconnects have been proposed to replace metal wires with optical links in on-chip interconnection networks for low latency and power consumptions as well as high bandwidth. To minimize the average token waiting time of the nanophotonic interconnects, the traditional credit-based flow control is removed. Thus, the handshake schemes increase link utilization and enhance network throughput.

To my wife, Joonhee, my daughter, Helen, and my parents for their support

ACKNOWLEDGEMENTS

First, I would like to thank my adviser, Dr. Eun Jung Kim, for her guidance and patience. Her advice made it possible for me to complete my Ph.D. degree at Texas A&M University. I also would like to thank my committee members, Dr. Rauchwerger, Dr. Walker, and Dr. Choi. I also thank Dr. Ki Hwan Yum for his support throughout the course of this research during my Ph.D. years, especially for his careful reading and comments on papers.

Second, I would like to thank all of the previous and current members of Dr. Kim's High Performance Computing Laboratory, especially Yuho Jin, Manhee Lee, Baiksong An, Lei Wang, and Jagadish Chandar Jayabalan for supporting and helping my research.

Lastly, I would like to thank my wife, Joonhee, for continuing support, patience, and love, and for believing in me while I studied in a small town, College Station, TX.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

It is projected that a future chip multiprocessor (CMP) will have more than hundreds or thousands of processor cores in a single chip due to the ever-shrinking feature size in CMOS process technology [1]. To support a massively large number of cores, a packet-switched on-chip interconnection network has become a de facto communication paradigm in CMPs because the shared bus architecture has a scalability problem. There have been several previous studies in on-chip interconnection networks, such as Intel 80-core Teraflop [2], Tilera 64-core [3], TRIPS [4], and RAW [5].

The on-chip interconnection networks have their own weakness. First, the network resources, such as buffers and virtual channels, are limited in CMPs. Since these resources are consumed by processing cores, caches, memory controllers, routers, and on-chip networks, it is imperative to maximize the utilization of every resource. Second, the communication latency becomes larger due to the increasing size of the network. As the number of processing cores in a single chip is growing, the size of the on-chip interconnection network is also increasing to provide connectivity to all cores. Since the number of hops in packet-switched networks dominates the communication latency, it is unavoidable to suffer from long latency in on-chip interconnection networks. Third, on-chip interconnection networks need high bandwidth as the number of

---

This dissertation follows the style of *IEEE Transactions on Parallel and Distributed Systems*.

processing cores is increasing. However, metal wires in the current design suffer from huge power consumption. To provide the sufficient bandwidth for communications, several alternatives have been proposed, but nanophotonic interconnects are the only candidate supporting high bandwidth density, low communication latency, and low power consumption.

First, interconnection networks have become prevalent not only in massively parallel processing systems, but also in CMPs. Unlike off-chip interconnection networks, on-chip interconnection networks have limited network resources due to a limited number of transistors in a single chip. These transistors are consumed not only for processing cores but also for on-chip interconnection networks. Thus, this limitation mandates to choose simple routing algorithms, which, in turn, provide low throughput. Fully adaptive routing algorithms improve throughput, but need escape channels for a deadlock recovery technique resulting in low utilization. To improve utilization, we propose an early transition scheme where packets are transferred to the escape channels earlier, before the normal channels are full. Our evaluation results using a cycle-accurate network simulator show that this scheme improves network throughput up to 12% in a concentrated mesh, compared to Duato's fully adaptive routing algorithm [6]. Because the proposed scheme has better utilization in the escape channels, the early transition scheme is less sensitive to the ratio of the number of the escape channels to the number of the total channels.

Second, it is well known that packet-switched networks suffer from high communication latency due to the increasing number of hops. To overcome the latency problem, we attempt to accelerate network communication by exploiting communication temporal locality with minimal additional hardware cost in the existing state-of-the-art router architecture. Communication temporal locality is a repeated communication pattern traversing through the same path, such as frequent pair-wise communication. This locality can be observed even in a crossbar connection pattern. Motivated by this observation, we propose a pseudo-circuit scheme. With the previous communication pattern, the scheme reserves previous crossbar connections creating pseudo-circuits, sharable partial circuits within a single router. It reuses the previous arbitration information to bypass switch arbitration if the next flit traverses through the same pseudo-circuit. For further communication acceleration, we also propose two aggressive schemes; pseudo-circuit speculation creates more pseudo-circuits using unallocated crossbar connections and buffer bypassing allows flits to skip buffer writes to eliminate one pipeline stage. We evaluate this scheme with traces from SPEComp2001, PARSEC, NAS Parallel Benchmarks, SPECjbb2000, and Splash-2, showing 16% improvement in overall communication latency and up to 5% reduction in average energy consumption in routers. Evaluated with synthetic workload traffic, the simulation results show up to 11% latency improvement. Unlike previous techniques in the low-latency router design, the proposed pseudo-circuit scheme can be applicable to any topology, such as Multidrop Express Cube or Flattened Butterfly, improving communication latency by up to 20%.

Third, nanophotonic interconnects have been proposed to design low latency and high bandwidth as well as low power consumption independently from the

communication length. Recent nanophotonic Network-on-Chip (NOC) designs hire token-based arbitration coupled with credit-based flow control, which leads to low bandwidth utilization. To increase the utilization, we propose two handshake schemes for nanophotonic interconnects, Global Handshake (GHS) and Distributed Handshake (DHS). These two schemes get rid of the traditional credit-based flow control, reduce the average token waiting time, and finally improve the network throughput. For further enhancement, we use setaside buffers and circulation techniques to solve the Head-of-Line (HOL) blocking problem. Our evaluation shows that the proposed handshake schemes improve network throughput by up to 11× under synthetic workloads. With the extracted trace traffic from real applications, the handshake schemes can reduce the communication latency by up to 55%. The handshake schemes add only 0.4% hardware overhead for optical components and negligible power consumption. In addition, the performance of the handshake schemes are independent of on-chip buffer space, which makes them feasible in a large scale nanophotonic interconnect design.

CHAPTER II

ON-CHIP INTERCONNECTION NETWORKS

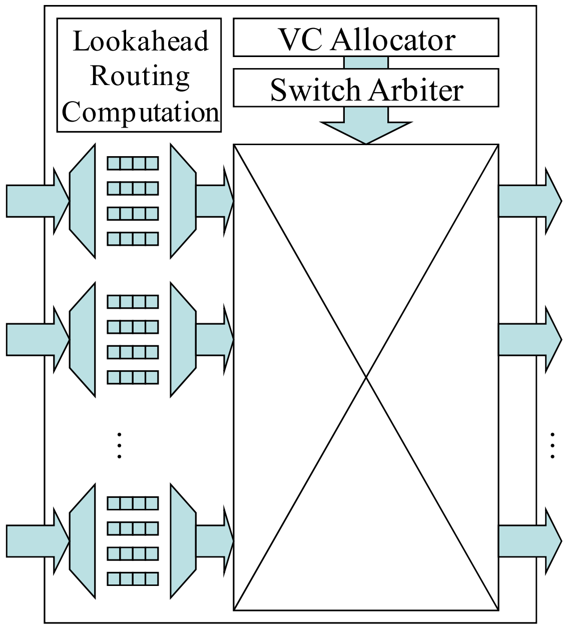


Figure 1. Baseline Router Microarchitecture

Throughout this dissertation, we use a state-of-the-art router architecture [7] as shown in Figure 1. It has 2 pipelined stages performing virtual-channel allocation (VA) and switch arbitration (SA) at the first stage, and switch traversal (ST) at the second stage. Routing calculation (RC) is removed from the critical path by adopting lookahead routing [8] that generates routing information of the downstream router. SA is speculatively performed in parallel with VA. The VC allocator logic allocates one available VC at the input port of the downstream router. The switch arbiter logic arbitrates input and output ports of the crossbar. When a flit enters a router, it should be written into the buffer in one of VCs of the input port during buffer write (BW). Once a flit is granted in SA stage, it enters the crossbar. After the flit traverses through the crossbar, it is sent to the downstream router during link traversal (LT). We assume link traversal takes one cycle.

Each router has multiple VCs per input port for low Head-of-Line blocking. It uses flit-based wormhole switching [9] and credit-based VC flow control [10] for a small buffer cost to minimize the area cost in on-chip interconnection networks. This flow control provides back-pressure from downstream routers to upstream routers to avoid buffer overflow.

Communication messages are transmitted as packets. A sender network interface (NI) splits a packet into multiple flits to fit in the communication bandwidth for flow control and injects them serially. At the receiver NI, flits are restored to a packet after receiving all flits. The first flit of a packet is called a header flit, where routing information is stored, and the last flit is a tail flit. The other flits are called body flits. Once the header flit is routed to its destination according to its routing information, the remaining flits follow the header flit from source to destination.

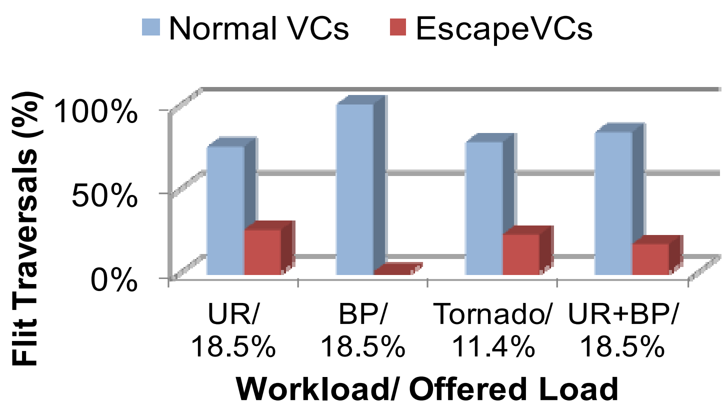
CHAPTER III

EARLY TRANSITION FOR FULLY ADAPTIVE ROUTING ALGORITHMS

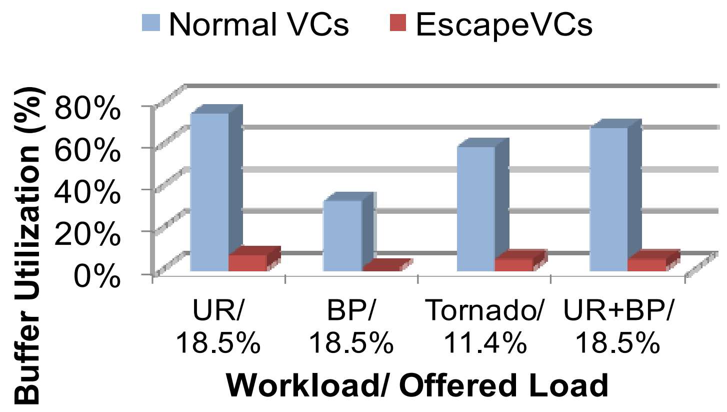
A. Introduction

Interconnection networks have been developed in massively parallel multiprocessor systems to connect processors, memories, and I/O devices. As the feature size keeps shrinking in CMOS process technology, chip multiprocessors (CMPs) are projected to have more than hundreds of processing cores in a single chip [1]. To support a large number of processing cores, interconnection networks replace shared buses due to the scalability problem. There are many studies on CMPs using on-chip interconnection networks, such as Intel 80-core Teraflop [2], Tilera 64-core [3], TRIPS [4], and RAW [5]. Compared to off-chip interconnection networks, on-chip interconnection networks have a limited area and power budget in a single chip. This constraint limits the number of hardware resources in the on-chip interconnection networks, such as the number of buffers and the number of virtual channels (VCs).

Because of the limited amount of hardware resources in on-chip interconnection networks, simple routing algorithms have been widely used, such as dimension order routing algorithms and O1TURN [11]. However, these routing algorithms suffer from poor network throughput, especially when hotspots exist. Theoretically, adaptive routing algorithms provide better performance than deterministic and oblivious routing algorithms because traffic is adaptively distributed around hotspots using network status information. To maximize adaptiveness, fully adaptive routing algorithms use all possible output ports as routing candidates without any restriction, thus resulting in better throughput.



(a) Flit Traversals



(b) Buffer Utilization

Figure 2. Traffic and Utilization in Duato's Fully Adaptive Routing Algorithm

To recover from a deadlock, the routing algorithms use escape channels [6], which occupy a small number VCs. These VCs are not utilized until a deadlock is detected. This condition causes low utilization of the escape channels because a deadlock can be detected only if the normal channels are full. Figure 2 shows that the normal channels have 58% utilization per VC, while the escape channels have less than 4% utilization per VC. If a network can provide many VCs, assigning a few VCs to the escape channels does not critically reduce the overall utilization. If, for example, off-chip interconnects can provide 16 VCs and 2 VCs are allocated to the escape channels, the average utilization is 51%, which is 7% lower than the utilization of the normal channels. Unlike off-chip interconnection networks, on-chip interconnection networks have limited numbers of VCs due to the limited resources. If there are only 4 VCs and the escape channels occupy 2 VCs, the average utilization can be 31%, which is around half of the utilization of the normal channels. Therefore, it is imperative to revisit the fully adaptive routing algorithm design in on-chip interconnection networks to improve buffer utilization.

In this chapter, we propose an early transition scheme to increase the utilization of the escape channels. Our main idea is to use the escape channels not only for deadlock recovery but also for load-balancing. Packets are transferred to the escape channels if the queue occupancy of the normal channels is larger than the queue occupancy of the escape channels, which still guarantees that the routing algorithm is deadlock-free. By increasing the utilization of the escape channels, the early transition scheme improves network throughput.

Our evaluation results using a cycle-accurate network simulator show that the proposed early transition scheme increases network throughput by up to 12% compared to Duato's fully adaptive routing algorithm [6] under synthetic workload traffic in a concentrated mesh [12]. The escape channels support 13% ~ 24% more traffic and provide at least doubled buffer utilization. Performance is also improved up to around 8% in a flattened butterfly [13, 14] and a generic mesh. The proposed scheme has no remarkable performance degradation even with the larger ratio of the escape channels to the total channels, showing that it is less sensitive to the number of the escape channels than Duato's fully adaptive routing algorithm.

The reminder of this chapter is organized as follows. We briefly discuss related work in Section B. After providing background information on this work in Section C, we describe the proposed scheme in Section D. In Section E, we illustrate the evaluation methodology, followed by presenting simulation results in Section F. Finally, we conclude this chapter in Section G.

B. Related Work

To provide better throughput in the networks, there are many studies on deadlock prevention and deadlock-free routing algorithms in 2D mesh networks. The turn model [15] is introduced for deadlock avoidance of adaptive routing algorithms in mesh topologies. By limiting some turns, routing algorithms are completely free from deadlocks without any deadlock recovery scheme. Furthermore, PFNF (Positive First Negative First) [16] achieves better performance by extending adaptiveness with two turn models, one in each virtual network separately because it efficiently distributes all traffic symmetrically. Similarly, O1TURN [11] has two virtual networks each of which uses a different dimension order routing algorithm. These routing algorithms are either a partial adaptive routing algorithm or an oblivious routing algorithm, resulting in limited throughput improvement.

Deadlock-free fully adaptive routing algorithms have been studied to maximize adaptiveness. Duato [6] first proposes to use extra escape channels to prevent deadlocks. Basically, messages traverse through unrestricted normal channels until they are full. Thus, the escape channels suffer from low utilization. 3p routing algorithm [17] is the most recently proposed fully adaptive routing algorithm in 2D meshes, which divides VCs into two classes, waiting and non-waiting. It is deadlock-free because the waiting channels have two separate networks, positive and negative, and there is no cycle on channel dependency graphs in each network. However, this fully adaptive routing algorithm may have traffic imbalance because two separate networks are dependent on traffic directions.

There are several studies on fully adaptive routing algorithms in other topologies not applicable to on-chip interconnection networks. GOAL [18] and UGAL [19] are load-balanced fully adaptive routing algorithms in tori and hypercubes to achieve better network throughput by balancing the network workload into minimal and non-minimal paths. More recently, Kim et al. [20] propose adaptive routing algorithms in a folded-Clos network with high-radix routers [21]. These routing algorithms also provide load-balancing between multiple minimal paths in the network. They also propose the precision reduction of network information to minimize the hardware overhead of adaptive routing algorithms and pre-computation to minimize the impact on the router pipeline delay without significant performance degradation. Jiang et al. [22] propose several indirect ways of adaptive routing algorithms, such as credit round trip, progressive adaptive routing, piggyback routing, and reservation routing.

C. Deadlock Prevention

A deadlock occurs when some packets in the network are blocked infinitely without any advance. If packets are waiting for network resources consumed by other packets, it makes a dependency. If the dependency becomes a cycle, it makes a deadlock. There are two techniques [23] to prevent deadlocks in routing algorithms. First, deadlock avoidance prohibits some turns to avoid cyclic dependency. Turn model [15] prevents some turns in routing algorithms to avoid circular dependency in mesh networks. The other turns are still allowed to the adaptive routing algorithms making routing algorithms partially adaptive.

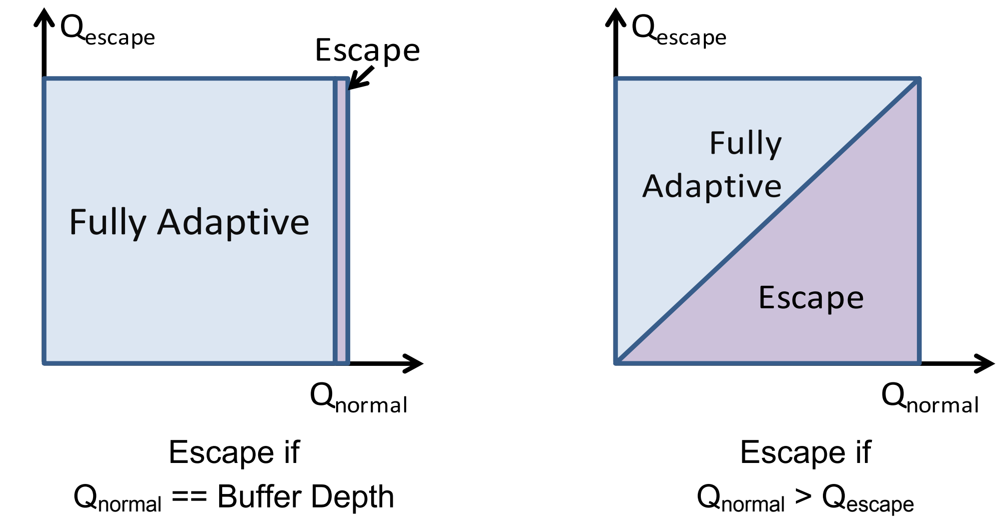
The second technique is deadlock recovery which breaks the cyclic dependency when a deadlock is detected. Deadlock recovery techniques allow routing algorithms to make any turn to all possible directions until a possible deadlock is detected. Duato [6] proposes a simple and necessary condition to detect a deadlock. It simply checks if the normal channels are full. If so, there is a possibility of deadlocks. Otherwise, no deadlock happens. Once this conservative deadlock detection condition is satisfied, the packet is removed from the normal channels to the restricted escape channels to recover from a deadlock.

D. Early Transition for Fully Adaptive Routing Algorithms

In this section, we describe an early transition scheme for fully adaptive routing algorithms in on-chip interconnection networks to improve network throughput.

1. Increasing Escape Channel Utilization

Virtual channels [10] are proposed to reduce head-of-line (HOL) blocking by virtually dividing one physical channel into several VCs. These virtual channels are also used to provide separate virtual networks for either deadlock avoidance (PFNF [16] or O1TURN [11]) or deadlock recovery (escape channels in Duato's fully adaptive routing algorithm [6]).



(a) Duato's Fully Adaptive Routing Algorithm

(b) Early Transition

Figure 3. Flit Transition to Escape Channels

Fully adaptive routing algorithms generally use a deadlock recovery technique by adopting escape channels where routing algorithms must be deadlock-free [6], such as dimension order routing (DOR) algorithms. Generally, a deadlock could happen only if there is no available buffer space in the normal virtual channel. Therefore, the escape channels are not utilized until there is a possibility of a deadlock. In the previous deadlock recovery scheme, the escape channels have low utilization because packets are not traversing until the unrestricted normal channels for adaptive routing algorithms are full as shown in Figure 3 (a). We observe that this could cause low utilization in the escape channels, resulting in earlier network saturation.

To solve this early saturation problem, we propose an early transition scheme as shown in Figure 3 (b). To increase the utilization of the escape channels, it transfers packets to the escape VCs earlier, before the normal channels are full. Packet transfer is performed when the queue occupancy of the escape VCs is smaller than the queue occupancy of the unrestricted normal VCs.

There are two conditions to make fully adaptive routing algorithm deadlock-free [6]. First, the routing algorithm used in the escape channels must be deadlock-free. Second, the deadlock detection condition necessarily detects deadlocks. The proposed early transition scheme still uses deterministic routing algorithms, which satisfies the first condition. The second condition is also satisfied since it transfers packets to the escape channels when the normal channels are full in the proposed scheme. Therefore, it makes the routing algorithms still deadlock-free.

To achieve further load-balancing in all directions, we use both DOR algorithms like O1TURN [11] in the escape VCs instead of one DOR algorithm. We observe that using a simple DOR algorithm causes non-uniformity in the network physical channels of the escape channels. Unlike the DOR algorithm, O1TURN achieves load-balancing by using two orthogonal DOR algorithms (XY and YX) together. To avoid deadlocks, it partitions VCs into two virtual networks and each DOR algorithm is assigned to each virtual network. To accommodate two DOR algorithms in the escape channels, all escape channels must be partitioned into two, each of which uses one DOR algorithm independently. When a packet enters the escape VCs, one of two virtual networks is randomly selected and the same DOR algorithm is used in the same virtual network until the packet arrives at the destination.

2. Reducing the Overhead of Adaptive Routing Algorithms

To make a fair comparison with other deterministic or oblivious routing algorithms, the computational overhead of adaptive routing algorithms is minimized by reducing the precision in credit information and pre-computing the allocation [22]. First, precision reduction in credit information has a little performance degradation compared to full precision. If each virtual channel has n-flit buffers, $log_2\ n$ bits are needed for credit information. Since the buffer depth in each VC is 4 in our experiment, the number of bits in the credit information is still minimal. In addition, the credit information is changing by 1 per cycle. There are not many changes within a couple of cycles. Therefore, applying adaptiveness in lookahead routing has a minimal hardware overhead. Another way to reduce the computation overhead is pre-computation. With the credit information of the previous cycle, the routing calculation can be pre-computed before a packet is arriving. Basically pre-computation loss is only minimal because the difference of the credit information is maximally just one per cycle. Kim et al. [20] show that reducing the precision in credit information and pre-computing minimize the computation costs without significant performance degradation compared to the full precision and no pre-computation.

Fully adaptive routing algorithms check a deadlock detection condition during VA stage. If the condition is satisfied, it moves packets to one of escape channels after performing escape RC/VA. To avoid performance overhead, escape RC/VA can be performed in parallel with VA. If the deadlock condition is not satisfied, escape RC/VA information is ignored and unused. Since the routing algorithm used in the escape channels is simple compared to the fully adaptive routing algorithm, the hardware overhead is minimal, compared to the fully adaptive routing algorithm.

E. Experimental Methodology

We evaluate performance using our cycle-accurate on-chip network simulator implementing pipelined routers with buffers, VCs, arbiters, and crossbars. The network simulator is configured with 4-flit buffer per each VC and 4 VCs per each input port. We assume that the bandwidth of a link is 128 bits with additional error correction bits. We use a concentrated mesh topology [12] for on-chip interconnection networks where each router connects 4 communication nodes and routers are connected as a 2D mesh topology. In this topology, the number of communication nodes is 64, and the number of routers is 16.

