Intra-Clustering: Accelerating On-Chip Communication for Data Parallel Architectures

Wen Yuan[1], Rahul Boyapati[2], Lei Wang[3], Hyunjun Jang[2], Yuho Jin[4], Ki Hwan Yum[2], and Eun Jung Kim[2]

[1]Samsung Austin Research Center wen.yuan@samsung.com
[2]Texas A&M University College Station {rahulboyapati, hyunjun, yum, ejkim}@cse.tamu.edu
[3]Yahoo! Inc. wanglei828@yahoo-inc.com
[4]New Mexico State University yuho@nmsu.edu

***Abstract*—Modern computation workloads contain abundant Data Level Parallelism(DLP), which requires specialized data parallel architectures, such as Graphics Processing Units(GPUs). With parallel programming models, such as CUDA and OpenCL, GPUs are easily to be programmed for non-graphics applications, and therefore become a cost-effective approach for data parallel architectures. The large quantity of available parallelism places a heavy stress on the memory system as the limited number of pins confines the number of memory controllers on the chip. This creates a potential bottleneck for performance scalability of the GPUs. To accelerate communication with the memory system, we propose the *Intra-Clustering* on-chip network for data parallel architectures, which is built upon a traditional two-dimensional electrical mesh network with memory controllers connected through a nanophotonic ring and compute cores grouped into different clusters. Our evaluations with CUDA benchmarks show that the Intra-Clustering architecture can improve communication delay by an average of 17%(up to 32%) and IPC by an average of 5%(up to 11.5%).**

I. Introduction

Modern computation workloads, such as computer vision, audio processing and graphics rendering, include abundant Data Level Parallelism (DLP). Specialized data parallel architectures are needed to accelerate codes with significant amounts of DLP. Advances in technology have made it possible to accommodate an increasing number of transistors on a die, enabling designers to integrate a vast number of diverse components on a single chip, which facilitates modern Graphic Processing Units (GPUs), such as NVIDIA's Tesla [1] and Fermi [2], to have a huge number of parallel processors. Parallel programming models, such as CUDA [3] and OpenCL [4], make it easier to program for non-graphics applications on modern GPUs. Therefore, GPUs have recently obtained attention as a cost-effective approach for data parallel architectures.

In the GPU paradigm threads which are supposed to communicate with each other are allocated and executed on compute cores in a tight-knit manner. In turn, threads running in different compute cores have minimal communication [5]. The large quantity of parallelism places a heavy stress on the memory system. While the number of compute cores keeps increasing with high transistor densities, the number of memory controllers (MCs) is limited by the available pins on a chip that increase by only 10% per year [6]. This results in the many-to-few-to-many traffic pattern [7], where majority of requests are sent from compute cores to MCs, and replies are sent reversely. Therefore, MCs become the hot spots in GPUs.

The sheer number of components on modern GPUs and the specific DLP communication pattern raise the issue of how to connect them. Networks-On-Chip (NOCs) have emerged as an efficient and scalable solution to the communication problem [8]. While many-core systems using electrical interconnects may not be able to meet scalability and high bandwidth, 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 leveraged to build various on-chip networks [9], [10], [11], [12].

Although interconnection networks have been well researched in Chip Multiprocessors (CMPs) [13], the NOC design for GPUs is still in the infant stage. To accelerate the communication with the memory system, we propose the *Intra-Clustering* on-chip network for data parallel architectures in this work, which is built upon a traditional 2D mesh electric network with all the MCs connected through a nanophotonic ring. Compute cores are grouped into different regions or clusters, in which they share the same host MC. Inside a cluster, compute cores communicate with the host MC using the electrical network. When accessing MCs in different clusters, compute cores should first send requests to the host MC that relays the requests to the destination MC through high speed a nanophotonic ring. Since the distance between a compute core and a host MC is normally one or two hops and the traversal delay in the optical ring is much smaller than electrical links, the communication delay between a compute core and any on-chip MC is nearly the same, which relieves the effect of imperfect threads and memory allocation on system performance. We model our

Intra-Clustering design with a modified version of GPGPU-sim [14]. Our evaluations with CUDA benchmarks show that the Intra-Clustering architecture can improve communication delay by average 17% (up to 32%) and IPC by average 5% (up to 11.5%).

The rest of this paper is organized as follows. Section II describes the detailed design of the Intra-Clustering architecture. We evaluate our design in Section III. Finally, we draw conclusions in Section IV.