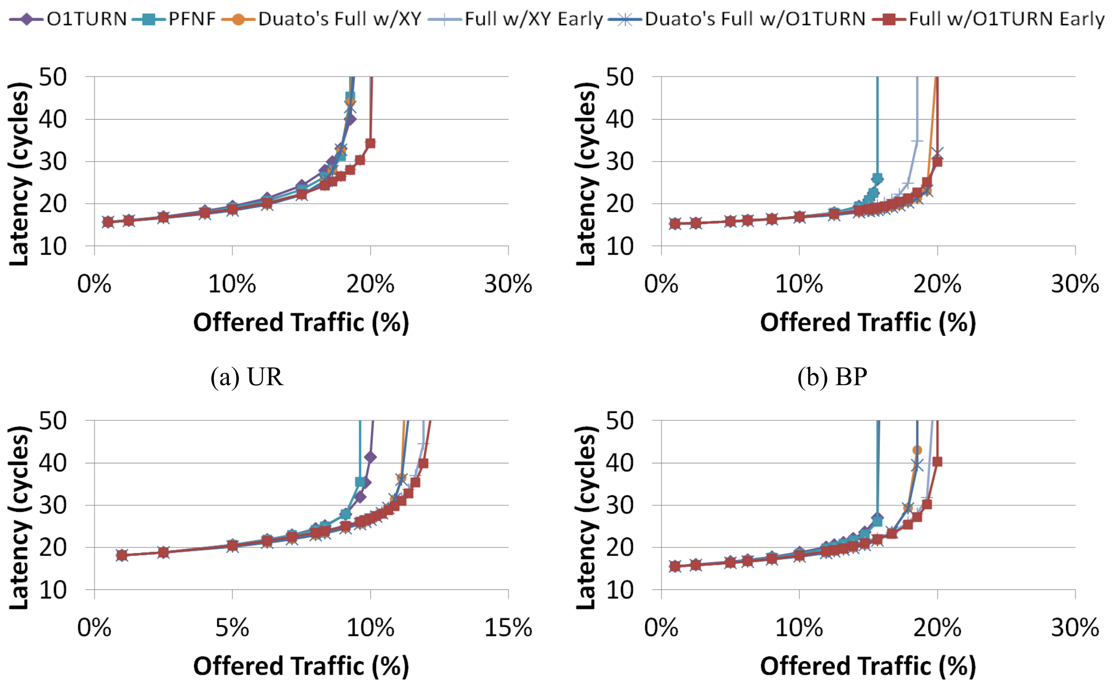
To compare with the most recently proposed routing algorithms in 2D meshes, we use O1TURN [11] and PFNF [16]. For fair comparison to O1TURN, we use minimal adaptive routing algorithms which select the best candidate among those in the shortest paths only. We also use dynamic VA policy to maximize network throughput and latency. It chooses an output VC where the number of available buffers is the greatest among all possible VCs in the downstream router. If there is no available buffer in all candidate output VCs, VA fails. To select the best VC, we use credit information. Thus, it has evenly distributed workload between VCs of the same kind. This allocation policy is generally used in on-chip interconnection networks.

To apply two DOR algorithms in the escape channels, we need at least two VCs in the escape channels. Since there are 4 VCs per input port, we assign 2 VCs to the normal channels and the other 2 VCs to the escape channels. To make a fair comparison, we assign the same number of VCs to the escape channels in every configuration of fully adaptive routing algorithms.

In this experiment, we use several different basic patterns of synthetic workload traffic. First, we use uniform random (UR) traffic, where the destination is uniformly distributed to all nodes in the network. Thus, it has equal chances to use all links. The second traffic is bit permutation (BP) whose communication pattern is generated based on matrix transpose. This traffic has only one destination for each source. Due to symmetry on patterns, it does not have any deadlock, but creates spatial hotspots in the diagonal line. The last traffic is tornado (Tornado), where all communications are going clockwise in 2D meshes and the destinations are 4-hop away, resulting in asymmetric usage of links. To see performance in complicated workload traffic, we also create one more pattern, mixing UR and BP together (UP+BP) to create complicated and dynamic hotspots in time and space. In the experiment with synthetic workload traffic, all packets are 5 flit long.

F. Performance Evaluation

In this section, we evaluate the proposed load-balancing techniques. To compare performance with other routing algorithms, we also use two previously proposed routing algorithms, such as O1TURN [11] and PFNF [16]. In this section, the fully adaptive routing algorithm using XY in the escape channels is indicated by **Full with XY**, the fully adaptive routing algorithm using two DOR algorithms in the escape channels is indicated by **Full with O1TURN**.

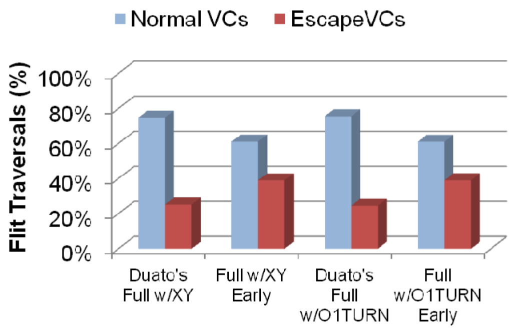


(a) UR (b) BP

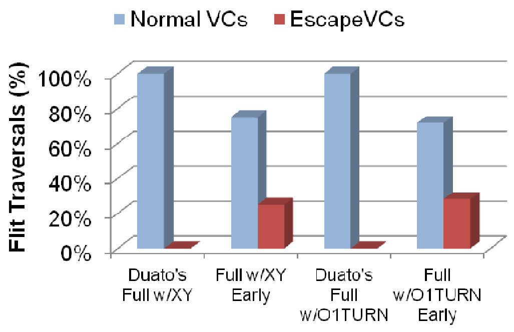
(c) Tornado (d) UR+BP

Figure 4. Overall Performance in a Concentrated Mesh

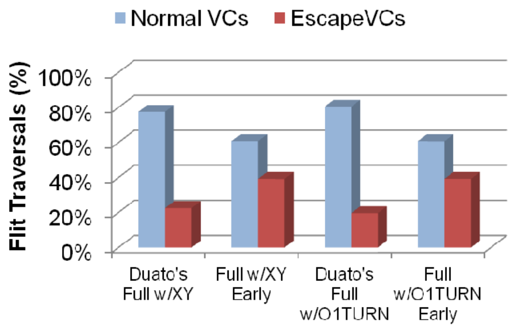
We evaluate performance of the proposed early transition scheme in load-balanced fully adaptive routing algorithms in a concentrated mesh [12] as shown in Figure 4. Since UR has equally distributed traffic to all possible directions, there is no performance difference between **Full with XY** and **Full with O1TURN**. Figure 4 (a) shows that the proposed early transition scheme increases network throughput by 12% normalized to Duato's routing algorithm. We observe that UR has temporal hotspots randomly with a small amount of traffic burst. BP has hotspots in the diagonal line because all traffic is concentrated to those hotspot routers. Thus, **Full with XY** using the early transition scheme has performance degradation compared to **Full with XY** using Duato's routing algorithm. However, **Full with O1TURN** has better performance as shown in Figure 4 (b) because O1TURN distributes traffic into both dimensions in the escape channels. The other two synthetic workload patterns show the performance difference in detail. Inherently, O1TURN has better performance than any other dimension order routing algorithm [11]. Consequently, the routing algorithm used in the escape channels affects network performance. Besides, the proposed early scheme has more flit traversals and increases the buffer utilization in the escape channels as shown in Figure 5 and Figure 6, respectively. Therefore, **Full with O1TURN** using the early transition scheme has the best performance because of the additional improvement in the escape channels. Overall, it achieves 12% normalized network throughput improvement in Tornado and UR+BP, compared to **Full with XY** using Duato's routing algorithm.



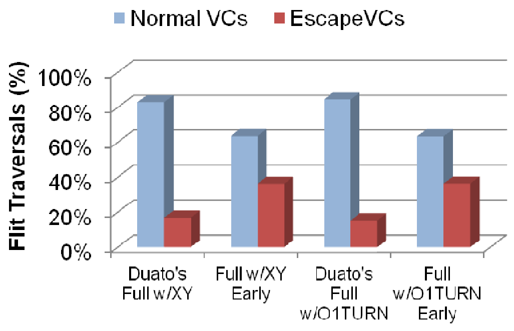
(a) UR 18.5%



(b) BP 18.5%



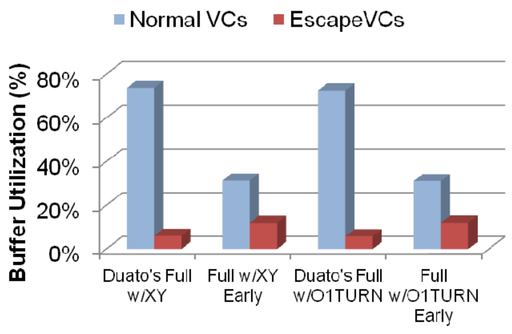
(c) Tornado 11.4%



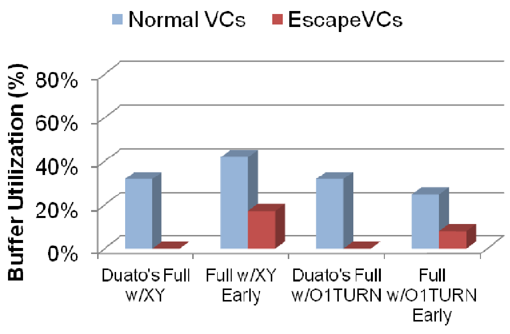
(d) UR+BP 18.5%

Figure 5. Flit Traversals on Workload and Offered Load

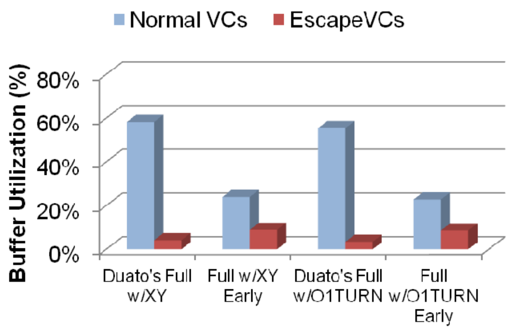
Figure 5 shows the percentage of flit traversals in the normal channels and the escape channels. The proposed early transition scheme increases traffic in the escape channels 13% ~ 24% compared to Duato's routing algorithm. In other words, the early transition scheme tries to evenly distribute traffic workload to both channels. Figure 6 shows the buffer utilization, the percentage of buffer occupancy on average. The early transition, at least, doubles the buffer utilization in the escape channels. Since the proposed early transition scheme does not transfer packets to the escape channels if both queue occupancies are equal. Thus, the normal channels still have higher buffer utilization than the escape channels. Note that packets are still traversing in the normal channels if both queue occupancies are equal. Therefore, the escape channels still have less utilization than the normal channels.



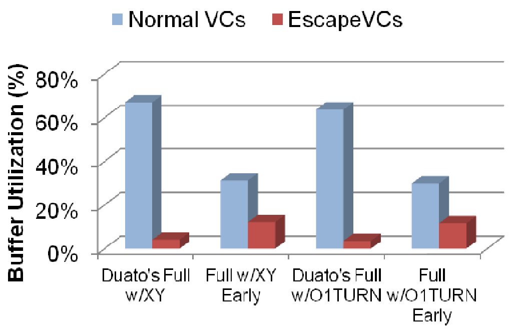
(a) UR 18.5%



(b) BP 18.5%



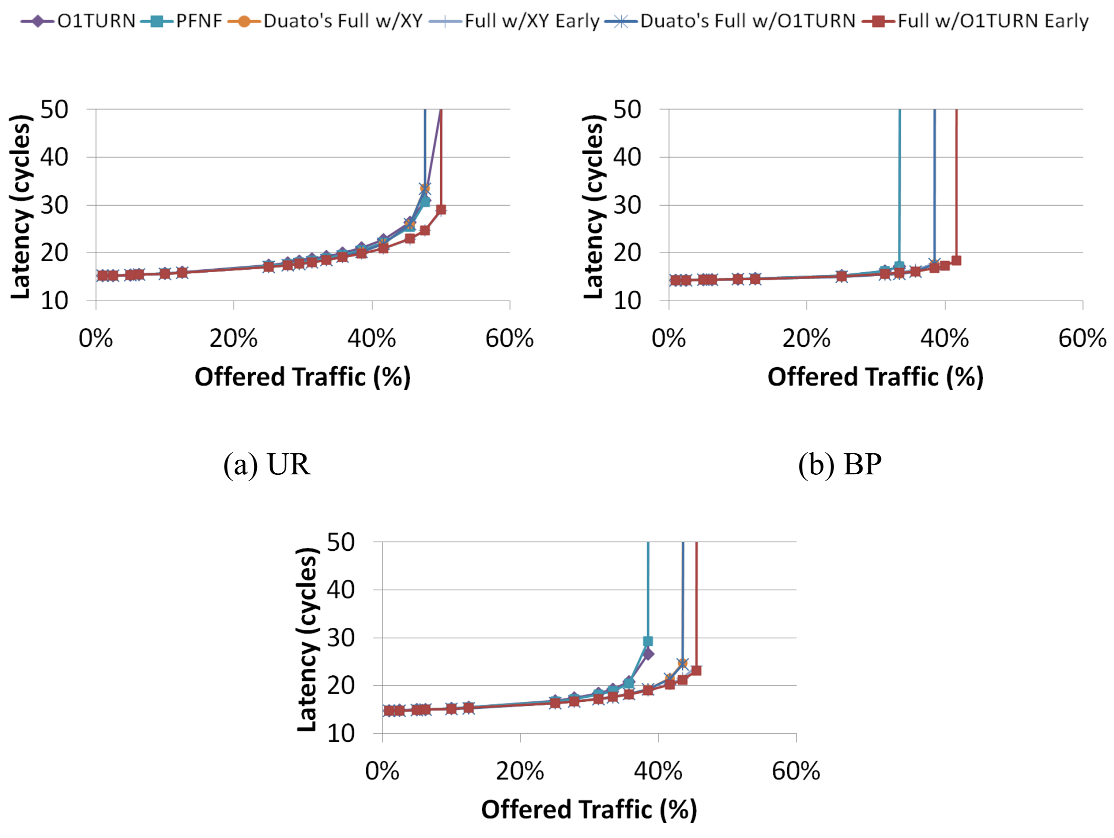
(c) Tornado 11.4%



(d) UR+BP 18.5%

Figure 6. Buffer Utilization on Workload and Offered Load

We evaluate performance of the proposed early transition in load-balanced fully adaptive routing algorithms in other topologies, such as a flattened butterfly [13, 14] and a general mesh. The flattened butterfly has the same concentration as the concentrated mesh. However, it has additional express channels unlike the concentrated mesh. Since we assume that each link has the same bandwidth as in the concentrated mesh, this topology can accept more communication. Because the express channels can bypass hotspots without any adaptiveness, performance improvement in Flattened Butterfly is smaller than in the concentrated mesh with UR and UR+BP traffic. However, BP has up to 8.3% throughput enhancement, compared to Duato's routing algorithm, because the express channel can separate flows in this synthetic workload reducing hotspot traffic as shown in Figure 7.

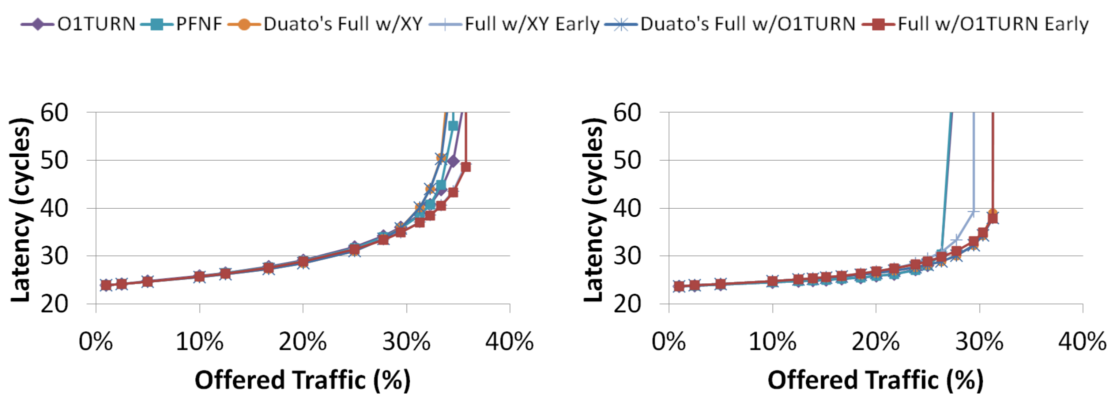


(a) UR (b) BP

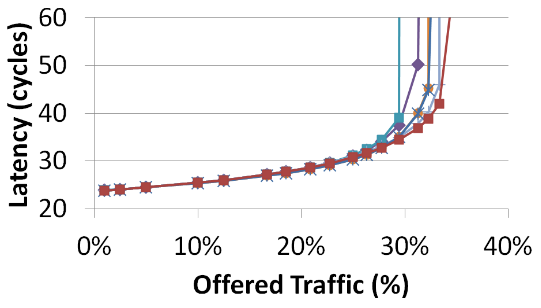
(c) UR+BP

Figure 7. Overall Performance in a Flattened Butterfly

To support the same number of cores, the mesh topology has 64 routers, building an 8x8 network. Thus, it has higher communication latency because of the larger average number of hops. Similarly in the concentrated mesh, **Full with XY** using the early transition scheme has performance degradation in BP, but **Full with O1TURN** using the early transition scheme still is as good as **Full with XY** using Duato's routing algorithm in this synthetic workload traffic. However, network throughput is improved with **Full with O1TURN** using the early transition scheme in UR and UR+BP by 7.14% and 3.35%, respectively, compared to **Full with XY** using Duato's routing algorithm as shown in Figure 8.



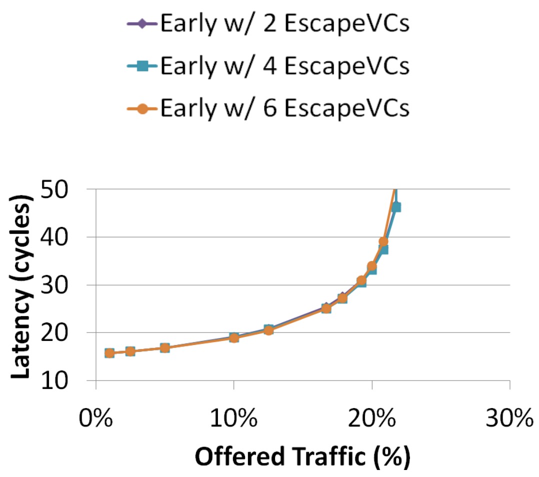
(a) UR (b) BP



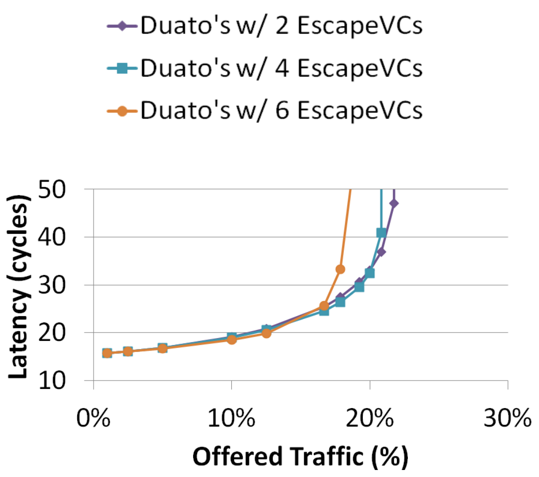
(c) UR+BP

Figure 8. Overall Performance in a Generic Mesh

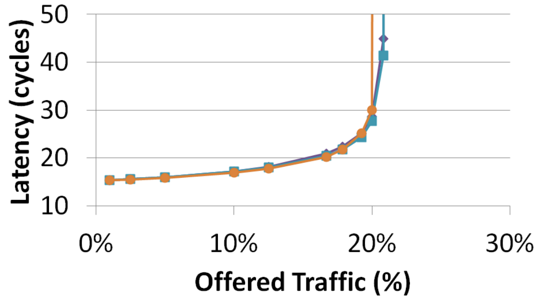
We also conduct an experiment with several different configurations on the number of VCs in the escape channels to see the sensitivity on the number of escape channels. In this experiment, we use fully adaptive routing algorithms in the normal channels and O1TURN in the escape channels, in a concentrated mesh. To make more configurations on the number of escape VCs, we increase the total number of virtual channels to 8 and generate 3 different configurations. The first configuration has 6 normal and 2 escape VCs. The second has 4 normal and 4 escape VCs. The last has 2 normal and 6 escape VCs. Figure 9 shows that the proposed early transition scheme has similar performance regardless of the number of the escape VCs whereas, Duato's routing algorithm has significant performance degradation when the number of the escape VCs is larger than that of the normal VCs. This is because the proposed scheme increases the utilization of the escape channels.



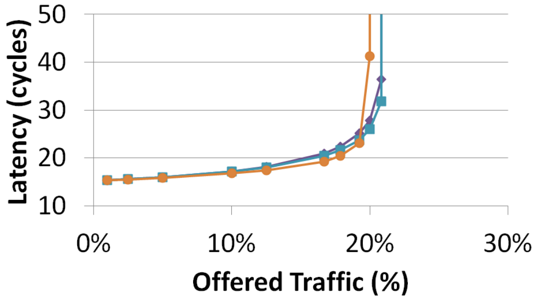
(a) UR in Early Transition



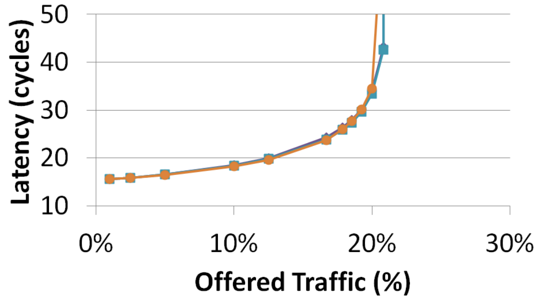
(d) UR in Duato's



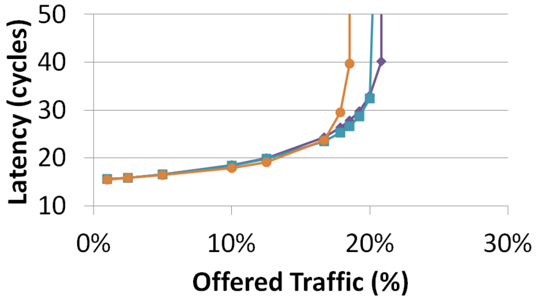
(b) BP in Early Transition



(e) BP in Duato's



(c) UR+BP in Early Traistion



(f) UR+BP in Duato's

Figure 9. Performance Sensitivity on the Number of Escape VCs

G. Conclusions

On-chip interconnection networks have been used in chip multiprocessors as communication architecture. Unlike off-chip interconnection networks, on-chip interconnection networks have limited resources due to area and power budget in a single chip. With the limited resources, on-chip interconnection networks have a small number of VCs. Consequently, fully adaptive routing algorithms with a deadlock recovery technique have low utilization. To increase the utilization, we propose an early transition scheme for fully adaptive routing algorithms in on-chip interconnection networks. This scheme moves packets to escape channels earlier, before the normal channels are full. Our cycle-accurate network simulator reveals that the proposed scheme improves network throughput up to 12% in a concentrated mesh because of better utilization of the escape channel than Duato's fully adaptive routing algorithm. It increases traffic in the escape channels up to 13% ~ 24% because of doubled buffer utilization. The proposed early transition scheme improves performance by around 8% in a flattened butterfly and a generic mesh. It becomes less sensitive to the number of VCs in the escape channels because the escape channels have better utilization.

To achieve perfect load-balancing between the escape channels and the normal channels, we will improve the early transition scheme for future work. Furthermore, we will also develop the early transition scheme to apply to other adaptive routing algorithms, such as UGAL in a flattened butterfly.

CHAPTER IV

LOCALITY-BASED ON-CHIP INTERCONNECTION NETWORK COMMUNICATION ACCELERATION

A. Introduction

Due to the ever-shrinking feature size in CMOS process technology [1], it is projected that a future chip multiprocessor (CMP) will have more than hundreds or thousands of processor cores in a single chip. However, the shared bus architecture cannot support a massively large number of processing cores because of a scalability problem. Thus, it is widely accepted that communication architecture for CMPs will be packet-switched on-chip interconnection networks. There are several previous researches on CMPs using on-chip interconnection networks, such as Intel 80-core Teraflop [2], Tilera 64-core [3], TRIPS [4], and RAW [5].

In the packet-switched network, the number of hops dominates communication latency. To provide high performance in communication, ultra-low per-hop router delay is required within a limited area budget and power constraints. There have been several studies on the low-latency router design in packet-switched networks to reduce per-hop router delay using speculation, pre-computation, or aggressive flow control [7, 24, 25, 26, 27]. However, the recent state-of-the-art router architecture has a complicated pipeline design to support various kinds of communications in on-chip networks. As a result, communication latency and power consumption are subject to the router overhead, such as additional hardware and link overhead.

We observe that every application has a certain amount of communication temporal locality, frequent pair-wise communication in the network. Circuit reusing [28] uses the communication locality to improve communication latency. Even more in each router, this communication temporal locality can be seen such that the recently used communication path from an input port to an output port can be reused. Figure 10 shows communication path reusability as communication temporal locality in end-to-end communication and in crossbar connections, which are links within a single router connecting input ports to output ports. It shows about 22% communication temporal locality in end-to-end communication because the locality can be reused only if both the source and the destination are the same as previous. However, communication temporal locality of crossbar connections is increased up to 31%. A crossbar connection can be more possibly reused by future flits even when the future flits traverse the same crossbar connection within a router.

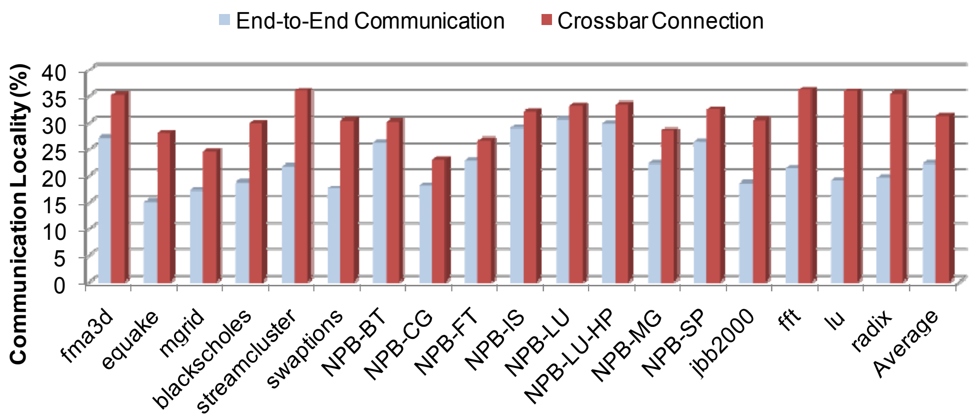


Figure 10. Communication Temporal Locality Comparison

This observation motivates locality-based communication acceleration in on-chip interconnection networks. We use the communication temporal locality of crossbar connection. If a flit traversal reuses the arbitration information created by previous flits, it does not need switch arbitration in the router pipeline, thus reducing per-hop router delay. Our main goal is bypassing router pipeline stages in packet-switched on-chip interconnection networks with minimal hardware addition by exploiting the reusability of the crossbar connections in each router to minimize per-hop router delay.

In this chapter, we propose a novel pseudo-circuit scheme for on-chip interconnection networks, which reuses the previous arbitration information if the next flit needs to traverse through the same crossbar connection within a single router. It enables the flit to be sent directly to the downstream router, bypassing the switch arbitration stage, thus reducing one pipelining stage in the router. A pseudo-circuit is defined as a currently available crossbar connection from an input port to an output port within a single router made by the switch arbiter using previous communication. It is created by a flit traversal within a single router and remains connected for future uses after the traversal. The pseudo-circuit is terminated when another recent flit claims either the input port or the output port. To avoid buffer full at the downstream router, a pseudo-circuit is also terminated when no credit of the downstream router is available. This scheme does not need any performance overhead to terminate a pseudo-circuit because pseudo-circuits are managed by crossbar switches within each router.

We also propose two aggressive pseudo-circuit schemes to accelerate communication latency further. First, pseudo-circuit speculation generates more pseudo-circuits with currently unallocated crossbar connections for the future communication based on history information. Second, buffer bypassing allows flits to skip buffer writes at input virtual channels (VCs), thus removing one more pipeline stage. These aggressive schemes increase pseudo-circuit reusability and decrease average per-hop router delay.

To show the effectiveness of the proposed pseudo-circuit scheme, we evaluate communication latency and energy consumption using our cycle-accurate flit-based wormhole switching on-chip network simulator with credit-based VC flow control. We use both traces and synthetic workloads for traffic models. Traces are extracted from SPEComp2001, PARSEC, NAS Parallel Benchmarks, SPECjbb2000, and Splash-2 on a self-throttling CMP network with 4 MSHRs (Miss Holding Status Register [29]) per processing core in Simics [30]. The pseudo-circuit scheme improves overall communication latency up to 16% with traces when combined with both aggressive schemes. It also saves about 5% of energy consumed in routers. Evaluated with synthetic workload traffic, it has latency improvement by up to 11%.

To provide more analysis on the sensitivity of the proposed scheme, we also examine the pseudo-circuit scheme with several different routing algorithms (Dimension order routing [31], O1TURN [11], PFNF [16]) and two VC allocation policies. We obtain the best performance with dimension order routing algorithms and a static VC allocation policy. When the pseudo-circuit scheme is applied to the recently proposed network topologies, such as Multidrop Express Cube [32] and Flattened Butterfly [14], our results show that it improves communication latency by up to 20%, resulting in more than 50% latency reduction compared to the baseline system with a mesh topology. Compared to HCS [28], the proposed scheme has more than 13% latency improvement in a single network. If multiple parallel networks are deployed, it achieves 4% more latency improvement with circuit reusing [28] due to 10% more pseudo-circuit reusability.

The remainder of this chapter is organized as follows. We briefly discuss related work in Section B. After presenting our pseudo-circuit scheme in Section C, we describe two aggressive schemes in Section D. After analysis on the sensitivity of the proposed pseudo-circuit scheme in Section E, we discuss evaluation methodology and present simulation results in Sections F and G, respectively. After discussing performance comparisons with other techniques in Section H, we conclude this chapter in Section I.

B. Related Work

To provide low latency in on-chip interconnection net-works, several techniques have been proposed to reduce per-hop router delay. A speculative pipelined router design [7] reduces 1 pipeline stage by parallelizing switch arbitration (SA) with virtual-channel allocation (VA) using speculation. A low-latency router [27] reduces per-hop router delay to fit in one cycle in low traffic by removing control overhead from the critical path using pre-computation. It has huge hardware overhead to avoid traversal in case of wrong pre-computation. SPAROFLO [24] switch allocation increases efficiency of crossbar connections by separating allocation mechanisms and prioritizing old requests. Token flow control [26] uses tokens to disseminate information about availability of network resources and lookahead link traversal to send a setup flit one cycle prior to flit traversal, thus resulting in bypass router pipelines with additional wiring overheads for tokens and lookahead link traversal. In Express Virtual Channel (EVC) [25], packets are virtually bypassing intermediate routers by reserving some VCs with higher priority than the other normal VCs. It forms an express virtual channel within a single dimension using latches in the intermediate routers, thus minimizing per-hop router delay when packets keep traversing in the same dimension.

Several topologies for on-chip interconnection net-works are proposed in [12] to reduce the average number of hops using high-radix routers [21]. Multiple independent parallel networks are also introduced in several topologies such as concentrated meshes to improve wire and area utilization [12]. By accommodating multiple injection channels, traffic can be evenly distributed into all parallel networks. Flattened butterfly [14] minimizes communication latency by providing express channels between nodes in the same dimension. A recently proposed express cube topology [32] uses multidrop express channels to send packets to the nodes in the same dimension in a bandwidth-efficient manner with overhead of intelligent repeaters in every junction.

Circuit-switched Coherence [28] proposes a hybrid flow control between packet-switching and circuit-switching, named hybrid circuit switching. This flow control builds a circuit using the setup network while sending messages in the packet-switching data network. Then, the circuit is reserved for future uses after the usage without termination. Since the data network is divided into two or more parallel networks, it increases the possibility to reuse the previously established circuits, thus reducing the average circuit setup time by amortizing the delay. However, it has limited improvement due to frequent partial circuit termination.

C. Pseudo-Circuit: Reusing Crossbar Connections

The key design goal of the proposed pseudo-circuit scheme is to reduce the communication latency by reusing crossbar connections established by previous flits. In this section, we present the pseudo-circuit scheme which reuses arbitration information created by previous communication.

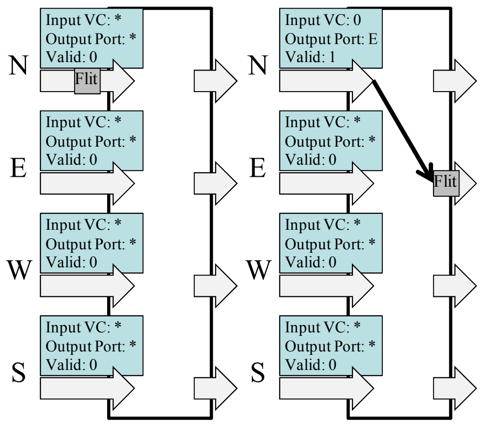
1. Pseudo-Circuit Creation and Reuse

A pseudo-circuit is defined as a currently available crossbar connection from an input port to an output port within a single router made by the switch arbiter using previous communication. Every flit traversal in a router creates a crossbar connection from an input port to an output port after arbitrated by the switch arbiter, which is written to the pseudo-circuit register in the input port. After the traversal, the pseudo-circuit remains connected for future uses until it is terminated.

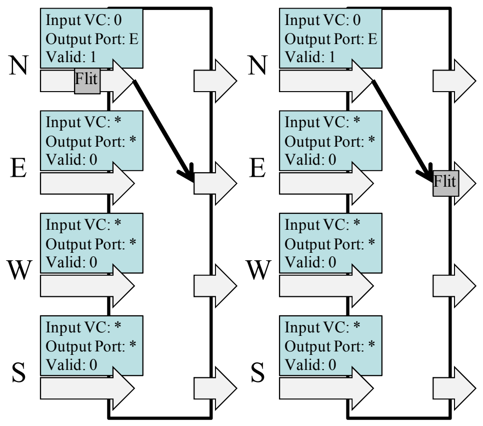
Figure 11 (a) shows pseudo-circuit creation by a flit traversal. The flit traversal from the input port N to the output port E creates a pseudo-circuit, and its information is written to the pseudo-circuit registry in the input port. If the next flit arriving at the same input VC in the same input port needs to traverse the same crossbar connection in the router as shown in Figure 11 (b), it simply goes directly to the crossbar bypassing SA because the crossbar connection is already set up and remaining connected. To bypass SA completely, each pseudo-circuit needs to hold the recent arbitration history of the switch arbiter. Since the first part of the switch arbiter arbitrates the input VCs, each pseudo-circuit should hold the recently used input VC number, so the next flit is expected to come to the same input VC in order to reuse the pseudo-circuit.

A pseudo-circuit is reused if a flit coming from the same VC at the input port has the same routing information. In order to check if the flit can traverse the pseudo-circuit, the router needs to compare the routing information of the flit with the pseudo-circuit information created by previous communication. It performs one simple comparison of routing information stored in input VCs. If the comparison logic asserts a matching signal, the flit can traverse the pseudo-circuit. If the routing information of the flit is different from the pseudo-circuit, the pseudo-circuit is terminated by the switch arbiter (See Section C.2). In this case, there is no performance overhead because the flit experiences the baseline pipeline stages. The routing information is always carried by header flits. Once the header flit has matching routing information with the pseudo-circuit, the following flits coming to the same VC can bypass SA without the routing information until the pseudo-circuit is terminated.

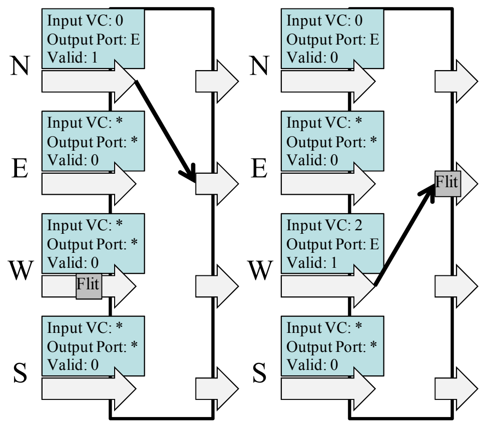
The pseudo-circuit remains connected unless there is another recent pseudo-circuit conflicting with the pseudo-circuit. If the pseudo-circuit is connected, it is ready to send flits directly to the downstream router through the crossbar without SA. Therefore, if the comparator generates a matching signal with the current pseudo-circuit, the flit can traverse the crossbar bypassing SA. When the flit goes to the crossbar without SA, we assume that VA is performed independently from SA and ST. If SA is speculative, VA information is not needed until the flit is arriving at the downstream router.



(a) Pseudo-Circuit Creation



(b) Pseudo-Circuit Reuse



(c) Pseudo-Circuit Termination by Conflict

Figure 11. Pseudo-Circuit Creation, Reuse, and Termination

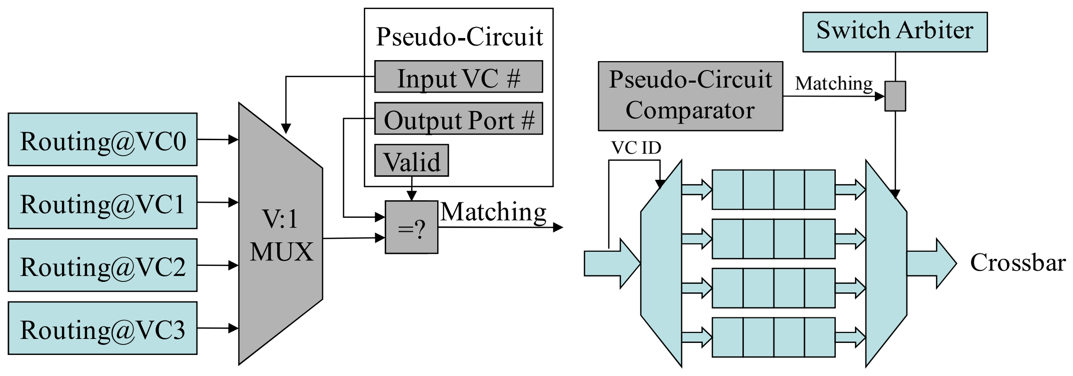
2. Pseudo-Circuit Termination

There are two conditions for pseudo-circuit termination; (1) a conflict with another recent pseudo-circuit and (2) congestion at the downstream router on the output port. If either the input port or the output port is assigned to another pseudo-circuit, the previous pseudo-circuit must be terminated because one input or output port cannot have more than one pseudo-circuit. If, for instance, another flit at a different input port claims the same output port as shown in Figure 11 (c), a new pseudo-circuit is created, and the previous pseudo-circuit is terminated and disconnected while clearing the valid bit in the pseudo-circuit registry without changing the registers. After termination, no flit is accepted for the terminated pseudo-circuit without SA because there is no pseudo-circuit. Thus, router latency is improved only when there is a proper pseudo-circuit connected from an input port to an output port in the router. Since pseudo-circuit traversal is made only when no other flit in SA claims any part of the pseudo-circuit, this scheme is free from starvation.

A pseudo-circuit is also terminated when the downstream router connected to the output port is congested. In order that the pseudo-circuit existence guarantees credit availability of the output port, the router needs to check the credit of the output port when performing SA. If congestion is detected, the pseudo-circuit at the output port with no credit should be immediately terminated to avoid buffer overflow. After the pseudo-circuit is terminated, no flit can go directly to the crossbar without SA. Since the switch arbiter performs arbitration based on the credit availability in the downstream routers, flits cannot traverse to the congested output port until the congestion is relieved. In this case, they need to stay in the buffer at the input VC. If the router also experiences congestion because of the congestion at the output port, flits coming from the upstream routers occupy buffers without traversing. This congestion produces Head-of-Line blocking and the following flits are also stored in the buffer. Once the buffers in the input port are full, credit back-pressure results in pseudo-circuit termination in the upstream router. If network congestion is maintained for a long time, it is finally propagated to the source node.

Pseudo-circuit termination does not need any performance overhead. Pseudo-circuits are managed by the switch arbiter in each router and each pseudo-circuit is connected by a crossbar switch within a crossbar. Terminating the pseudo-circuit simply disconnects the crossbar switch while clearing the valid bit at the pseudo-circuit register in the input port. If the switch arbiter needs to connect another crossbar connection using either the input port or the output port of the pseudo-circuit, it turns on the crossbar switch while terminating the previous pseudo-circuit. Thus, there is no performance overhead when a pseudo-circuit is terminated.

3. Pseudo-Circuit Architecture



(a) Pseudo-Circuit Comparator (b) Control Path

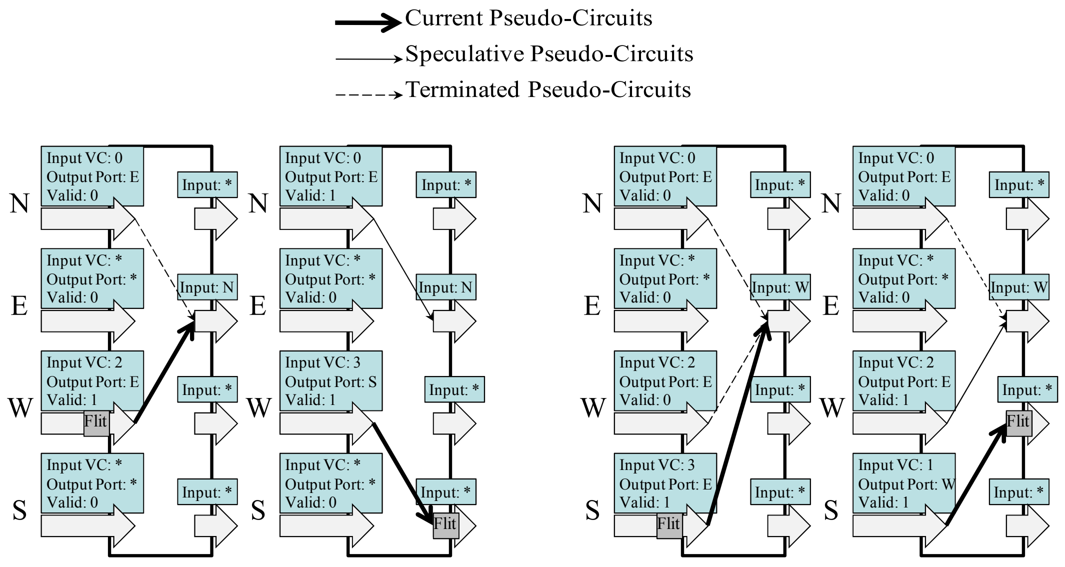
Figure 12. Pseudo-Circuit Architecture

Figure 12 (a) shows the pseudo-circuit comparator logic that determines whether the flit can use the pseudo-circuit. This logic contains two registers, a one-bit flag, one multiplexer, and one comparator; a register for the input VC of the pseudo-circuit, a register for the output port of the pseudo-circuit, a one-bit flag indicating whether or not the pseudo-circuit information stored in the registers is valid, a multiplexer to select one input VC, and a comparator to compare the routing information of the selected flit. Every input port has pseudo-circuit comparator logic because every input port can be connected with a pseudo-circuit. Since the area overhead of the logic is very small compared to the other router control logic, we assume the hardware overhead of the pseudo-circuit scheme is negligible.

This scheme needs additional delay to compare routing information with the pseudo-circuit before sending flits to the crossbar in ST. According to our HSPICE analysis at 45nm, the pseudo-circuit comparator takes 37ps, which can be fit into ST because ST takes only 215ps in the baseline microarchitecture and the pipeline period is 299ps which is determined by VA. SA has the same delay as it does in the baseline (290ps) because switch allocation in SA is performed independently from pseudo-circuits. Since pseudo-circuit information is updated in parallel with crossbar switch setup after switch allocation and termination is performed while switch allocation is done, no additional delay is required in SA. Therefore, this scheme has no additional overhead in delay analysis and does not affect the router clock frequency.

D. Aggressive Pseudo-Circuit Schemes

In this section, we present two aggressive pseudo-circuit schemes for further latency improvement. First, pseudo-circuit speculation creates more pseudo-circuits from currently unallocated crossbar connections for future flits using history information to increase pseudo-circuit reusability. Second, buffer bypassing allows flits to skip buffer writes at input VCs to reduce per-hop router delay. These two aggressive schemes can be combined with the basic pseudo-circuit scheme for further latency improvement.



(a) Speculative Pseudo-Circuit Creation (b) Conflict Resolution

Figure 13. Pseudo-Circuit Speculation

1. Pseudo-Circuit Speculation

Generally, not all of the possible crossbar connections are allocated to flits or pseudo-circuits. However, those unallocated crossbar connections may be claimed by near-future flits. If near-future flits traverse these crossbar connections as pseudo-circuits, the flits can bypass SA, thus reducing per-hop router delay. Speculatively creating more pseudo-circuits by connecting the unallocated crossbar connections improves communication latency because it increases pseudo-circuit reusability.

The prediction is performed based on the history information at each input port. If the output port most recently connected to an input port becomes available, pseudo-circuit speculation restores the pseudo-circuit when performing SA, connecting the input port and the output port again. There are two conditions for speculative pseudo-circuit restoration. First, if the output port becomes available, pseudo-circuit speculation revives the pseudo-circuit. Second, if there is congestion relief at the downstream router of the output port, the pseudo-circuit can be speculatively reestablished.

Figure 13 (a) shows an example where a speculative pseudo-circuit is created in the router. In this figure, the input port N is speculatively connected to the output port E because the previous pseudo-circuit from the input port N was recently connected to the output port E and this output port is now available and unallocated. To resolve conflict when more than one input port have the same history information to the same output port, pseudo-circuit speculation has a register to store the input port number of the last pseudo-circuit at each output port and connects the output port only to the input port the register indicates. This history register at each output port indicates the input port of the most recently terminated pseudo-circuit. For instance, the previous pseudo-circuit at the input port W is speculatively connected to the output port E instead of the input port N because the input port W is more recently connected to the output port E as shown in Figure 13 (b). To avoid buffer overflow in the downstream router, pseudo-circuit speculation does not create any pseudo-circuit to the output port of the congested downstream router which has no available credit.

Since every input port has a pseudo-circuit comparator, the pseudo-circuit scheme performs comparison of the pseudo-circuit with the routing information of an incoming flit. If the speculation is correct, the flit can traverse directly to the crossbar because the comparator generates a matching signal. If not, the comparator does not assert a match. In the mismatch case, flits go through SA resulting in no additional penalty after terminating the speculative pseudo-circuit. Since the termination of a speculative pseudo-circuit is performed in parallel with switch allocation, pseudo-circuit speculation has no negative performance impact.

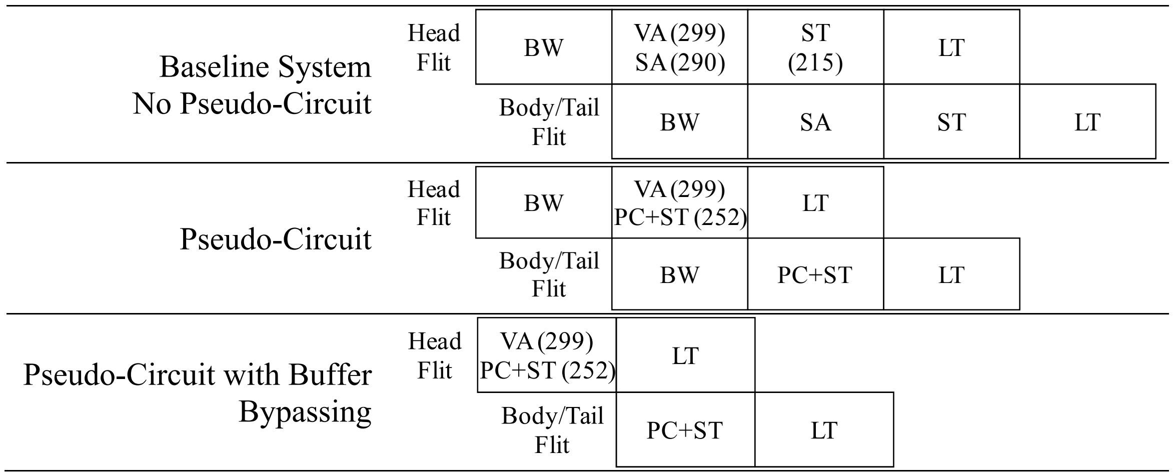


Figure 14. Pipeline Stages: Numbers are indicating critical path delay (ps).

2. Buffer Bypassing

We observe that flits traversing pseudo-circuits go directly to the crossbar at the next cycle after buffer write (BW) in most cases. Since the router anticipates flits after creating pseudo-circuits, there is no need to spend one cycle to write the flits to buffers at input VCs if the flits can traverse the pseudo-circuits. Instead, the flits can bypass buffers through the bypass latch at the input VC and go directly to the crossbar only if the pseudo-circuit is already available and connected from the input port to the designated output port. Only if a flit arriving at the input port has the same routing information to the same output port of the pseudo-circuit, it can bypass the input buffer at the input VC. In this case, the flit goes directly to the crossbar, saving 2 cycles in per-hop router delay as shown in Figure 14. Otherwise, the flit is stored in the input.

Buffer bypassing can be implemented with write-through input buffers [33] with a bypass latch per each input VC. When a pseudo-circuit is created, the bypass latch is turned on to enable incoming flits to bypass the buffer at the corresponding input VC unless the buffer is occupied. A flit can go to the bypass latch when the VC and the routing information of the flit are matched with the pseudo-circuit. If, for example, an incoming flit has the same routing information when the bypass latch is turned on, it goes directly to the crossbar through the bypass latch. If, however, the incoming flit has different routing information, the flit is stored in the buffer without bypassing buffer. Due to this conflict, the pseudo-circuit is immediately disconnected and invalidated while the bypass latch is turned off. To avoid buffer overflow, the pseudo-circuit is also terminated and the bypass latch is turned off if the output port is out of credit before a flit arrives. Thus, the pseudo-circuit can guarantee credit availability of the output port. Since buffer bypassing condition is controlled by a pseudo-circuit and its comparator logic, buffer bypassing has small hardware overhead.

When buffer bypassing is turned on, there is no need to store the whole flit. VA is performed only for header flits and it needs the output port numbers only. However, an incoming flit is always written to buffer just in case when the speculation is failed. If the speculation is correct, there is no pointer increment and the buffer will be overwritten by the next flit.

E. Sensitivity Analysis of the Pseudo-Circuit Scheme

In this section, we discuss the performance sensitivity of the pseudo-circuit scheme on various routing algorithms, such as deterministic, oblivious, and adaptive routing algorithms. We also discuss the sensitivity on VC allocation policies.

1. Routing Algorithms

There are several kinds of routing algorithms for inter-connection networks. First, deterministic routing algorithms select an output port deterministically like dimension order routing (DOR) algorithms, such as XY and YX. Once the destination is determined, they always route packets to the same communication path. Second, oblivious routing algorithms select a communication path randomly. Valiant's randomized algorithm [34] is one of the oblivious routing algorithms. Third, adaptive routing algorithms find an output port adaptively in every hop. If there are multiple output ports to a destination, they select one of them using the network information, such as queue occupancy [23]. Theoretically, adaptive routing algorithms have better performance than oblivious routing algorithms. Specifically, when the network traffic is concentrated to a certain hotspot, adaptive routing algorithms spread traffic to other possible directions, achieving load-balancing, while oblivious routing algorithms spread traffic randomly without knowing the network information. On the other hand, deterministic routing algorithms always select the same output port to a destination regardless of the network status or randomness. If a routing algorithm chooses an output port within the shortest path only, it is called a minimal routing algorithm. For example, PFNF (Positive First Negative First) [16] is a partially adaptive routing algorithm, fully adaptive in region of adaptiveness and oblivious in the restricted area [35], while O1TURN [11] is an oblivious and minimal routing algorithm because it chooses either XY or YX randomly in 2-dimensional meshes. Since they utilize all possible directions in 2-dimensional topologies, they achieve better throughput than any DOR algorithm because of load-balancing.

Basically, the pseudo-circuit scheme improves communication latency depending on the possibility that two consecutive packets traverse the same crossbar connection in a single router. If these packets are destined to the same destination, deterministic routing algorithms can make the packets reuse the pseudo-circuit in every hop because they always select the same output port. Thus, it results in better performance enhancement. However, oblivious and adaptive routing algorithms may have multiple communication paths to the same destination because they might select a different output port. If the next packet traverses the network in a different path to the same destination from the previous one, pseudo-circuits cannot be reusable. Thus, oblivious and adaptive routing algorithms may have less reusability and a smaller amount of latency improvement than deterministic routing algorithms when they are combined with the pseudo-circuit schemes. As shown in Figure 18 and Figure 19, PFNF has the lowest pseudo-circuit reusability and latency enhancement among all routing algorithms when working with the pseudo-circuit schemes.

To improve performance of the pseudo-circuit schemes with adaptive routing algorithms, routing decision can select the output port connected by the pseudo-circuit instead of the best candidate in adaptive routing algorithms. According to our simulation results, this approach can increase pseudo-circuit reusability if the current pseudo-circuit is one of the possible communication paths to the destination. For instance, when an adaptive routing algorithm selects the south output port, the best candidate for load-balancing, and the current pseudo-circuit is connected to the west output port, another candidate for routing, we can select the west port to reduce per-hop router delay in the current router. This choice can reduce the latency in the current router, but it causes more traffic concentration to the downstream router of the pseudo-circuit. Additionally, reusability improvement in this approach is quite marginal if the adaptive routing algorithm is minimal. Since there are a limited number of possible output ports to a destination in minimal routing algorithms, the possibility that the pseudo-circuit is a routing candidate is small. Therefore, we observe that this adaptive pseudo-circuit scheme results in no latency enhancement but a little latency degradation due to unbalanced traffic load, compared to the original pseudo-circuit scheme with adaptive routing algorithms.

2. Virtual Channel Allocation Policies

The pseudo-circuit scheme can completely bypass SA only if two conditions are both satisfied. The incoming flit must route to the same output port and be placed in the same input virtual channel as the previous flit. If one of them is not satisfied, the incoming flit cannot bypass SA. The first condition is dependent on the routing algorithm because the routing decision determines the output port. The second condition depends on the VA policy because the VA policy selects one VC in each input port.

Employing multiple virtual channels (VCs) increases network throughput because it reduces the possibility of head-of-line (HOL) blocking [10]. Virtual output queues [36] have no HOL blocking but this scheme requires many VCs when the radix of a router is high. To reduce this overhead, a dynamic VA policy is generally used with a smaller number of VCs. This policy assigns an output VC that has the least queue occupancy. Since a difference output VC can be selected depending on the network state, the dynamic VA policy has low pseudo-circuit reusability, while a static VA policy chooses the same VC per flow, resulting in better reusability than the dynamic VA policy. Therefore, the static VA policy always has better reusability and consequently better latency improvement than the dynamic VA policy. We have detailed sensitivity analysis results in Section G.2.

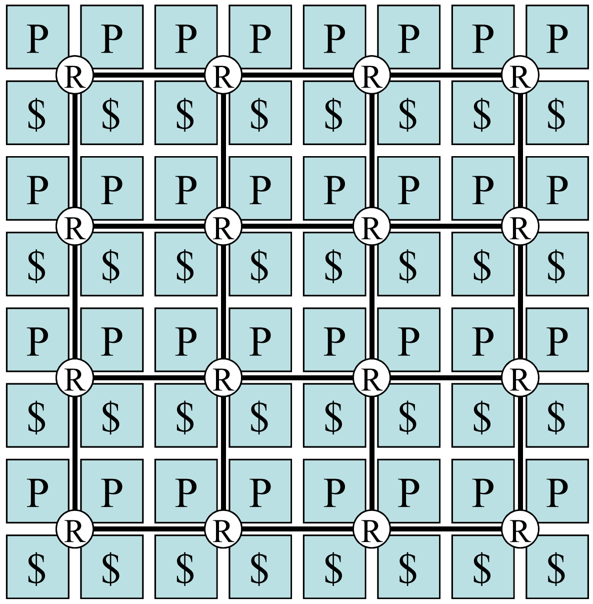


Figure 15. Layout of On-Chip Networks

F. Experimental Methodology

We evaluate communication latency and energy consumption using our cycle-accurate on-chip network simulator implementing pipelined routers with buffers, VCs, arbiters, and crossbars. The network simulator is configured with 4-flit buffer per each VC and 4 VCs per each input port. We assume that the bandwidth of a link is 128 bits with additional error correction bits. We use both traces and synthetic workloads for traffic models. We extract traces from several multi-threaded benchmarks; fma3d, equake, and mgrid from SPEComp2001 [37]; blackscholes, streamcluster, and swaptions from PARSEC [38]; NAS parallel benchmarks [39]; SPECjbb2000 [40]; FFT, LU, and radix from Splash-2 [41]. To extract the traces information, we use Simics [30], a full system simulator, configured as a SunFire multiprocessor system with UltraSPARCIII+ processors running with Solaris 9 operating system.

Table 1. CMP Configuration Parameters

L1I Cache	1-way 32KB	# Cores	32 out-of-order
L1D Cache	4-way 32KB	# L2 Banks	32 512KB bank
L1 Latency	1 cycle	Cache Block Size	64B
Unified L2 Cache	16-way 16MB	Memory Latency	300 cycles
L2 Bank Latency	20 cycles	Clock Frequency	5GHz

We develop a customized timing-model interface which has out-of-order cores with 4 MSHRs per each processing core to implement a self-throttling CMP network [29]. Our CMP configuration has 32 out-of-order processors and 32 L2 cache banks in a single chip, modeling a static non-uniform cache architecture (S-NUCA) [42]. Figure 15 shows the layout of the CMP configuration in this experiment. Processing cores and L2 cache banks are connected through the on-chip interconnection network. We use the concentrated mesh topology [12] for the on-chip interconnection network where each router connects 2 processing cores and 2 L2 cache banks to the interconnection network and routers are connected as a 2D mesh topology. Each core has 32KB L1 caches for data and instructions. L2 caches are unified and shared by all processing cores in an address-interleaved manner. We use CACTI model [43] to estimate the latency and the area constraint of the caches. Table 1 shows the detailed parameters used in the experiment.

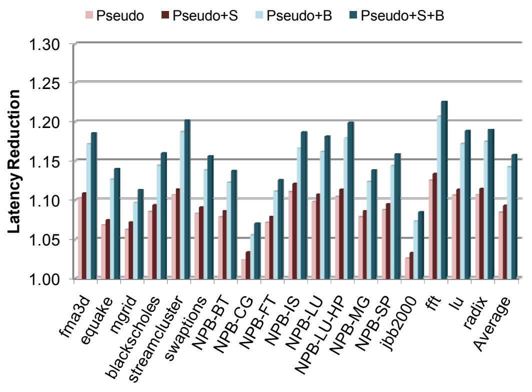
We use a directory-based MSI cache coherence protocol. If an L1 cache has a miss, it always generates a request to the corresponding L2 cache bank. After retrieving the data blocks, the L2 cache bank sends a response to the requesting L1 cache. To simplify the cache coherence states, we use write-through and write-invalidation. The cache coherence protocol has 3 different types of transactions. A read transaction is initiated by a read miss in L1 caches; a write transaction is initiated by a write miss in L1 caches; a coherence management transaction is initiated to keep shared copies coherent. The size of a network packet depends on whether it contains a data block. If the packet has an address only, it is 1 flit long because the size of an address is fit into a flit. If the packet has both an address and a data block, it is 5-flit long because the last 4 flits contain the data.

Table 2. Energy Consumption Characteristics of Router Components

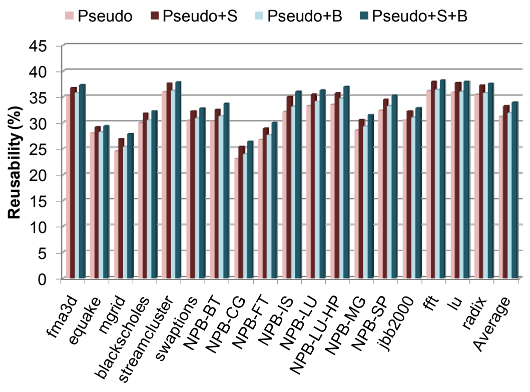
Buffer	Crossbar	Arbiter
20.19pJ	65.38pJ	0.20pJ
23.54%	76.22%	0.24%

We use Orion [44] to estimate router energy consumption. Table 2 shows characteristics of energy consumption and the percentage of energy consumed in each router component at 45nm. In this experiment, we assume that the amount of energy consumed in pseudo-circuit comparators can be negligible because it can be implemented with simple logics compared to the other router control logics.

In this experiment, we use two simple dimension order routing (DOR) [31] algorithms (XY, YX), O1TURN [11], and PFNF [16] in order to test the sensitivity of the pseudo-circuit scheme. O1TURN is the most recently proposed load-balanced oblivious routing algorithm in 2D meshes. To achieve uniformity, it randomly chooses the first dimension to traverse between XY and YX. PFNF is a recently proposed load-balanced adaptive routing algorithm working in 2D meshes. It partitions the network into two virtual networks like O1TURN and uses different turn models [15] in each virtual network, Positive-first and Negative-first, respectively. Since each virtual network is completely separated, PFNF is deadlock-free. Thus, it performs like a fully adaptive routing algorithm in region of adaptiveness and O1TURN in restricted area [35]. Due to fair comparison to DOR algorithms and O1TURN, we use minimal adaptive PFNF only in this experiment. We also use two VC allocation policies for sensitivity test. First, a dynamic VA policy chooses an output VC based on buffer availability in a downstream router. This allocation policy is generally used in interconnection networks. Second, a static VA policy chooses an output VC based on the destination of the communication path. If, for example, two different communications have the same destination ID, these are on the same VC at all input ports. If they share the same communication path from a certain point in the network, they use the same pseudo-circuit in each router in this shared communication path. This static VA policy is similar with [45] because one VC is statically allocated per flow, but we use the destination ID only in order to increase reusability.



(a) Latency Reduction



(b) Pseudo-Circuit Reusability

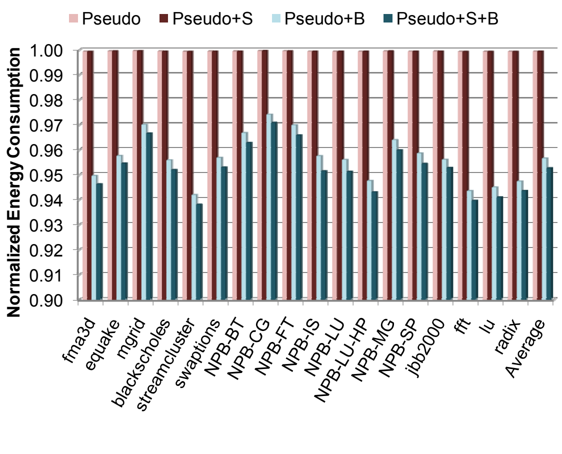
Figure 16. Overall Performance of the Pseudo-Circuit Scheme

G. Performance Evaluation

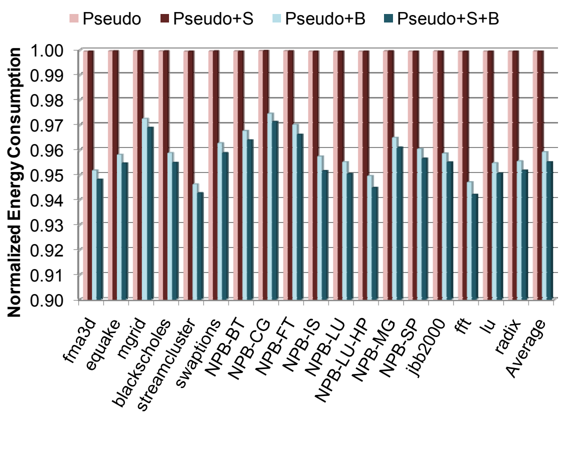
In this section, we evaluate the proposed pseudo-circuit scheme to examine how it affects communication latency and energy consumption in on-chip interconnection networks with traces from several benchmarks. We also evaluate its performance with synthetic workload traffic.

1. Performance and Energy Consumption with Traces

Figure 16 (a) shows overall communication latency of our proposed scheme when the application traces are used. We use network latency reduction normalized to latency of the baseline system without any pseudo-circuit scheme. To make a fair comparison, we choose the baseline system with O1TURN and the dynamic VA policy, which is the optimal in performance and hardware overhead. Throughout this chapter, the pseudo-circuit scheme without any aggressive scheme is indicated by **Pseudo**. The pseudo-circuit scheme with pseudo-circuit speculation is indicated by **Pseudo+S**. The pseudo-circuit scheme with buffer bypassing is indicated by **Pseudo+B**. The pseudo-circuit scheme with both aggressive schemes is indicated by **Pseudo+S+B**. Pseudo-circuit speculation has small contribution in latency reduction due to limited prediction capability as shown in Figure 16 (a). On average, we achieve 16% of latency reduction when the pseudo-circuit scheme with both aggressive schemes is applied.



(a) XY

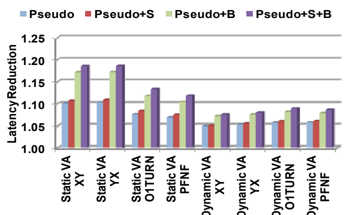


(b) YX

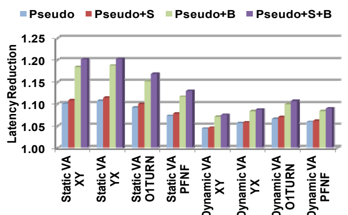
Figure 17. Overall Energy Consumption of the Pseudo-Circuit Scheme

Figure 16 (b) shows overall pseudo-circuit reusability in benchmark applications. Pseudo-circuit reusability, simply reusability, is defined as percentage of flits reusing pseudo-circuits. The higher reusability is, the more latency improvement is expected. Note that buffer bypassing does not actually increase reusability, but reduces one more cycle in per-hop router delay when the incoming flit bypasses buffer writing. If pseudo-circuit speculation and buffer bypassing are combined together, more latency enhancement is achieved because average per-hop router delay can be reduced more than when the two aggressive schemes are working separately.

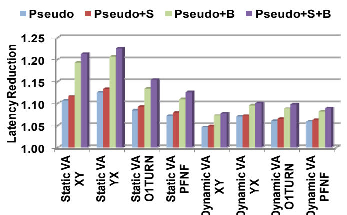
Figure 17 shows normalized energy consumption in routers. Since the amount of energy consumed in arbiters is much smaller than the amount of energy consumed in buffers, the pseudo-circuit schemes without buffer bypassing virtually have no energy saving. However, buffer bypassing reduces energy consumption because the energy consumed in buffer read and write is quite large. When combined with both buffer bypassing and pseudo-circuit speculation, the pseudo-circuit scheme has more energy saving due to more buffer bypassing. Thus, it achieves about 5% energy saving on average.



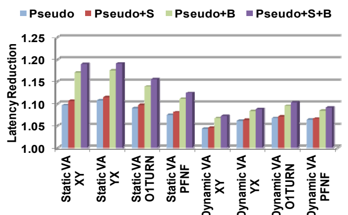
(a) fma3d



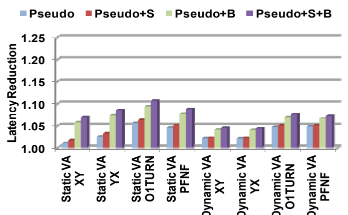
(b) streamcluster



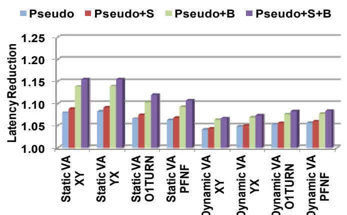
(c) fft



(d) radix

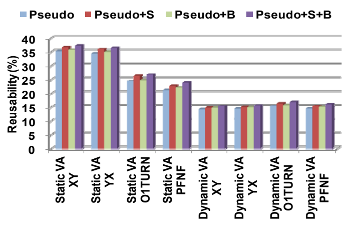


(e) jbb2000

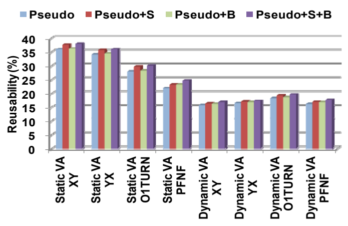


(f) Average

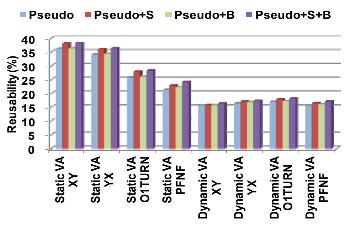
Figure 18. Network Latency Reduction



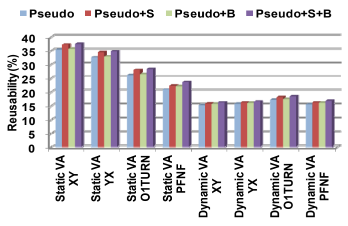
(a) fma3d



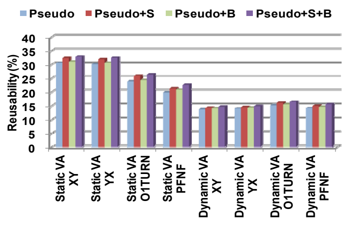
(b) streamcluster



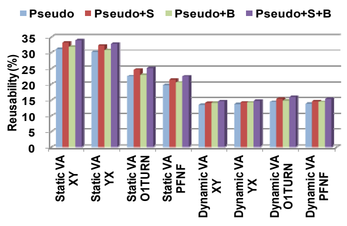
(c) fft



(d) radix



(e) jbb2000



(f) Average

Figure 19. Pseudo-Circuit Reusability

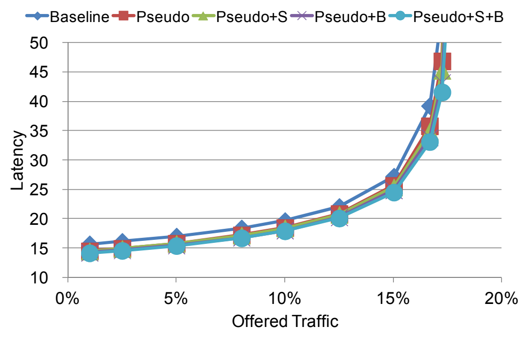
2. Performance on Sensitivity Test

We test the sensitivity of the pseudo-circuit scheme to various routing algorithms and VC allocation policies. In this experiment, we use two dimension order routing (DOR) algorithms (**XY** and **YX**), a recently proposed load-balanced oblivious routing algorithm (**O1TURN**), and a load-balanced adaptive routing algorithm (**PFNF**). To perform the sensitivity to VC allocation policies, we apply two different policies (**Static VA**, **Dynamic VA**).

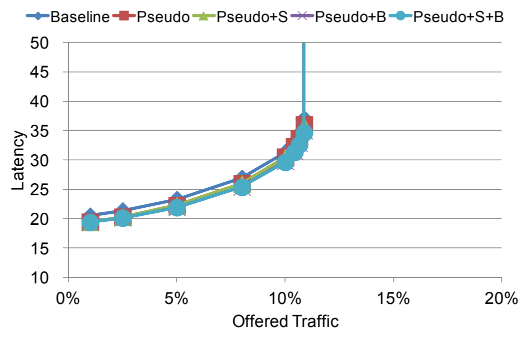
Figure 18 and Figure 19 show the sensitivity of the pseudo-circuit scheme in latency reduction and pseudo-circuit reusability, respectively. Among all possible combinations, DOR with the static VA policy has the best latency reduction in all pseudo-circuit schemes although there is a subtle difference in communication latency between XY and YX. This difference between two DOR algorithms comes from asymmetry in application traces. Generally, higher pseudo-circuit reusability results in higher latency reduction. As shown in Figure 19, DOR with the static VA policy maximizes pseudo-circuit reusability compared to the other combinations because it always chooses the same output port and the same VC for flows to the same destination. We observe that routing algorithms and VA policies have higher impact on reusability than application locality does. Leveraged by this, our pseudo-circuit scheme with DOR and the static VA policy has the best latency improvement in most multi-threaded benchmarks in general. Note that YX with the static VA policy has traffic concentration, causing slightly better pseudo-circuit reusability but less latency reduction than XY with the static VA policy, due to contention.

Since O1TURN and PFNF may have multiple paths from a single source to a single destination, they have less pseudo-circuit reusability and consequently less latency improvement than deterministic routing algorithms as shown in Figure 18 and Figure 19. Likewise, the dynamic VA policy has less reusability than the static VA policy because it might choose a different VC for a single flow. However, these graphs show that the oblivious and adaptive routing algorithms with the dynamic VA policy still have a certain amount of communication locality. Thus, we conclude that communication behavior still affects the communication locality regardless of the routing algorithm and the VA policy used in the network.

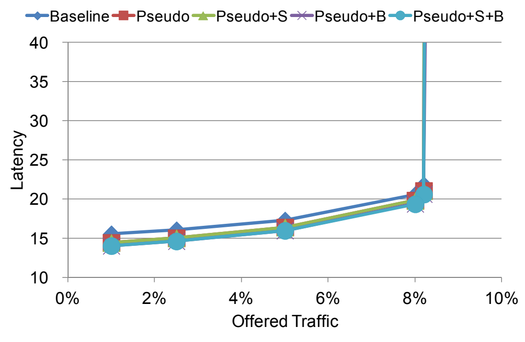
Our proposed pseudo-circuit scheme generally has the best latency improvement with DOR and the static VA policy in most benchmarks. If, however, an application like jbb2000 has highly skewed and over-utilized network hotspots, DOR cannot relieve the hotspot traffic. Instead, O1TURN and PFNF can achieve better latency with the pseudo-circuit scheme than DOR because they can distribute traffic into every dimension and utilize every possible link to achieve load balancing. As shown in Figure 18 (e), jbb2000 has better latency with other than DOR unlike the other benchmarks due to uneven traffic caused by hotspots. However, we observe that the static VA policy has better performance than the dynamic VA policy because of better pseudo-circuit reusability.



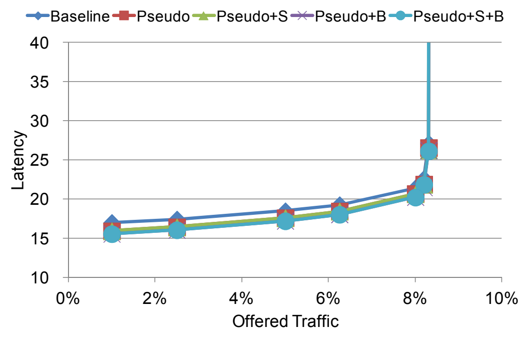
(a) UR



(b) BC



(c) BP



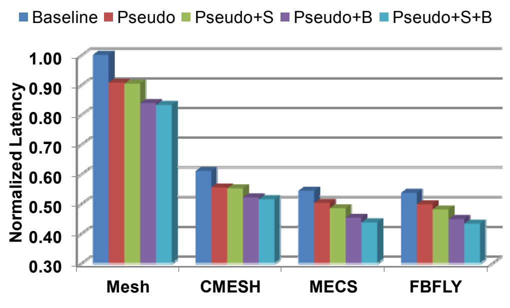
(d) Tornado

Figure 20. Performance Comparison with Synthetic Workload

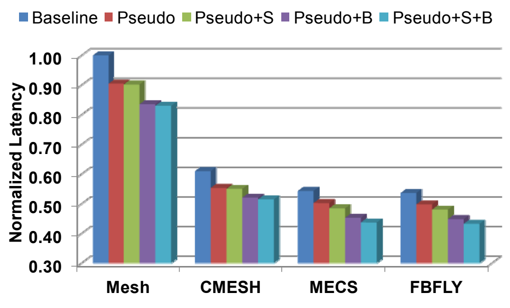
3. Performance Evaluation with Synthetic Workload Traffic

We also conduct experiments with several different kinds of synthetic workload traffic. First, we generate uniform random (**UR**) traffic, which randomly sends packets evenly to all nodes. Thus, it has equal chances to use all links. In this traffic, the destination of each communication path may differ from the previous one because it is randomly selected every time. The second traffic is bit complement (**BC**). It generates traffic from every source to one single destination based on bit complement operation. Thus, it has a longer average Manhattan distance than UR. This longer distance results in faster saturation in the network. The third traffic is bit permutation (**BP**), whose communication pattern is generated based on matrix transpose. This traffic has only one destination for each source like BC, but the average Manhattan distance is same as UR. Every communication in this traffic needs to cross the diagonal line. Since it traverses the same point with the dimension order routing algorithms, this traffic saturates the network much earlier than BC. Lastly, we generate tornado (**Tornado**) traffic where all communication traffic is going clockwise and the destination nodes are always 4-hop away from source nodes. In the experiments with synthetic traffic, all packets are 5 flits long.

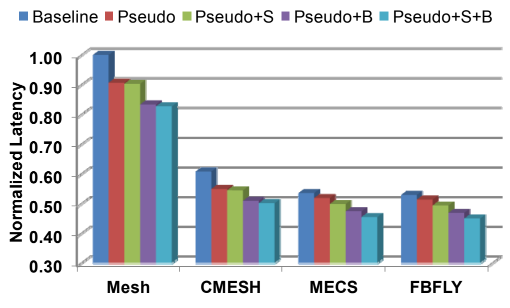
Figure 20 shows latency improvement in synthetic workload traffic. We show only the results of XY with the static VA policy because the results of YX are exactly same in synthetic workload traffic. At any traffic load before saturation, the pseudo-circuit scheme performs better than the baseline system with all synthetic workload. In low-load traffic, UR and BP have nearly 11% latency improvement while BC and tornado have only around 6% and 9% latency improvement, respectively. It is expected that UR has less communication locality because the next packet can be sent to a different destination from a previous packet. However, these two consecutive communications may have a common path in dimension order routing algorithms. Since the pseudo-circuit scheme exploits pseudo-circuit reusability within a single router, it can improve communication latency within the common path. This figure also shows that there is nearly no improvement in network throughput because pseudo-circuits are frequently terminated in high-load traffic due to contention.



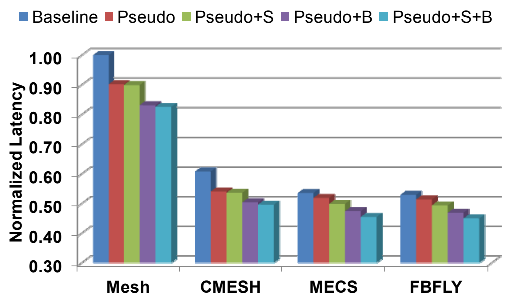
(a) fma3d - XY



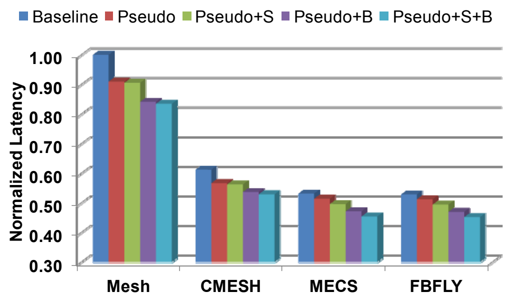
(b) fma3d - YX



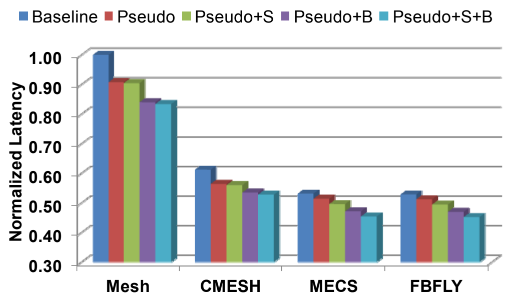
(c) fft - XY



(d) fft - YX



(e) blackscholes - XY



(f) blackscholes - YX

Figure 21. Performance Improvement on Various Topologies

H. Discussion

In this section, we evaluate the pseudo-circuit scheme in various topologies, such as Multidrop Express Cube [32] and Flattened Butterfly [14], to show how it is affected by topologies. We also compare the pseudo-circuit scheme with other on-chip network techniques, such as Express Virtual Channels (EVC) [25], Hybrid Circuit Switch [28], and *circuit reusing* in multiple independent parallel networks [28].

1. Impact on Various Topologies

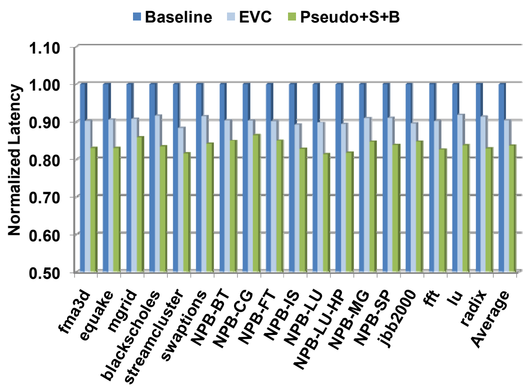
Figure 21 shows communication latencies of several topologies, normalized to the baseline system in a mesh topology. Since DOR with the static VA policy has the best performance improvement among other combinations, we show the results of this combination. This figure shows the results of some benchmarks, but we observe that the other benchmarks have the same trend. To show latency improvement of the pseudo-circuit scheme in various topologies, we use a mesh, a concentrated mesh (**CMESH**) [12], Multidrop Express Cube (**MECS**) [32], and Flattened Butterfly (**FBFLY**) [14]. In this experiment, we assume all physical channels have the same bandwidth and each input port has 4 VCs. MECS is configured without any replicated channel, thus resulting in less crossbar complexity than FBFLY.

The communication latency is calculated as $T = H_{avg} * t_{router} + D * t_{link} + T_{serialization}$ where H_{avg} is the average number of hops, t_{router} is per-hop router delay, D is average distance from source to destination, and t_{link} is unit length delay. Since link latency and serialization latency are constants, the total communication latency is determined by the product of per-hop router delay and the average number of hops. The pseudo-circuit scheme reduces per-hop router delay regardless of the underlying topology while recently proposed topologies reduce the number of hops. Simulation results show the pseudo-circuit scheme achieves up to 20% latency improvement in any topology. Therefore, combination with recent topologies results in more than 50% latency reduction compared to the baseline system with a mesh topology.

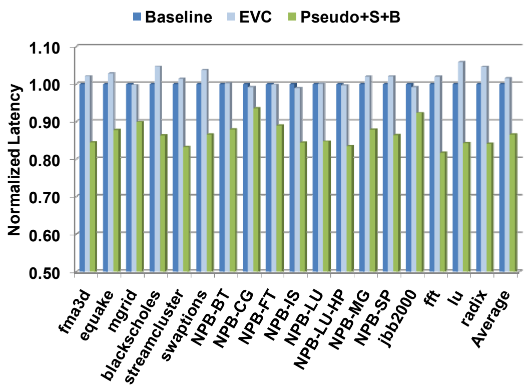
2. Comparison with Express Virtual Channels

Figure 22 shows performance comparison with EVC [25]. We use 4 VCs per each input port with a 4-flit buffer per each VC in both techniques. We use dynamic EVC with the maximum number of hops $l_{max} = 2$ where 2 VCs are reserved for express VCs (EVCs) and the other 2 VCs are used for normal VCs (NVCs). In each graph in this figure, latency is normalized to the baseline system in each topology.

EVC reserves some VCs to send packets to routers in multiple hops away in the same dimension. Thus, it improves communication latency by prioritizing these EVCs over the other NVCs because it provides express channels virtually. If, however, a network topology has a small number of routers in a single dimension, our experiment shows that EVCs are not sufficiently utilized and this unbalanced usage may cause performance degradation due to the reduced number of NVCs. For example, the concentrated mesh topology does not have latency improvement on average with EVC as shown in Figure 22 (b) because most flits are stored in NVCs, and these NVCs are easily filled with flits. EVCs cannot be much utilized because the number of routers in a single dimension is small. Thus, EVC is heavily dependent on the topology used in on-chip interconnection networks.



(a) Mesh



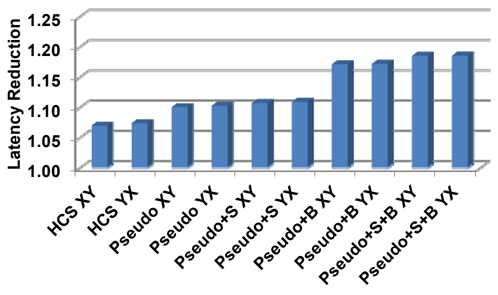
(b) Concentrated Mesh

Figure 22. Comparison with Express Virtual Channels (EVC)

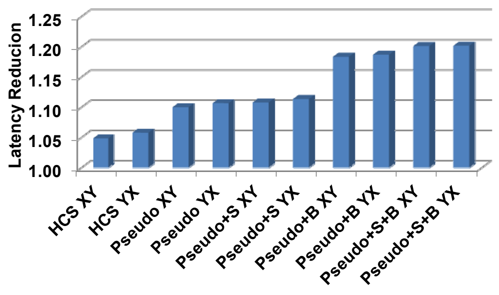
The pseudo-circuit scheme has several advantages. First, there is no topological restriction when creating pseudo-circuits. Pseudo-circuits can be established from any input port to any output port. Thus, it is topologically independent as shown in the previous section. Second, the pseudo-circuit scheme does not reserve any resource when creating pseudo-circuits. If there is a pseudo-circuit in the input port and the next flit is destined to another output port, the pseudo-circuit is terminated to enable the flit to traverse through the crossbar to the routed output port. Besides, there is no performance overhead to terminate the pseudo-circuit. Finally, there is no starvation. Pseudo-circuits are simply disconnected and terminated immediately to avoid starvation when there is a conflict with other flits in SA.

3. Comparison with Hybrid Circuit Switching

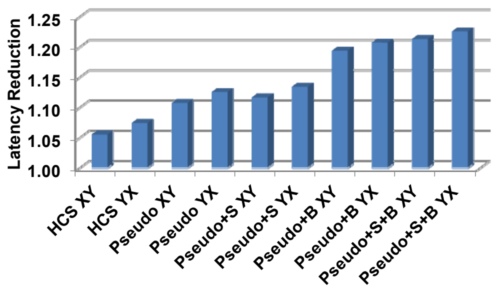
Recently proposed Hybrid Circuit Switch (HCS) [28] reduces communication latency by reusing circuits, communication paths from source to destination, in order to overcome the long setup delay in circuit switching networks. Specifically, hybrid circuit switching does not terminate a circuit after communication and reserves it for future uses. Due to communication temporal locality, it is expected that the reserved circuit can be reused in the future. However, HCS assumes only a whole circuit from a source to a destination can be shared. If there is no more network resource for a new circuit, it terminates a part of previous circuits and creates a new circuit. This conflict with other circuits causes frequent partial circuit termination. Thus, this technique cannot fully utilize the communication temporal locality because of partial circuit termination.



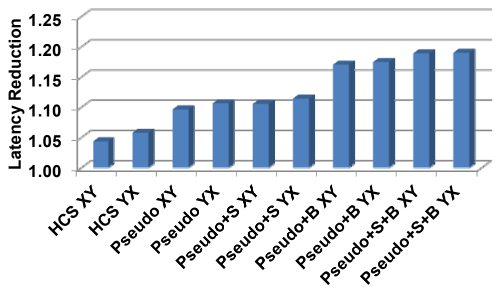
(a) fma3d



(b) swaptions



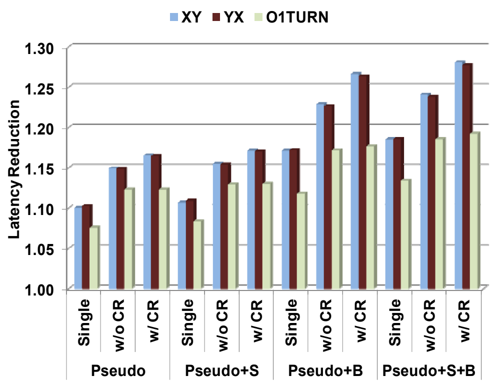
(c) fft



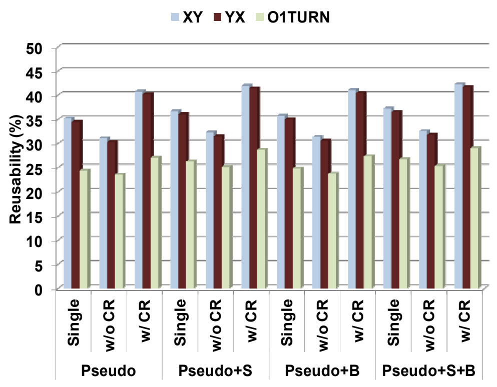
(d) radix

Figure 23. Comparison with Hybrid Circuit Switching (HCS)

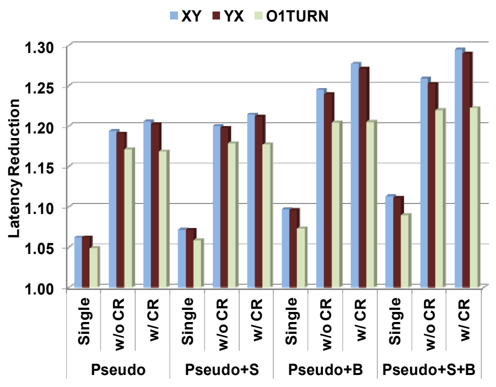
Figure 23 shows latency reduction of HCS and the pseudo-circuit scheme, normalized to the baseline system. It shows the results of some benchmark applications, but we observe the same trend in all benchmarks. On-chip interconnection networks are configured with only one data network in both schemes. Thus, HCS suffers from frequent partial circuit termination. Due to limited reusability, HCS achieves less latency improvement than the pseudo-circuit scheme. However, the pseudo-circuit scheme maximizes reusability with less termination. Therefore, the pseudo-circuit scheme improves network latency 13% more than HCS on average in all benchmarks.



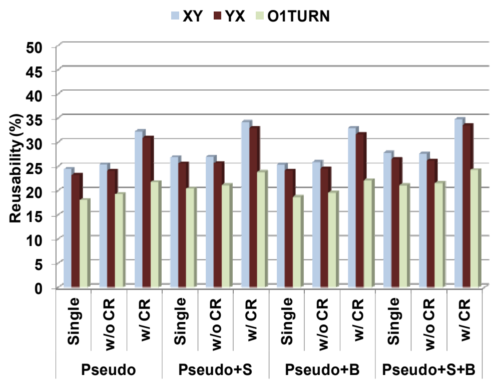
(a) Latency Reduction on fma3d



(b) Pseudo-Circuit Reusability on fma3d



(c) Latency Reduction on mgrid



(d) Pseudo-Circuit Reusability on mgrid

Figure 24. Performance Enhancement with Circuit Reusing (CR)

4. Performance Enhancement with Circuit Reusing

When multiple independent bandwidth-divided parallel networks [12] are deployed in on-chip interconnection networks, each communication node has one injection/ejection port per parallel network to provide connectivity to all parallel networks. When a packet is injected, an injection port needs to be assigned to select a parallel network. If the same parallel network previously used for the same destination is assigned again, it increases possibility to reuse pseudo-circuits. This parallel network assignment, called *circuit reusing* [28], was originally proposed to reuse circuits in Hybrid Circuit Switching. Since this history-based circuit reusing increases pseudo-circuit reusability, we can expect more communication latency improvement. To minimize the overhead of history retrieval, we store only one history of the most recent destination per each injection port at each source node.

Generally, circuit reusing improves communication latency in all pseudo-circuit schemes. Since it always chooses the same parallel network recently used for the same destination, it has higher pseudo-circuit reusability than random selection. For instance, circuit reusing in 2 independent parallel networks increases pseudo-circuit reusability by nearly 10% as shown in Figure 24. Thus, circuit reusing has 4% more latency improvement in fma3d than without circuit reusing. We present only the results of fma3d with the static VA policy, but we observe the similar trend in other benchmarks.

I. Conclusions

CMPs have a performance bottleneck in on-chip interconnect networks due to communication latency. To overcome the bottleneck, it is crucial to design a low-latency on-chip network. We introduce a pseudo-circuit scheme to reduce per-hop router delay by reusing pseudo-circuits, crossbar connections within a router with previous arbitration information. This scheme enables flits to bypass SA when they are traversing through the same communication path created by previous communication. For further latency improvement, we also propose two more aggressive schemes; pseudo-circuit speculation and buffer bypassing. Pseudo-circuit speculation generates more pseudo-circuits using currently unallocated crossbar connections for future communication while buffer bypassing allows flits to skip buffer writes at input VCs and removes one more pipeline stage from per-hop router delay. Combined with both aggressive schemes, the pseudo-circuit scheme enhances overall performance of on-chip interconnection networks by 16% with traces from SPEComp2001, PARSEC, NAS Parallel Benchmarks, SPECjbb2000, and Splash-2. It also improves about 5% of energy consumption in routers. Evaluated with synthetic workload traffic, this scheme shows latency improvement by up to 11%. If applied to the recently proposed topologies, it improves communication latency by up to 20%, resulting in more than 50% latency reduction, compared to the baseline with a mesh topology.

CHAPTER IV

A CASE FOR HANDSHAKE IN NANOPHOTONIC INTERCONNECTS

A. Introduction

With the prevalence of dual-core and quad-core processors, a many-core era with thousands of cores in a single die has been expected. Providing efficient communication in a single die is becoming a critical factor for high performance Chip Multi-Processors (CMPs) [46]. Network-On-Chip (NoC) is a promising architecture that orchestrates chip-wide communications in the many-core era. As the on-chip network size continues to increase, the bandwidth required to support concurrent computations on all cores increases by the order of magnitude. Evidence suggests that many-core systems using electrical interconnects may not be able to meet scalability and high bandwidth while maintaining acceptable performance within power and area budgets [47]. Hence, architects have explored alternative technologies including electrical transmission lines [48], radio frequency (RF) signaling [49], and nanophotonics [50, 51, 52]. While electrical transmission lines and RF suffer from low bandwidth density and relatively large components, nanophotonics provides high bandwidth density, low latency, and distance-independent power consumption, which makes it a promising candidate for future NoC designs.

Optical interconnects have been developed to provide better performance with low power consumption. Kirman et al. [50] propose to use optical components to build on-chip buses. Some studies [53, 54] directly migrate the topologies widely used in electrical networks to optical interconnects, which are overlaid over an electrical network with the same topology, and the optical network uses circuit-switching by sending set-up packets in the electrical network. Corona [52] and Firefly [51] propose ring-based networks, which win popularity by getting rid of the overhead of a secondary electrical network and few or no waveguide crossing even in a large scale network.

Since on-chip channels and buffers are limited resources, arbitration and flow control become critical factors in the NOC design. In electrical on-chip networks with the hop-by-hop transmission manner, packets need to compete for the buffer resource in the middle although their destinations are different. Credit-based flow control fits into the electrical NoC design because the short and fixed transmission delay between neighboring nodes makes the flow control information easier to be synchronized. However, in ring-based optical interconnects, all the on-chip traffic logically becomes one-hop communication so no intermediate buffer allocations are required. Only senders with the same receiver compete for the buffer resource at the receiver side. Meanwhile, the diverse delays from different senders to the same receiver make the flow control information hard to be synchronized. Credit-based flow control becomes inefficient in ring-based optical interconnects.

In this work, we propose two handshake schemes for nanophotonic interconnects, Global Handshake (GHS) and Distributed Handshake (DHS). Instead of using traditional credit-based flow control, the proposed handshake schemes rely on acknowledgements between senders and receivers. A sender begins to transmit packets right after winning the channel arbitration without knowing the buffer status at the receiver side. A receiver sends back *ACK* or *NACK* messages as a feedback. Packet dropping and retransmission may occur if the buffer is full. While the basic handshake schemes suffer from the Head-Of-Line (HOL) blocking problem, we overcome this with setaside buffer and circulation techniques that improve the channel utilization further. Our evaluation shows that the proposed handshake schemes improve network throughput by up to 11× under synthetic workloads with the packet dropping and retransmission rates below 1%. With the extracted trace traffic from real applications, the handshake schemes can reduce the communication latency by up to 55%. The handshake schemes add only 0.4% hardware overhead for optical components and negligible power consumption. In addition, the performance of the handshake schemes is independent of on-chip buffer space, which makes them feasible in a large scale nanophotonic interconnect design.

The rest of this chapter is organized as follows, In Section B, we provide background on silicon nanophotonic technology and present a motivating case study to highlight the inefficiency of existing optical arbitration and flow control schemes. We present optical handshake schemes in Section C. Section D describes the architecture of a handshake optical network. In Section E, we describe the evaluation methodology and summarize the simulation results. Then, we briefly summarize the related work in Section F. Finally, we draw conclusions in Section G.

B. Motivation

In this section, we first present an overview of optical interconnects, including communication components, interconnect patterns, arbitration and flow control. Then, we discuss the inefficiency of existing optical arbitration and flow control schemes, which is the main issue we attempt to solve in this chapter.

1. Optical Communication Components

Optical communication structures consist of a laser source (normally located off-chip), waveguides carrying light, and micro-rings or silicon ring resonators that modulate and detect optical signals. The light from the laser source travels in a single directional through the waveguides with negligible losses. Multiple wavelengths can use the same waveguide with no interference. With dense-wavelength-division-multiplexing (DWDM), up to 128 wavelengths can be generated and carried by the waveguides [55]. Micro-rings are tuned to a particular wavelength and can be used to modulate or detect light of the particular wavelength when placed next to a waveguide. Meanwhile, rings can switch the light from one waveguide to another. The modulation, detection and diversion are controlled by an electrical signal, which tunes the ring between resonance “on” and “off” states. Functioning ring resonators are described in [56] and Figure 25 shows a conceptual optical link.

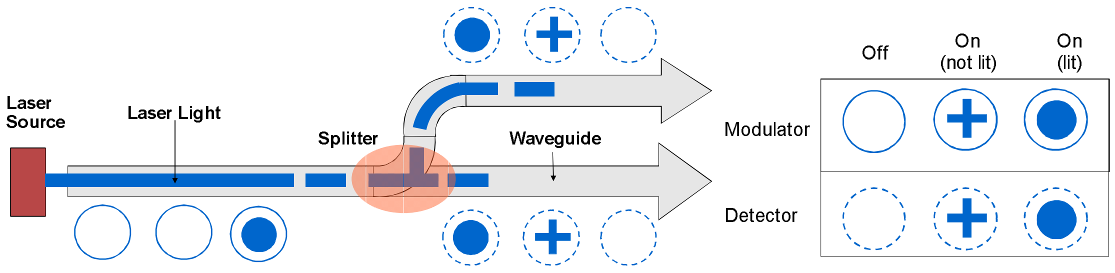
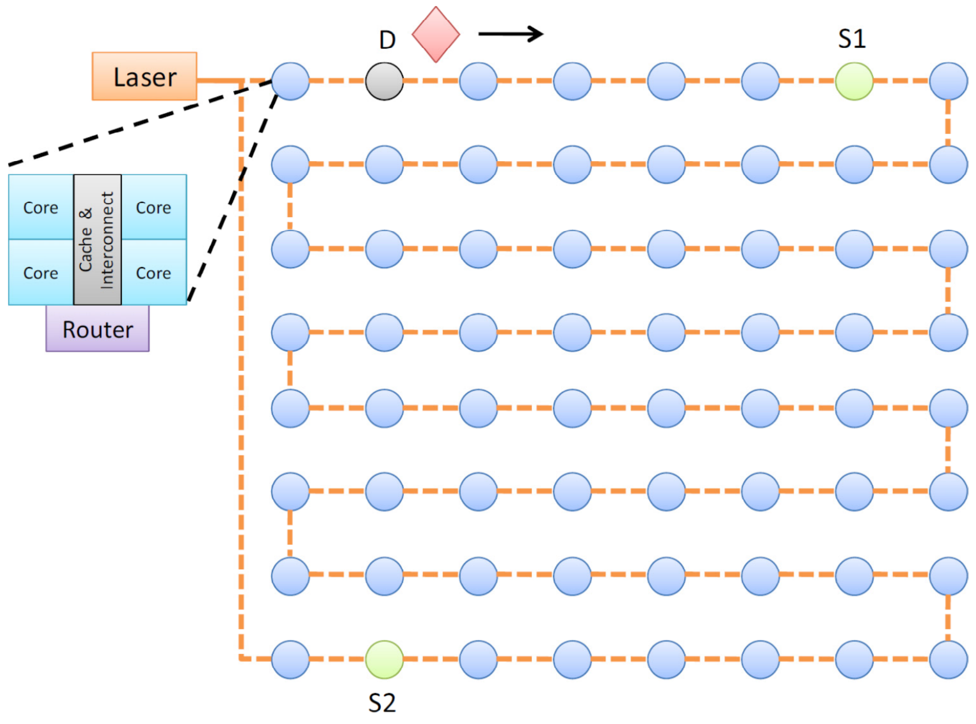


Figure 25. A Conceptual Optical Link

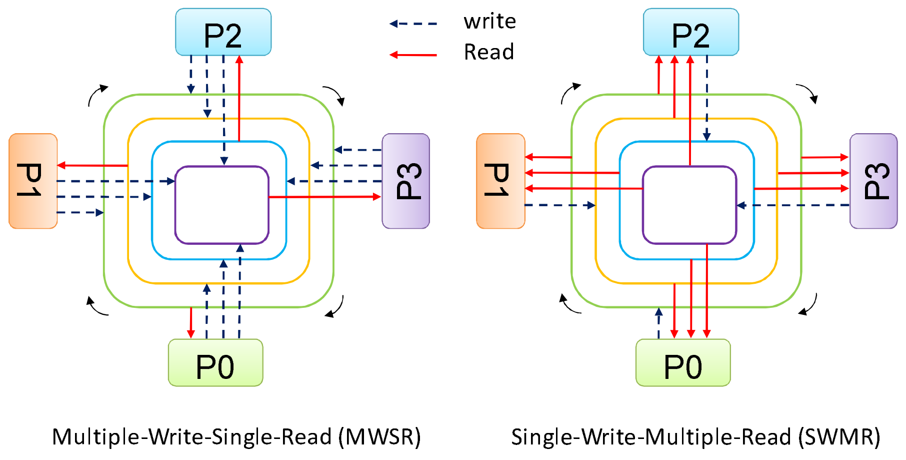
Ring detection is destructive, which means that an active ring detector removes all the light during the process of detection. Thus, any downstream detectors will not be able to detect the light. In other words, an active detector detects a light signal only when no upstream detector is activated. A ring splitter is used for switching a fraction of light to another waveguide without affecting modulated light signals.

2. Interconnect Patterns

In traditional electrical interconnects, each node is connected to its neighboring nodes using separate electrical links, such as a 2D Mesh network, while in optical interconnects nodes are normally attached to a single communication media forming a ring-based network as shown in Figure 2 (a). 64 nodes, each of which contains 4 cores, are connected through unidirectional optical rings. The ring-based optical interconnect falls into two categories: Multiple Write Single Read (MWSR), such as Corona [52], or Single Write Multiple Read (SWMR), such as Firefly [51]. Figure 26 (b) shows these two interconnects. In MWSR, a node can write to all the channels except one specific channel from which the node can read, while in SWMR a node can write to a specific channel from which any other nodes can read. MWSR needs arbitration in the sender side, since a destination node can only receive one light signal at a time. SWMR benefits from not requiring any arbitration in the sender, but introduces extra communication complexity. Considering multiple nodes can read from one given channel in SWMR, a reader should activate its detector. Since ring detection is destructive, we cannot allow all the nodes to keep their detectors activated all the time. Only the destination node is allowed to open its detector. To handle this situation, before sending data signals, the sender must notify the receiver of the future communication to activate the receiver's detector, which costs extra bandwidth and needs relatively expensive broadcast waveguides. Although our handshake schemes can be applied to both MWSR and SWMR, we choose MWSR as our interconnect pattern for its simplicity and low cost.



(a) Ring-Based network Architecture



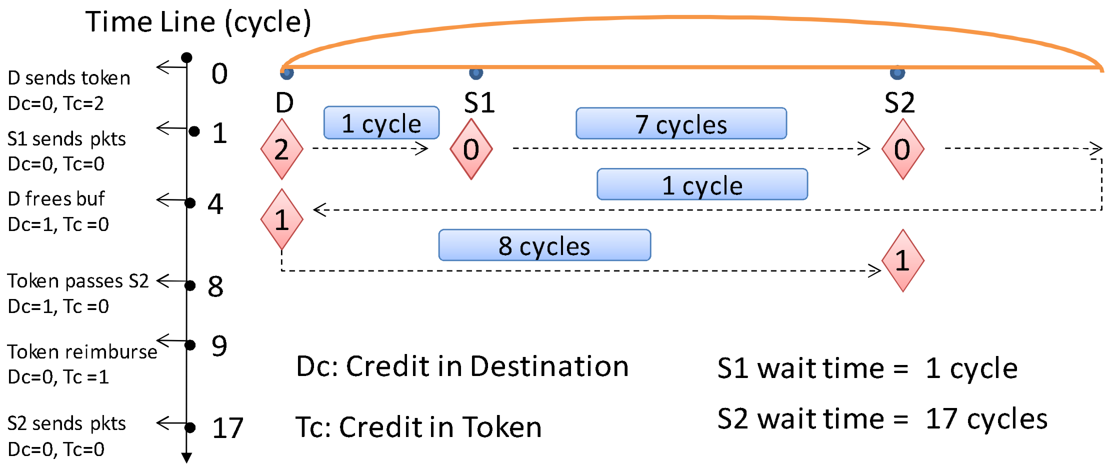
write
Read
P0
P1
P2
P3
Multiple-Write-Single-Read (MWSR)
Single-Write-Multiple-Read (SWMR)

(b) MWSR and SWMR

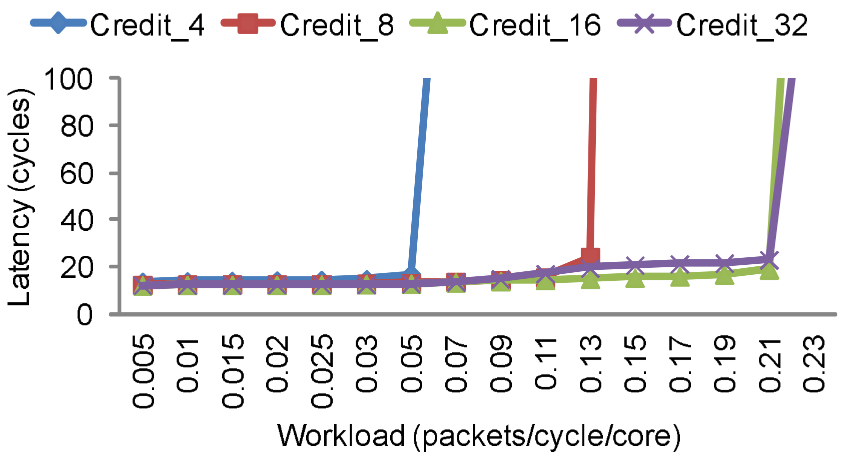
Figure 26. Optical Interconnect Patterns

3. Arbitration and Flow Control

With limited on-chip channel and buffer resources, arbitration and flow control become the most critical factors in the NOC design. In nanophotonic interconnects, packets traverse through optical channels in a wave-pipelined manner, which allows a single optical channel to be divided into several segments, and each segment is similar to a single-cycle bus. For example, on a 576 mm^2 chip with 64 nodes and a 5GHz clock, the round trip time for an optical channel is 8 cycles [52], so it can be divided into 8 segments. Considering the specific characteristics of optical channels, the arbitration of a shared optical channel can take two methods: global arbitration or distributed arbitration. Global arbitration is like a bus-based interconnect. In the whole round trip time, only one sender and one receiver will use the channel. Distributed arbitration considers the wave-pipelined manner of packet transmission. If two packets are not overlapped in the same segment at the same time, they can traverse in the same optical channel. Prior work [57, 58] adopts token-based arbitration, in which a photonic token represents the right of transmitting packets on a channel. Token channel is proposed for global arbitration, while token slot and token stream are designed as distributed arbitration. Traditional electrical on-chip interconnects hire credit-based flow control, in which upstream routers keep a record of the number of free buffers in downstream routers. When a router forwards a flit to the next hop, it sends a credit backward to its upstream router. Inherited from credit-based flow control, all the above token-based arbitration schemes integrate the credit information into the arbitration token.



(a) Coupled Arbitration and Flow Control

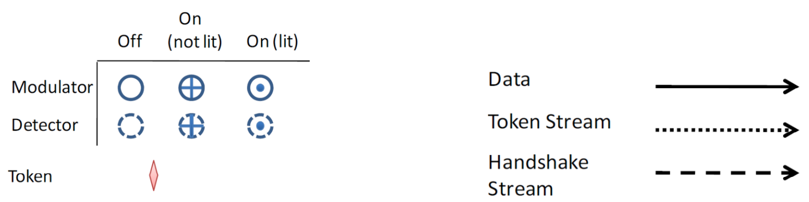


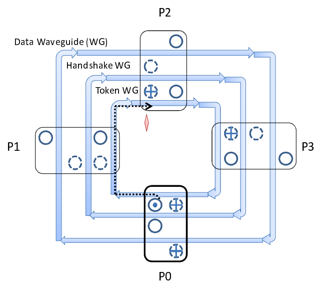
(b) Performance of Token Slot with a Different Number of Credits in Uniform Random

Figure 27. Arbitration and Flow Control in Token-Ring Network Architecture

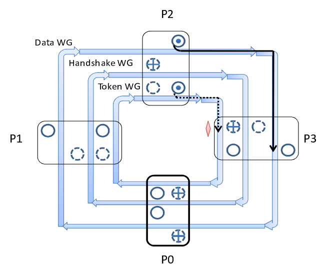
Traditional credit-based flow control benefits from the short and fixed transmission delay (normally one cycle) between neighboring nodes. However, in optical interconnects, the transmission latency between neighboring nodes is not always one cycle, which delays the synchronization of the credit information between the sender and the receiver. Figure 27 (a) shows such a situation. We assume the round trip time for the ring is 8 cycles. Nodes S_1, S_2 and D are connected in a ring, as shown in Figure 26 (a). Nodes S_1 and S_2 want to send packets to Node D. Before sending a packet, S_1 and S_2 need to get a token from Node D, which also carries the credit information of Node D, indicated by T_c in Figure 27 (a). In cycle 0, Node D sends out the token, and its local credit (shown as D_c) becomes zero. In cycle 1, the token arrives at Node S1, which consumes all the credits. When Node S_1 releases the token, there are no credits left in the token, which means Node S_2 cannot send a packet when the token arrives at Node S_2. Node S_2 should wait until the token returns to Node D and gets reimbursed. As shown in Figure 27 (a), Node D has newly freed buffer space (D_c becomes 1) in cycle 4. However, the token cannot get this information immediately since it is in the middle of transmission. Finally, it takes 17 cycles before Node S_2 has a chance to send a packet.

Token slot and token stream try to solve the above problem by adopting multiple tokens. Instead of piggybacking all the credits in a single token, token slot and token stream represents one credit with one token. The number of tokens depends on the number of credits at destination nodes. Destination nodes stop generating tokens if no more credits are available, making the network performance rely on the size of on-chip buffer space as shown in Figure 27 (b). We observe that a certain amount of on-chip buffers should be provided to avoid performance degradation. Therefore, credit-based flow control coupled with token-based arbitration is inefficient in the ring-based nanophotonic interconnect design.

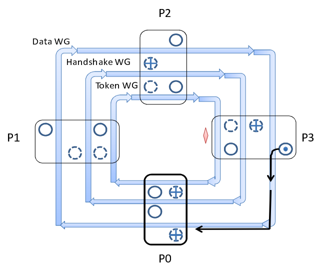




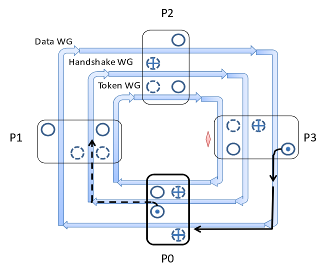
(a) In Cycle 0 and 1, a token passes Node P_1 with no request and arrives at Node P_2.



(b) In Cycle 2, Node P_2 sends a packet and releases the token.



(c) In Cycle 3, Node P_3 sends a packet following the packet from Node P_2. The token stays in Node P_3.



(d) In Cycle 4, the packet from Node P_2 arrives at the home node with free buffer slots. An ACK message is sent to Node P_2.

Figure 28. A Global Handshake Example

C. Optical Handshake

In this section, we propose two handshake schemes, Global Handshake (GHS) and Distributed Handshake (DHS). Both GHS and DHS are built upon Token-Ring protocol, which comes from the 802.5 Token-Ring LAN standard [59], in which a node must wait for a "free" token to transmit data. GHS uses global arbitration, while DHS adopts distributed arbitration. In the NOC design, a phit is the unit of information that can be transferred across a physical channel in a single cycle. In general, the size of the basic flow control unit (flit) is equivalent to the phit size. A packet can consist of one or multiple flits. Given the high bandwidth density of nanophotonics, the channels are often wide enough so that a large data packet can fit in a single flit5. In this work, we assume each packet contains a single flit. Thus, interleaving flits in the handshake schemes is not a serious problem.

1. Global Handshake

With global arbitration, GHS has a single token relayed among different senders. Since there are multiple writers but only a single reader in MWSR, the reader or the destination node is responsible for sending out the arbitration token. We define the single reader or destination node as a home node. When a node detects and removes the token, it has exclusive access to the data channel and starts to send packets in the next cycle. If there are no more packets to be sent, the token will be released to the other nodes and finally return to the home node. It will take multiple cycles for a packet to arrive at the home node. Since senders have no information about the buffer status of the home node, after the packet is sent, it cannot be removed from the sender side. When the packet arrives, the home node checks its buffer status. If there is free buffer space, the packet is stored into the buffer and an *ACK* message is sent back to the source node. Otherwise the packet is dropped and a *NACK* message is sent. When the source node receives an *ACK* message, the packet is removed from its input buffer and the following packets are ready for transmission. If a *NACK* message is received, the packet is waiting for retransmission.

Figure 28 shows the operation of Global Handshake. In this example, Node P_0 is set as the home node, and the other nodes try to send packets to P_0. We assume it takes one cycle for the token to traverse between two neighboring nodes. In Cycle 0, Node P_0 sends out the arbitration token, which will keep circulating in the token channel. Since Node P_1 has no request, the token passes Node P_1 and arrives at Node P_2 in Cycle 1. In Cycle 2, Node P_2 begins to send a data packet. Because Node P_2 has no more packets to send, it releases the token. In Cycle 3, Node P_3 gets the token and sends its data packet, which follows the packet from Node P_2 in a wave-pipelined manner. The token stays in Node P_3, since it has more packets to send. In Cycle 4, the packet from Node P_2 arrives at the home node, which has free buffer slots. An *ACK* message is sent to Node P_2 through the handshake waveguide.

Global Handshake gets rid of the traditional credit-based flow control. Senders can send a packet without knowing the buffer status at the home node even though there could be no credits available at the home node in the current cycle. If the home node frees a buffer slot one cycle before the packet arrival, the packet can be successfully delivered. With limited buffer space, packet dropping and retransmission may occur. Based on our evaluation, packet dropping and retransmission rate is less than 1% even in high workloads. Decoupled with flow control, GHS shortens the average waiting time and therefore improves the network throughput. Figure 29 shows the same example as Figure 27 (a) with GHS, where the waiting time for Node S_2 is reduced from 17 cycles to 8 cycles. Global Handshake has only one token circulating around the channel. After releasing the token, it takes a whole round trip time for a node to get the token again, even though other nodes have no packets to send. This situation becomes worse in a large network, in which the token round trip time can be tens of cycles. To solve this problem, multiple tokens should be provided, which introduces Distributed Handshake (DHS).

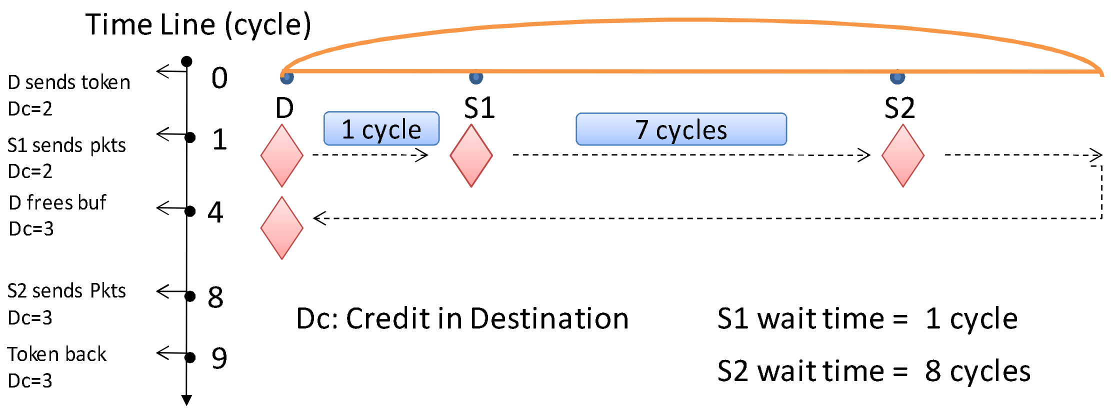
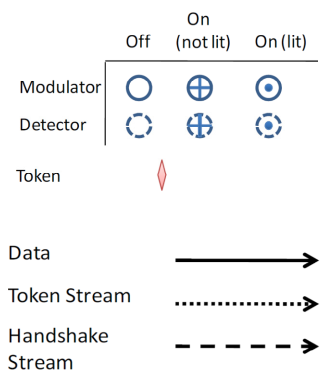


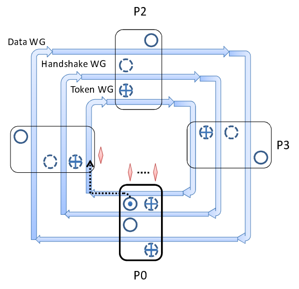
Figure 29. Global Handshake in a Token-Ring Network

2. Distributed Handshake

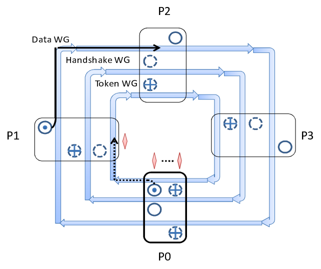
DHS considers the wave-pipelined manner of packet transmission in optical links. Home nodes keep generating a token every cycle. Multiple tokens divide the channel into fixed-size, back-to-back slots. In a cycle, only a portion of the network nodes are able to detect the token. If the token is taken by a node, there is no releasing operation for the token and other nodes cannot detect it forever. A sender can only send one flit after getting a token. Like GHS, packets cannot be removed from the sender side until an *ACK* message is received. Figure 30 shows the operation of DHS. Home Node P_0 keeps generating a token every cycle. In Cycle 0, a token arrives at Node P_1, which removes the token. In Cycle 1, Node P_1 starts to send a packet, and turns on the detector in Handshake Channel. Meanwhile, a new token from the home node is generated and arrives at Node P_1 again. However, since there is no new request from Node P_1, the token will keep traversing to Node P_2. In Cycle 2, the data packet from Node P_1 passes Node P_2, and the token arrives at Node P_2. Node P_2 takes the token, and starts its transmission in the next cycle. In Cycle 3, Node P_2 sends a data packet which follows the previous data packet from Node P_1, which arrives at the home node. After checking the buffer status, the home node, P_0, sends a handshake message to Node P_1 in Cycle 4.

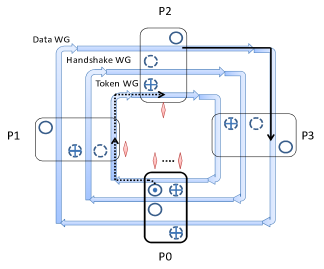
GHS and DHS allow senders to send packets without knowing the buffer status of destination nodes, decouple the optical channel arbitration with flow control, and consequently reduce the credit traversal time ideally to zero. However, basic GHS and DHS cannot avoid the Head-Of-Line (HOL) blocking problem. Note that Virtual Output Queue (VOQ) [36], which divides packets targeting for different destinations into separate queues, is an optional design for the buffers connected to shared optical channels. Before receiving an *ACK* message, senders cannot drop the packet that was sent, which makes the packet stay in the head of the input queue for at least a round trip time. The waiting packet will block the following packets in the same input queue. To avoid the HOL problem, we use a setaside buffer technique for GHS and DHS. Setaside buffers are small number of buffer slots, which are collocated with input queues. When a packet is sent out and waiting for the handshake message, it is temporally removed from the input queue and stored into the setaside buffer. Therefore, the next packet is moved to the head of the queue and is ready for transmission. The size of setaside buffers may affect the network performance, which is discussed in Section E.





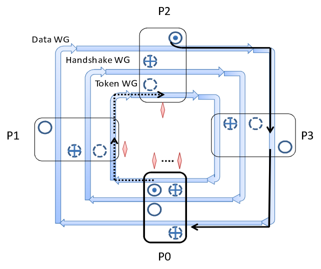
(a) In Cycle 0, a token is generated and traversing to Node P_1.

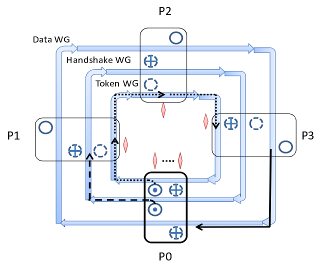




(b) In Cycle 1, Node P_1 sends a packet and turns on the detector in Handshake Channel.

(c) In Cycle 2, Node P_2 gets a token.

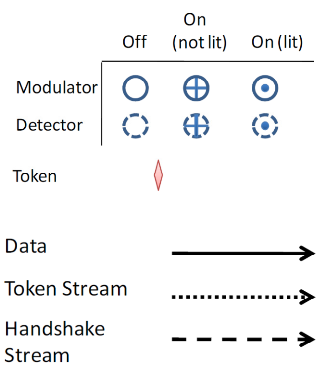


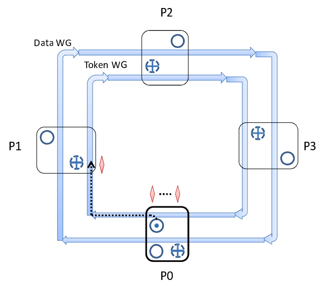


(d) In Cycle 3, Node P_2 sends a packet following the previous packet, which arrives at the home node.

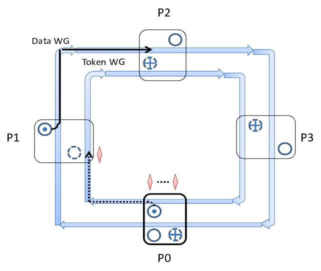
(e) In Cycle 4, a handshake message in sent to Node P_1.

Figure 30. A Distributed Handshake Example

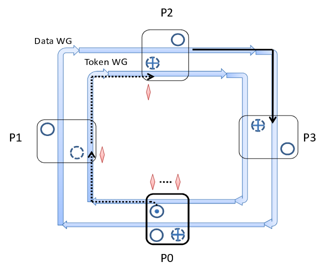




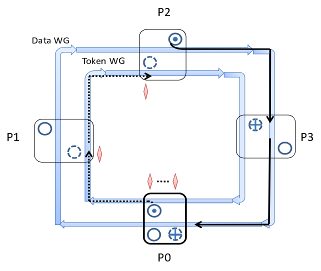
(a) In Cycle 0, a token is generated and traversing to Node P_1.

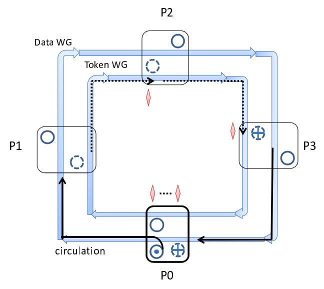


(b) In Cycle 1, Node P_1 sends a packet and removes the packet from its input buffer.



(c) In Cycle 2, Node P_2 gets a token.





(d) In Cycle 3, Node P_2 sends a packet following the previous packet, which arrives at the home node, which has no free buffer slots at that time.

(e) In Cycle 4, the home node reinjects the packet into the data channel. Meanwhile, the home node stop generating a token for that cycle.

Figure 31. Distributed Handshake with Circulation

3. Distributed Handshake with Circulation

While the setaside buffer technique tackles the HOL problem with additional buffer space, we propose another technique called circulation to remove this extra buffer overhead. The basic idea of circulation is that instead of packet dropping, receivers reinject packets into the same data channels if they do not have enough buffer space. The reinjected packet will circulate in the optical ring until the buffer is available at the receiver, which enables the sender to remove the packet from the head of queue immediately after sending it out. Therefore, we can avoid the HOL blocking problem. Since no packets are waiting for retransmission, senders do not need to send acknowledgments and we can also remove the handshake waveguide.

In basic DHS, each home node generates an arbitration token every cycle. When the home node needs to reinject packets into the data channel, any token will not be generated in the same cycle to avoid channel collision. The home node virtually consumes a token, and gets the permission to use the channel. Figure 31 describes the operation of DHS with the circulation technique.

Unlike DHS, the circulation technique cannot be applied to GHS. Note that GHS generates only one channel arbitration token, which is relayed among senders. Before the token returns to the home node, the home node cannot grant itself the permission of using the channel and thus no packets are allowed to be reinjected from the home node.

4. Fairness

One major problem of token-related protocol is fairness. Considering that a home node acts as a global controller to generate tokens for every sender, nodes close to the home node have higher priority over farther downstream nodes in obtaining tokens. Basic GHS and DHS partially solve the fairness issue because of the HOL blocking problem. Without receiving a handshake message from a home node, senders cannot remove the packets from the buffers. In other words, the following packet cannot request new tokens, potentially yielding a newly generated token to downstream nodes. However, with the setaside buffer and the circulation techniques, nodes close to home nodes can starve the farther downstream nodes. A similar problem has been addressed in [57], which proposes Fair Token Channel and Fair Slot with well served nodes sitting on their hands for a while and yielding the chance to other nodes. In this work, we adopt the same methods proposed in [57] to provide fairness for GHS and DHS.

D. Optical Handshake Architecture

In this section, we present the architecture of an optical network with the proposed handshake schemes.

1. Network Architecture

Figure 32 shows the architecture of an optical network with the handshake schemes. Each router is attached to global optical rings, which are composed of different channels, including data channels, token channels and handshake channels. A channel can consist of multiple waveguides, each of which carries 64 wavelengths. To support handshake schemes, extra components are added to the conventional virtual channel (VC) router, which are labeled as Output and Input modules. In the Output module, an output queue, designed as VOQ, is used to buffer the packets before Electronic/Optical (E/O) conversion. To avoid the HOL blocking problem, setaside buffers are added in parallel with the output buffer. Each setaside buffer slot is only one flit long, and connected to an output MUX. A handshake receiver processes ACK or NACK messages, and selects a flit to enter E/O conversion. In the Input module, the detector checks the status of the global optical ring. If any flit arrives, after Optical/Electronic (O/E) conversion, the flit will be stored into the router input buffer. In basic GHS and DHS schemes, if there are no empty slots in the input buffer, flits will be dropped. However, with the circulation technique, router buffer status is recorded in the circulation controller, which controls packet reinjection.

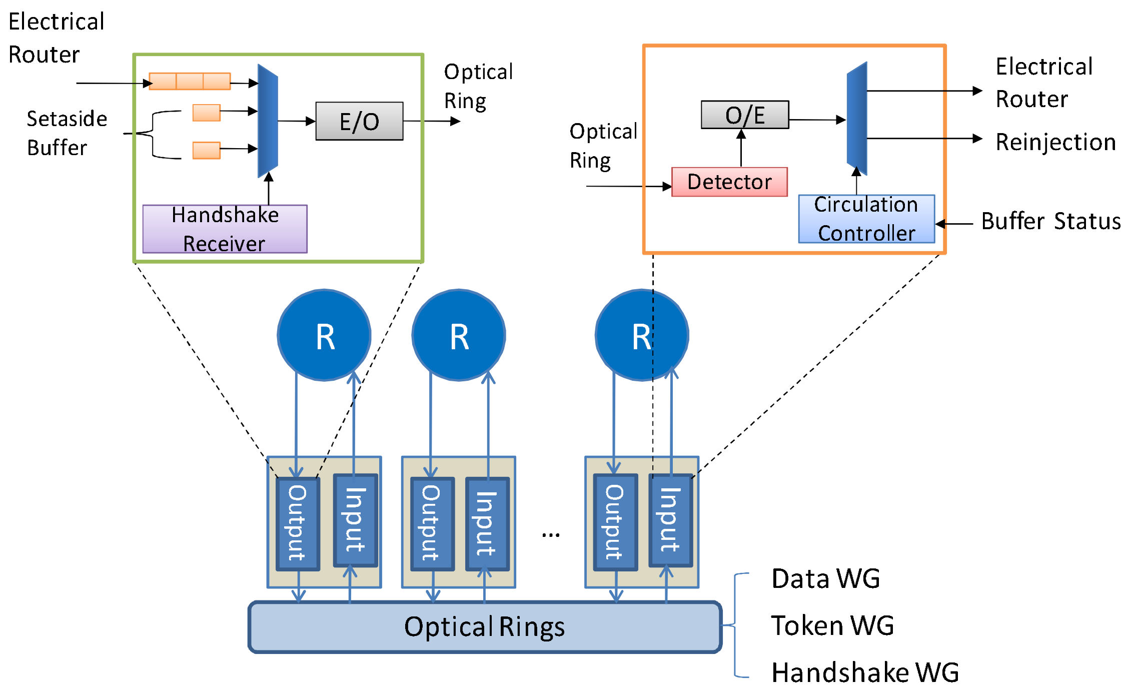


Figure 32. The Optical Network Architecture with the Handshake Schemes

2. Router Pipeline

A conventional electrical router processes packets with four pipeline stages, which are routing computation (RC), VC allocation (VA), switch allocation (SA), and switch traversal (ST). In optical on-chip networks, every router is attached to the global ring making any two routers become neighboring routers, which increases the overhead of recording the VC status for every neighboring router. Note that optical links can provide a wide link width, which is advisable for a single-flit packet design. There is no concern about flit interleaving in a network with only single-flit packets. Therefore, the VA stage can be removed from the traditional router pipeline, simplifying the electrical router logic. In this work, we adopt a two-stage electrical router, with RC and SA in one stage and ST in the other.

Table 3. Component Budgets for the Handshake Schemes in a 64-node Network

Optical Schemes	**Data WG**	**Token WG**	**Handshake WG**	**Micro-rings**
Token Slot [57]	256	1	0	1024K
GHS	256	1	1	1028K
DHS	256	1	1	1028K
DHS with Circulation	256	1	0	1024K

3. Hardware Overhead

The handshake schemes add handshake messages (*ACK* and *NACK*) into normal optical communication, which incurs extra hardware overhead. We analyze the hardware overhead in a network with 256 cores connected as 64 nodes. We advocate using a single bit for a handshake message. Note that in a segment of the channel only one node can get the arbitration token every cycle, and the round trip time for an optical ring is fixed. After sending a packet, the source node will receive a handshake message in a fixed amount of time. For example, if we assume the round trip time for the optical ring is 8 cycles, then a source node will receive the handshake message in 9 cycles. A source node only needs to turn on its handshake detector 9 cycles after sending a packet, while at other times it keeps the detector off and passes the handshake messages for other source nodes. That's why using a single bit, which indicate whether it is an *ACK* or a *NACK*, for handshake message is feasible. If we use one wavelength, modulated as 1 bit, for the handshake message of a node, 64 wavelengths are required in a 64-node network. Note that an optical waveguide can carry 64 wavelengths. Thus, only one waveguide is added to support the handshake schemes in a 64-node network. Since each wavelength requires 64 micro-rings to function as modulators or detectors, this extra waveguide needs total 4K micro-rings. Table 3 lists the budget of optical components for each handshake scheme. It indicates that the handshake schemes introduce only 0.4% overhead for both waveguides and micro-rings.

Table 4. Simulation Configuration

# Cores	128 out-of-order	Concentration	4
L1I Cache	1-way 32KB	Router Pipeline Stage	2
L1D Cache	4-way 32KB	Optical Link Latency	1 – 8 cycles
# L2 Banks	128 512KB/Bank	Data Channel Width/Flit Size	256 bits
Cache Block Size	64B	Clock Frequency	5GHz

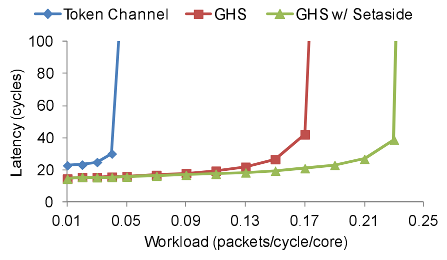
E. Experimental Evaluation

In this section, we first describe our evaluation methodology. Then, the performance of the proposed handshake schemes is analyzed, followed by comparison with previous designs. Based on the power model in [60, 61], we estimate the power consumption in the handshake schemes. Finally, we explore the schemes' sensitivity to a variety of network design points.

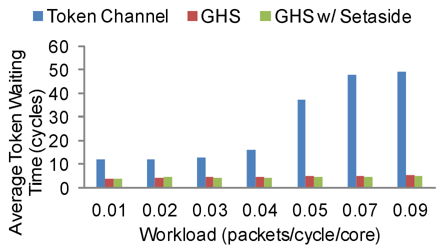
1. Methodology

Our evaluation methodology contains two parts. First, we use Simics [30], a full system simulator configured as a SunFire multiprocessor system with UltraSPARCIII+ processors running Solaris 9 operating system, to extract trace information from real applications. 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 [29]. The CMP system contains 128 out-of-order processing cores and 128 L2 cache banks in a single chip, connected as 64 nodes with 4-way concentration, modeling static non-uniform cache architecture (S-NUCA) [42]. Next, we evaluate performance and power consumption using a cycle-accurate on-chip network simulator that models a 2-stage pipelined router architecture. The total latency of E/O or O/E conversion is around 75ps [62] and is modeled as part of the nanophotonic link traversal time. Assuming a die size of 400mm^2 with a 5GHz clock, the nanophotonic link traversal time amounts to be 1 to 8 cycles based on the distance between the sender and the receiver. The workloads for our evaluation consist of synthetic workloads and traces from real applications. Three different synthetic traffic patterns, Uniform Random (**UR**), Bit

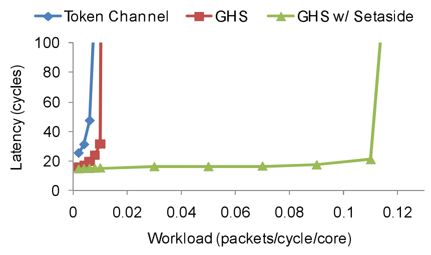
Complement (**BC**) and Tornado (**TOR**), are used. The real applications considered in this work are fma3d, equake, and mgrid from SPEComp2001 [37]; blackscholes, freqmine, streamcluster, and swaptions from PARSEC [38]; FFT, LU, and radix from SPLASH-2 [41]; NAS parallel benchmarks [39] and SPECjbb2000 [40]. Table 4 shows the simulation configuration.



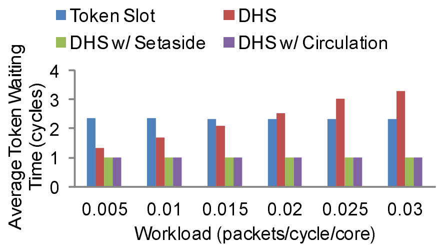
(a) UR



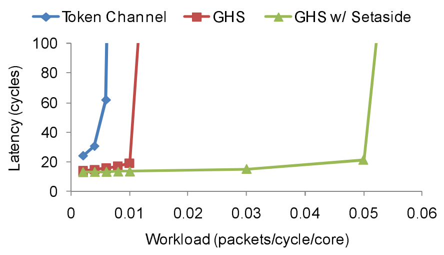
(d) UR



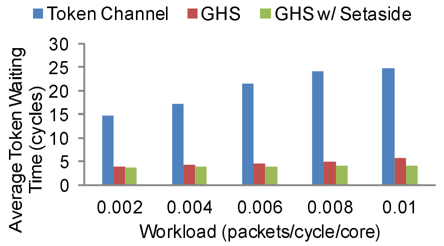
(b) BC



(e) BC



(c) TOR



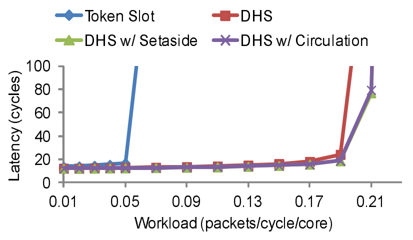
(f) TOR

Figure 33. Performance Evaluation of Global Handshake

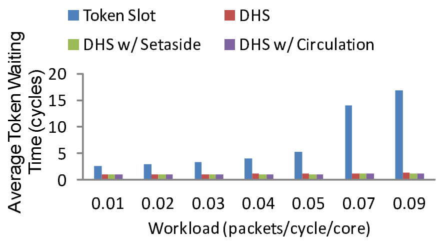
2. Performance

Given that GHS and token channel use global arbitration while DHS and token slot adopt distributed arbitration, we separate the performance evaluation into two groups. GHS related schemes are compared with token channel, and DHS related schemes are compared with token slot.

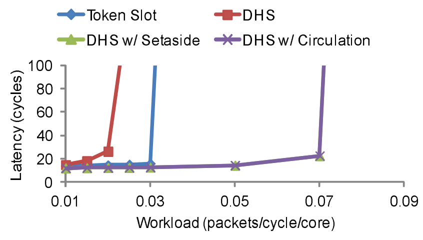
SyntheticWorkloads: We first evaluate average packet latency and saturation bandwidth with synthetic workloads. Figure 33 (a), (b) and (c) show the results of the schemes using global arbitration, in which only one arbitration token is circulating for each destination. The total amount of credits or buffer slots provided by each destination is four. The trend from the three traffic patterns is consistent. The handshake schemes achieve approximately 4-6× throughput improvement in UR and 5-11× in BC and TOR. Because token channel [57] suffers from the long token waiting time, especially after senders consume all the credits stored in the token, GHS produces better performance than token channel. In Figure 33 (d), (e) and (f), we evaluate the average token waiting time of different schemes. Since the three traffic patterns have different saturation points, we select different evaluation injection rates for the three traffic patterns in our experiments. Compared with token channel, GHS reduces the average token waiting time dramatically. With multiple arbitration tokens, distributed arbitration shortens the token waiting time. Figure 34 (a), (b), and (c) show performance improvement of distributed handshake scheme in synthetic workload traffic. Figure 34 (d), (e), and (f) show that the average token waiting time is reduced in the traffic patterns. Compared with token channel, token slot produces better performance. However, since the number of tokens depends on the number of credits at the destination, the destination node with full buffers will stop generating new tokens until a free buffer slot is available. Limited buffer space restrains the performance of token slot. Different from token slot, there is no credit-based flow control in the handshake schemes. Tokens are generated every cycle maximizing the transmission opportunity for senders, which is more efficient than token slot especially when destinations get free buffer space while packets are already in the middle of traversal.



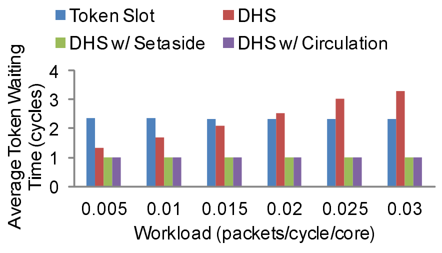
(a) UR



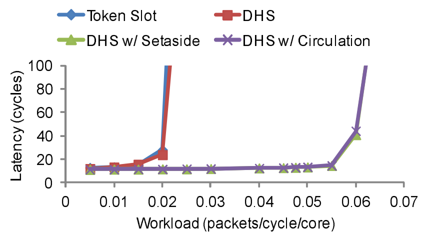
(d) UR



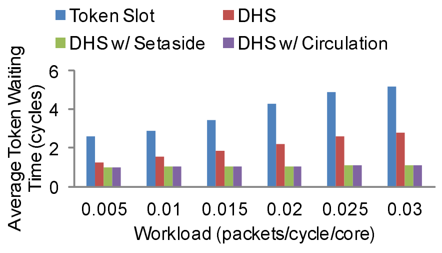
(b) BC



(e) BC



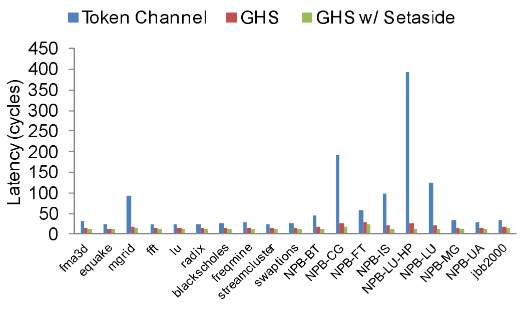
(c) TOR



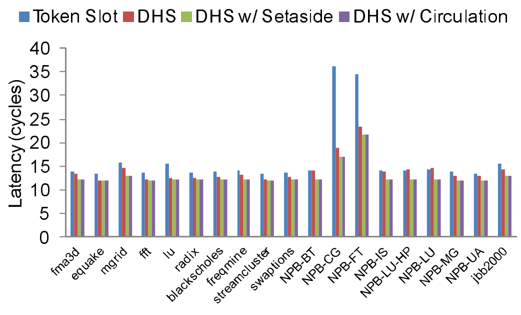
(f) TOR

Figure 34. Performance Evaluation of Distributed Handshake

The HOL blocking problem affects the performance of basic GHS and DHS. Although there are free tokens, the following flits cannot seize a token because the flit in the head of the queue is waiting for the acknowledgment. This situation becomes more obvious in the peer-to-peer communication patterns such as BC. From Figure 34 (b), we can see that token slot outperforms basic DHS. With the setaside buffer technique, flits can wait for the acknowledgments in the setaside buffer, yielding the chances to following flits, which brings significant throughput improvement. The setaside buffer and circulation techniques have almost the same effect on relieving the HOL blocking. However, compared with the setaside buffer technique, the circulation does not require additional buffer space, and is a more promising design.



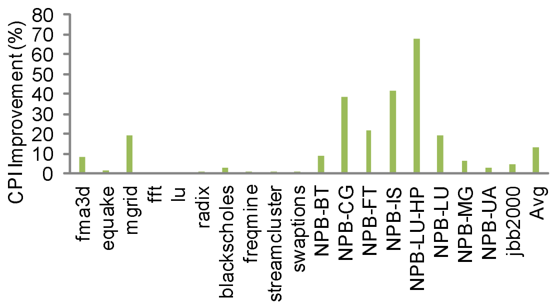
(a) Global Handshake



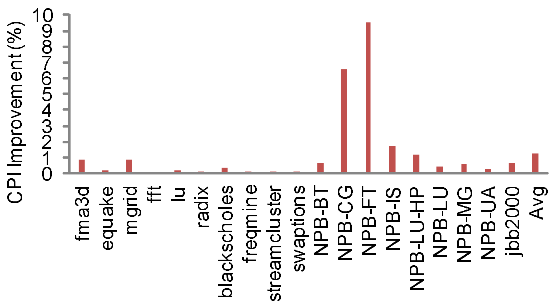
(b) Distributed Handshake

Figure 35. Performance Evaluation with Real Application

Real Applications: Figure 35 shows the performance results with real applications. It is clear that the handshake schemes produce obvious performance improvement, especially in NAS parallel benchmarks. Compared with token channel, GHS reduces communication latency by an average of 55%, while DHS achieves an average of 17% latency reduction over token slot. Suffering from the HOL blocking problem, basic GHS and DHS cannot perform as well as GHS and DHS with the setaside buffer and circulation techniques. However, in most of the selected benchmarks, basic GHS and DHS outperform the previous schemes. To study the effect of the handshake schemes on the system performance, we evaluate CPI as depicted in Figure 36. In this experiment, we select the handshake schemes with the setaside buffer technique to compare with token channel and token slot separately. GHS improves the CPI by an average of 13% compared with token channel, while DHS gets 1.3% CPI improvement over token slot.



(a) Global Handshake



(b) Distributed Handshake

Figure 36. CPI Improvement with the Handshake Schemes

3. Power

Different from conventional electrical network designs, in which buffers and switches dominate the total power consumption [63], the power dissipated in nanophotonic on-chip networks is composed of electrical router power, modulation/demodulation power, laser power and ring tuning power. Laser power and ring tuning power are also known as static power which dominates the overall power consumption. Modulation/demodulation power is determined by the number of E/O and

O/E conversions. Table 5 shows the energy costs of electrical back-end for optical links (modulator drives, receivers, and clocking), and we use 158fJ/b as the energy cost for each signal conversion. To calculate the laser power, we consider the E/O conversion losses as well as transmission losses in the waveguide.

Table 5. Estimated Energy of Electrical Back-End for Optical Links [60]

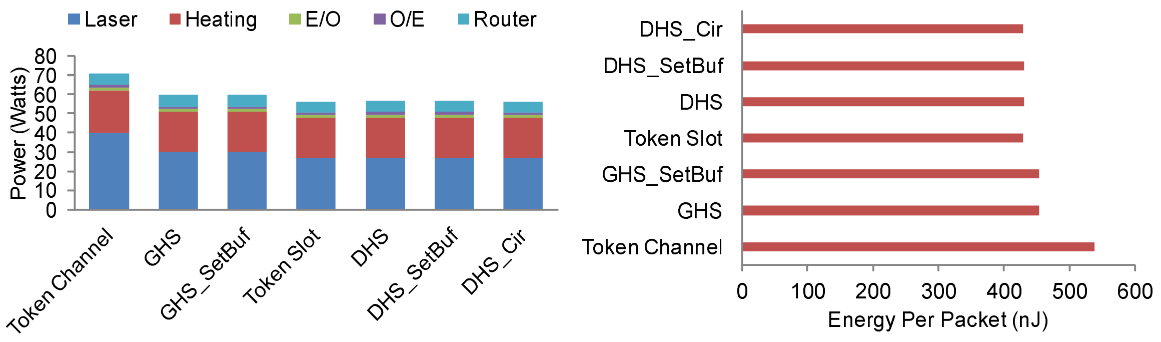
Component	**Energy (fJ/b)**
Serializer	1.5
Pre-Driver	19.0
Push-Pull Modulator	70.0
Analog Receiver Front End	40.0
Flip-Flop Sampling & Monitoring	12.0
Deserializer	1.5
Optical Phase Control	2.0
Clock Phase Control	12.0
Total	**158.0**

Table 6 lists various optical losses in the optical laser power and the corresponding electrical laser power (30% conversion efficiency [61]). Along the optical critical path coupler loss, modulation insertion loss, and filter drop loss are independent of the network layout, size and topology. Waveguide loss is length-dependent. A non-linearity limit of 30mW at 1dB loss is assumed for waveguides. In Corona [52], waveguide and ring through losses are dominant, due to the long waveguides (9 cm) and large number of rings (4096 rings per data waveguide). We assume 10μW for the sensitivity of photodetectors [58]. Additionally, all rings in the system must be thermally tuned to maintain their resonance under on-die temperature variations. We assume 1μW tuning power per ring per K, and a temperature range of 20K [61]. We use Orion 2.0 [64] power model to estimate the power consumption of an electrical router.

Table 6. Optical Loss [61]

Component	**Loss**
Coupler	1.0 dB
Splitter	0.2 dB
Non-linearity	1.0 dB
Modulator Insertion	0.001 dB
Waveguide Loss	1.0 dB/cm
Waveguide Crossing	0.05 dB
Ring Through Loss	0.001 dB/ring
Filter Drop	1.5 dB
Photo Detector	0.1 dB

Figure 37 (a) shows the power comparison among different schemes. As expected, laser power and ring heating power are dominant in all the schemes. Because the schemes with global arbitration, such as token channel and GHS, have only one shared token circulated in the network, which incurs more optical loss, they consume more laser power than the schemes with distributed arbitration such as token slot and DHS. Given that the token in GHS does not carry credit information, GHS has less laser power consumption than token channel. Among all the schemes, token slot has the lowest power consumption because the handshake schemes add additional handshake waveguides. However, the power overhead introduced by additional handshake waveguides is negligible as shown in Figure 37 (a). Figure 37 (b) indicates the average energy consumption for delivering a packet. With the passive writing nature of nanophotonics, where the modulation is done by imprinting messages onto a laser beam rather than driving a whole channel, the circulation technique has nearly no energy overhead for delivering a packet.

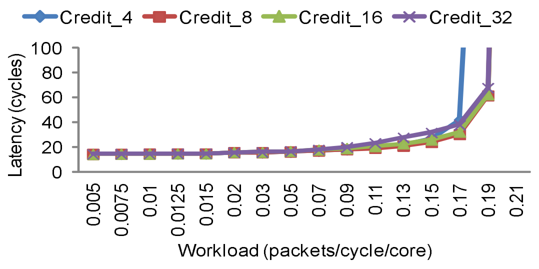


(a) Total Power Breakdown (b) Energy Consumption per Packet

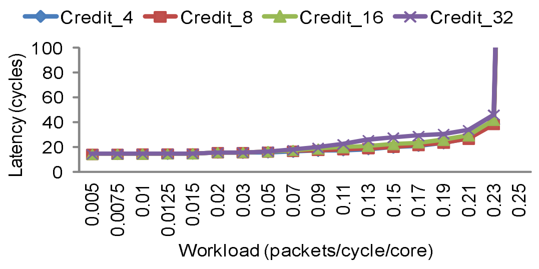
Figure 37. Energy and Power Analysis

4. Sensitivity Study

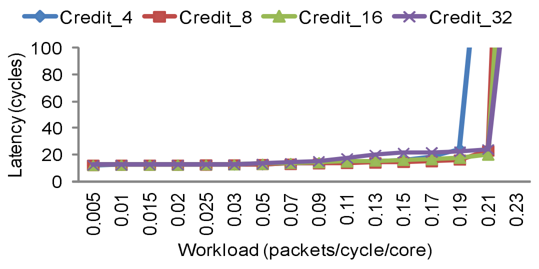
In this section, we present variations that provide insight into the performance of the handshake schemes in different environments. We select Uniform Random in this experiment. First, we evaluate the handshake schemes with various numbers of credits. Because in the handshake schemes, a token is used only for channel arbitration and no credit information is piggybacked, the performance of the handshake schemes is virtually independent of the number of credits, as shown in Figure 38 (a) to (e). Next, we analyze the performance of the handshake schemes with different sizes of the setaside buffer. Figure 38 (f) shows that the handshake schemes can produce comparable performance with only a small size of the setaside buffer.



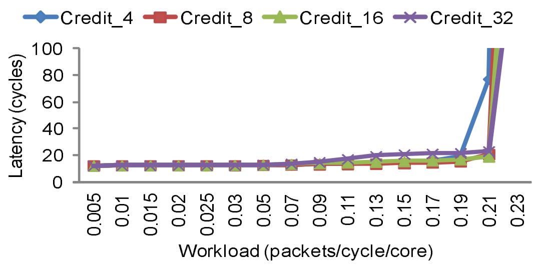
(a) GHS



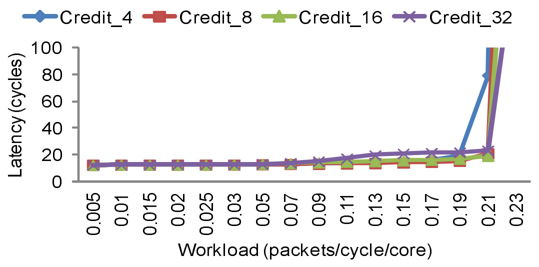
(b) GHS with Setaside Buffer



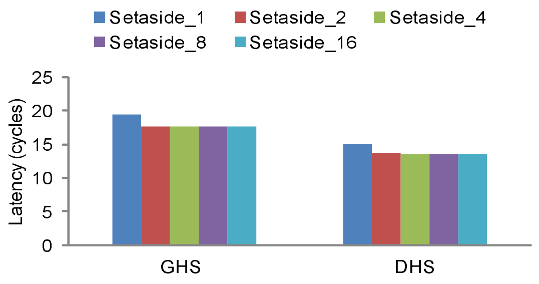
(c) DHS



(d) DHS with Setaside Buffer



(e) DHS with Circulation



(f) Setaside Buffer Size Study

Figure 38. Sensitivity Studies with Uniform Random Traffic

F. Related Work

Handshake has been widely used in the Internet. TCP/IP protocol adopts three-way handshake for reliable data transfer (RDT) [65]. In TCP/IP, a receiver sends an acknowledgment to the sender located thousands of miles away as a feedback after receiving a message. This acknowledgment does not provide flow control between senders and receivers. On-chip interconnect, which is considered to be reliable, also hires acknowledgment-based transmissions. In a circuit switching network, to set up a transmission circuit from source to destination, a routing probe is injected and traversing to the destination, which will send back an acknowledgment to notify the successful circuit set-up. SCARAB [66] introduces an optimized NACK network to provide retransmission in a bufferless network.

The emerging nanophotonic technology enables on-chip optical interconnect. Different on-chip network architectures have been proposed to exploit silicon nanophotonics. Kirman et al. [50] propose to use optical components to build on-chip buses. Shacham et al. [54] propose a circuit-switching photonic interconnect for data packets in parallel with an electric network. Nanophotonic switching is explored in the Phastlane [67]. The Corona [52] architecture implements a monolithic crossbar topology to support on-chip communication. Firefly [51] uses partitioned nanophotonic crossbars to connect clusters of electrically connected mesh networks. Joshi et al. [61] build a nanophotonics clos network, which provides uniform latency and throughput with low power.

Ha et al. [68] and Kodi et al. [69] advocate token-based protocols to arbitrate for optical off-chip interconnects. An optical arbiter can be found in [70]. Vantrease et al. [57] propose token channel and token slot for optical on-chip interconnects, which piggyback flow control information on the arbitration tokens. FlexiShare [58] reduces the number of channels across the network and proposes single-pass and two-pass token stream arbitration.

G. Conclusions

As the on-chip network size continues to increase, the bandwidth required to support concurrent computation on all cores increases by orders of magnitude. Optical interconnects have been leveraged to build various on-chip networks. In this chapter, we propose handshake schemes for nanophotonic interconnects, Global Handshake (GHS) and Distributed Handshake (DHS). By getting rid of the traditional credit-based flow control, GHS and DHS reduce the average token waiting time and improve the network throughput. To remove the HOL blocking problem existing in the basic handshake schemes, we propose the setaside buffer and circulation techniques, which improve the channel utilization further. Our evaluation shows that the proposed handshake schemes improve network throughput by up to 11× under synthetic workloads. For real applications, the handshake schemes can reduce the communication latency by up to 55%. The handshake schemes add only 0.4% hardware overhead for optical components and negligible power consumption. In addition, the performance of the handshake schemes are independent of on-chip buffer space, which makes them feasible in a large scale nanophotonic interconnect design.

CHAPTER IV

CONCLUSIONS

On-chip interconnection networks used in chip multiprocessors have brought another challenge to overcome their own weakness, such as limited on-chip resources, increasing communication latency, and insufficient communication bandwidth. To overcome the weakness, three schemes have been proposed to accelerate communication in on-chip interconnection networks.

First, the early transition scheme improves network throughput by increasing the utilization of escape channels in fully adaptive routing algorithms. Duato’s fully adaptive routing algorithm does not utilize the escape channels until normal channels are full, causing low utilization of the escape channels in on-chip interconnection networks. Transferring packets earlier to the escape channels increases overall utilization and consequently improves overall resource efficiency and network throughput.

Second, the pseudo-circuit scheme enhances communication latency by reducing per-hop router delay with communication temporal locality. It is observed that every application has a certain amount of communication temporal locality. With this locality, this scheme enables packets to bypass switch arbitration, thus reducing per-hop router delay. For further enhancement, two aggressive schemes have been proposed. Pseudo-circuit speculation generates more pseudo-circuits using currently unallocated crossbar connections for future communication. Buffer bypassing allows flits to skip buffer writes at input VC to eliminate one pipeline stage.

Third, two handshake schemes in nanophotonic interconnect improves network throughput by minimizing the average token waiting time. Due to uneven communication length in ring-based optical interconnects, it is difficult to synchronize the traditional flow control, causing inefficient link utilization. Removing the inefficient flow control minimizes the average token waiting time and consequently increases link utilization and enhances network throughput in nanophotonic interconnects.

REFERENCES

[1] The International Technology Roadmap for Semiconductors, http://www.itrs.net/, Accessed September 2008

[2] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-Ghz Mesh Interconnect for a Teraflops Processor," *IEEE Micro,* vol. 27, pp. 51-61, 2007.

[3] D. Wentzlaff, P. Griffin, H. Hoffmann, B. Liewei, B. Edwards, C. Ramey, M. Mattina, M. Chyi-Chang, J. F. Brown, and A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro,* vol. 27, pp. 15-31, 2007.

[4] K. Sankaralingam, R. Nagarajan, L. Haiming, K. Changkyu, H. Jaehyuk, D. Burger, S. W. Keckler, and C. R. Moore, "Exploiting Ilp, Tlp, and Dlp with the Polymorphous Trips Architecture," in *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)*, 2003, pp. 422-433.

[5] M. B. Taylor, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, and J. Kim, "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for Ilp and Streams," in *Proceedings on the 31st Annual International Symposium on Computer Architecture (ISCA)*, 2004, pp. 2-13.

[6] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 4, pp. 1320-1331, 1993.

[7] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[8] M. Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The Sgi Spider Chip," in *Proc. Hot Interconnects*, 1996, pp. 141-146.

[9] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers,* vol. C-36, pp. 547-553, 1987.

[10] W. J. Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 3, pp. 194-205, 1992.

[11] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi, "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks," in

Proceedings of the 32nd annual International Symposium on Computer Architecture (ISCA), 2005.

[12] J. Balfour and W. J. Dally, "Design Tradeoffs for Tiled Cmp on-Chip Networks," in *Proceedings of the 20th annual International Conference on Supercomputing (ICS)*, Cairns, Queensland, Australia, 2006.

[13] J. Kim, W. J. Dally, and D. Abts, "Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks," in *Proceedings of the 34th annual International Symposium on Computer Architecture (ISCA)* San Diego, California, USA: ACM, 2007, pp. 126-137.

[14] 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*, 2007.

[15] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," in *Proceedings of the 19th annual International Symposium on Computer Architecture (ISCA)* Queensland, Australia: ACM, 1992.

[16] J. H. Upadhyay, V. Varavithya, and P. Mohapatra, "Efficient and Balanced Adaptive Routing in Two-Dimensional Meshes," in *Proceedings of the First IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 1995, pp. 112-121.

[17] Y. M. Boura and C. R. Das, "Efficient Fully Adaptive Wormhole Routing in N-Dimensional Meshes," in *Proceedings of the 14th International Conference on Distributed Computing Systems (ICDC), 1994.*, 1994, pp. 589-596.

[18] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles, "Goal: A Load-Balanced Adaptive Routing Algorithm for Torus Networks," in *Proceedings of the 30th annual International Symposium on Computer Architecture (ISCA)* San Diego, California: ACM, 2003, pp. 194-205.

[19] A. Singh, "Load-Balanced Routing in Interconection Networks," *PhD Thesis, Stanford University,* 2005.

[20] J. Kim, W. J. Dally, and D. Abts, "Adaptive Routing in High-Radix Clos Network," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* Tampa, Florida: ACM, 2006, p. 92.

[21] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a High Radix Router," in *Proceedings of the 32nd annual International Symposium on Computer Architecture (ISCA)*, 2005, pp. 420-431.

[22] N. Jiang, J. Kim, and W. J. Dally, "Indirect Adaptive Routing on Large Scale Interconnection Networks," in *Proceedings of the 36th annual International Symposium on Computer Architecture (ISCA)* Austin, TX, USA: ACM, 2009, pp. 220-231.

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

[24] 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 *25th International Conference on Computer Design, ICCD*, 2007, pp. 63-70.

[25] 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*, San Diego, California, USA, 2007.

[26] A. Kumar, L. S. Peh, and N. K. Jha, "Token Flow Control," in *41st IEEE/ACM International Symposium on Microarchitecture, MICRO-41*, 2008, pp. 342-353.

[27] 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*, Munchen, Germany, 2004.

[28] N. E. Jerger, L.-S. Peh, and M. Lipasti, "Circuit-Switched Coherence," in *International Symposium on Networks-on-Chip, NOCS*, 2008.

[29] D. Kroft, "Lockup-Free Instruction Fetch/Prefetch Cache Organization," in *Proceedings of the 8th Annual International Symposium on Computer Architecture, ISCA*, Minneapolis, Minnesota, United States, 1981.

[30] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A Full System Simulation Platform," *Computer,* vol. 35, pp. 50-58, 2002.

[31] H. Sullivan and T. R. Bashkow, "A Large Scale, Homogeneous, Fully Distributed Parallel Machine, I," *SIGARCH Comput. Archit. News,* vol. 5, pp. 105-117, 1977.

[32] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express Cube Topologies for on-Chip Interconnects," in *IEEE 15th International Symposium on High Performance Computer Architecture, HPCA*, 2009, pp. 163-174.

[33] H.-S. Wang, L.-S. Peh, and S. Malik, "Power-Driven Design of Router Microarchitectures in on-Chip Networks," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture, MICRO-36*, 2003.

[34] L. G. Valiant and G. J. Brebner, "Universal Schemes for Parallel Communication," in *Proceedings of the thirteenth annual ACM symposium on Theory of computing* Milwaukee, Wisconsin, United States: ACM, 1981.

[35] P.-J. Chuang, J.-T. Chen, and Y.-T. Jiang, "Balancing Buffer Utilization in Meshes Using a 'Restricted Area' Concept," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 13, pp. 814-827, 2002.

[36] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *Communications, IEEE Transactions on,* vol. 35, pp. 1347-1356, 1987.

[37] Specomp 2001 Benchmark Suite, http://www.spec.org/omp/, Accessed September 2008

[38] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The Parsec Benchmark Suite: Characterization and Architectural Implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques* Toronto, Ontario, Canada, 2008.

[39] Nas Parallel Benchmarks, http://www.nas.nasa.gov/Resources/Software/npb.html, Accessed September 2008

[40] Specjbb 2000 Benchmark, http://www.spec.org/jbb2000/, Accessed September 2008

[41] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The Splash-2 Programs: Characterization and Methodological Considerations," in *Proceedings of the 22nd annual International Symposium on Computer Architecture, ISCA*, S. Margherita Ligure, Italy, 1995.

[42] C. Kim, D. Burger, and S. W. Keckler, "An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated on-Chip Caches," in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, San Jose, California, 2002.

[43] S. Thoziyoor, N. Muralimanohar, and N. P. Jouppi, "Cacti 5.0," *HP Technical Reports,* 2007.

[44] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," in *Proceedings. 35th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-35*, 2002, pp. 294-305.

[45] K. S. Shim, M. H. Cho, M. Kinsy, T. Wen, M. Lis, G. E. Suh, and S. Devadas, "Static Virtual Channel Allocation in Oblivious Routing," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*: IEEE Computer Society, 2009.

[46] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and P. Li-Shiuan, "Research Challenges for on-Chip Interconnection Networks," *IEEE Micro,* vol. 27, pp. 96-108, 2007.

[47] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling," *SIGARCH Comput. Archit. News,* vol. 33, pp. 408-419, 2005.

[48] A. P. Jose and K. L. Shepard, "Distributed Loss-Compensation Techniques for Energy-Efficient Low-Latency on-Chip Communication," *Solid-State Circuits, IEEE Journal of,* vol. 42, pp. 1415-1424, 2007.

[49] M. F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and S. W. Tam, "Cmp Network-on-Chip Overlaid with Multi-Band Rf-Interconnect," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, 2008, pp. 191-202.

[50] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins, and D. H. Albonesi, "Leveraging Optical Technology in Future Bus-Based Chip Multiprocessors," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*: IEEE Computer Society, 2006, pp. 492-503.

[51] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating Future Network-on-Chip with Nanophotonics," in *Proceedings of the 36th annual international symposium on Computer architecture* Austin, TX, USA: ACM, 2009, pp. 429-440.

[52] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System Implications of Emerging Nanophotonic Technology," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, 2008, pp. 153-164.

[53] H. Gu, J. Xu, and Z. Wang, "Odor: A Microresonator-Based High-Performance Low-Cost Router for Optical Networks-on-Chip," in *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis* Atlanta, GA, USA: ACM, 2008, pp. 203-208.

[54] A. Shacham, K. Bergman, and L. P. Carloni, "On the Design of a Photonic Network-on-Chip," in *First International Symposium on Networks-on-Chip, NOCS*, 2007, pp. 53-64.

[55] X. Zhang and A. Louri, "A Multilayer Nanophotonic Interconnection Network for on-Chip Many-Core Communications," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 156-161.

[56] L. Zhang, M. Song, T. Wu, L. Zou, R. G. Beausoleil, and A. E. Willner, "Embedded Ring Resonators for Microphotonic Applications," *Opt. Lett.,* vol. 33, pp. 1978-1980, 2008.

[57] D. Vantrease, N. Binkert, R. Schreiber, and M. H. Lipasti, "Light Speed Arbitration and Flow Control for Nanophotonic Interconnects," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture* New York, New York: ACM, 2009, pp. 304-315.

[58] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel Sharing for an Energy-Efficient Nanophotonic Crossbar," in *2010 IEEE 16th International Symposium onHigh Performance Computer Architecture (HPCA)*, 2010, pp. 1-12.

[59] ANSI/IEEE, "Token Ring Access Method and Physical Layer Specifications," Technical Report, STD 802.51989.

[60] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth, M. Popovic, H. Li, H. Smith, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic, and K. Asanovic, "Building Manycore Processor-to-Dram Networks with Monolithic Silicon Photonics," in *Proceedings of the 2008 16th IEEE Symposium on High Performance Interconnects*: IEEE Computer Society, 2008, pp. 21-30.

[61] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic, "Silicon-Photonic Clos Networks for Global on-Chip Communication," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*: IEEE Computer Society, 2009, pp. 124-133.

[62] P. Kapur and K. C. Saraswat, "Comparisons between Electrical and Optical Interconnects for on-Chip Signaling," in *Proceedings of the IEEE 2002 International Interconnect Technology Conference, 2002.* , 2002, pp. 89-91.

[63] W. Hang-Sheng, P. Li-Shiuan, and S. Malik, "A Power Model for Routers: Modeling Alpha 21364 and Infiniband Routers," *Micro, IEEE,* vol. 23, pp. 26-35, 2003.

[64] 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 *Proceedings of the Conference on Design, Automation and Test in Europe* Nice, France: European Design and Automation Association, 2009, pp. 423-428.

[65] V. G. Cerf and R. E. Kahn, "A Protocol for Packet Network Intercommunication," *SIGCOMM Comput. Commun. Rev.,* vol. 35, pp. 71-82, 2005.

[66] M. Hayenga, N. E. Jerger, and M. Lipasti, "Scarab: A Single Cycle Adaptive Routing and Bufferless Network," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009. MICRO-42.* , 2009, pp. 244-254.

[67] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi, "Phastlane: A Rapid Transit Optical Routing Network," in *Proceedings of the 36th annual international symposium on Computer architecture* Austin, TX, USA: ACM, 2009, pp. 441-450.

[68] J.-H. Ha and T. M. Pinkston, "A New Token-Based Channel Access Protocol for Wavelength Division Multiplexed Multiprocessor Interconnects," *J. Parallel Distrib. Comput.,* vol. 60, pp. 169-188, 2000.

[69] A. K. Kodi and A. Louri, "A Scalable Architecture for Distributed Shared Memory Multiprocessors Using Optical Interconnects," in *Proceedings. 18th International Parallel and Distributed Processing Symposium*, 2004, p. 11.

[70] C. Qiao and R. G. Melhem, "Time-Division Optical Communications in Multiprocessor Arrays," in *Proceedings of the 1991 ACM/IEEE conference on Supercomputing* Albuquerque, New Mexico, United States: ACM, 1991, pp. 644-653.

VITA

Min Seon Ahn received his Bachelor of Science degree in electric engineering and Master of Science degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1995 and 1997, respectively and received PhD degree in computer science and engineering from Texas A&M University in 2012. Before he was admitted to Texas A&M University in September 2004, he had several work experiences as a software engineer in Korea. His research interests consist of computer architecture and on-chip interconnection networks including nanophotonics.

Mr. Ahn may be reached at 3112 TAMU, College Station, TX 77843-3112. His email is msahn@cse.tamu.edu.