II. Design and Implementation

In this section, we first describe a state-of-the-art NOC router architecture and ring-based nanophotonic interconnects. Then we elucidate the detailed Intra-Clustering NOC design for data parallel architectures.

A. State-of-the-art NOC Router Architecture

Figure 1 shows a state-of-the-art NOC router architecture [15] for a 2D mesh network. The main building blocks of the router are input buffers, route computation logic, Virtual Channel (VC) allocator, switch allocator, and a crossbar. To achieve high performance, routers process packets with four pipeline stages, which are routing computation (RC), VC allocation (VA), switch allocation (SA), and switch traversal (ST). Additionally, the buffer is managed using credit-based flow control, where downstream routers provide back-pressure to upstream routers to prevent buffer overflow. Low-latency router designs parallelize RC, VA and SA using lookahead routing [16] and speculative switch allocation [17]. These two modifications lead to two-stage or even single-stage [18] routers, which parallelize the various stages in the router. In this paper, we model the NOC router as a two-stage router.

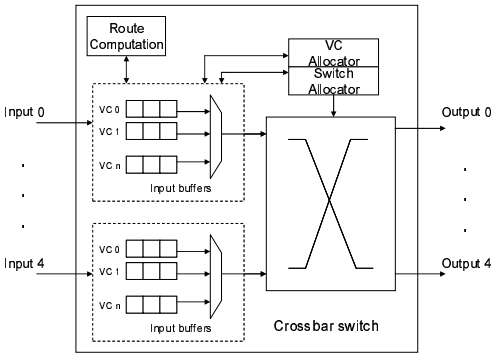


Figure 1. State-of-the-art NOC Router Architecture.

B. Nanophotonic Ring Interconnect

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 the optical signals. Light from the laser source travels unidirectionally in the waveguides with negligible losses. With dense-wavelength-division-multiplexing (DWDM), up to 128 wavelengths can be generated and carried by the waveguides [19]. 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. The modulation, detection and diversion are controlled by an electrical signal. Functioning ring resonators are described in [20] and Figure 2 shows a conceptual optical link.

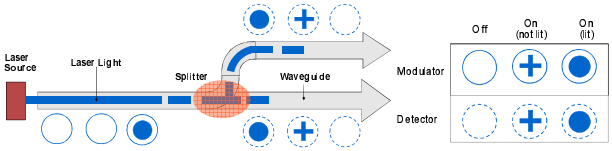


Figure 2. A Conceptual Optical Link.

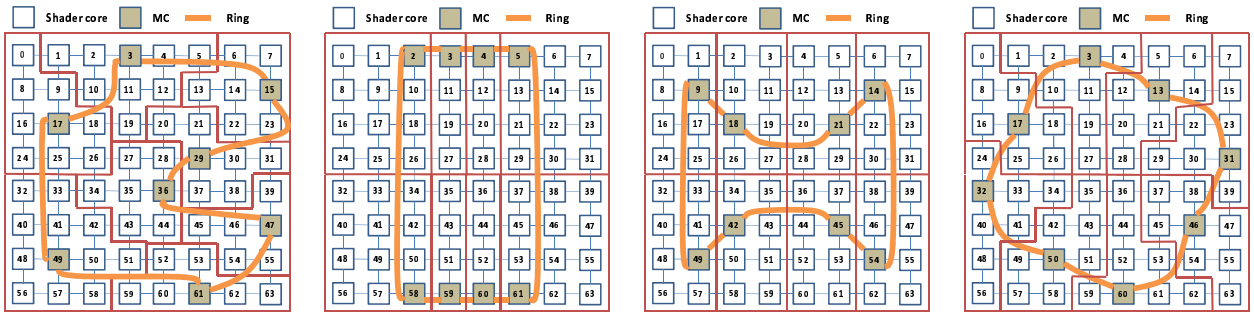
In nanophotonic interconnects, nodes are normally attached to a single communication media[1] forming a ring-based network. The ring-based optical interconnect falls into two categories: Multiple Write Single Read (MWSR) such as Corona [11], or Single Write Multiple Read (SWMR) such as Firefly [12]. Prior work [21], [22] adopts token-based arbitration, in which a photonic token represents the right of transmitting packets on a channel. In this work, we choose Token Slot [21], which takes care of the network flow control and arbitration, to build our ring-based MWSR network for MCs.

C. Intra-Clustering Architecture

In this section, we explain the details of building the Intra-Clustering architecture, including *clustering* and *routing*. Clustering divides the network into regions with different sizes. Routing takes care of the path selection between source and destination. With the nanophotonic ring, the Intra-clustering architecture provides more path diversities.

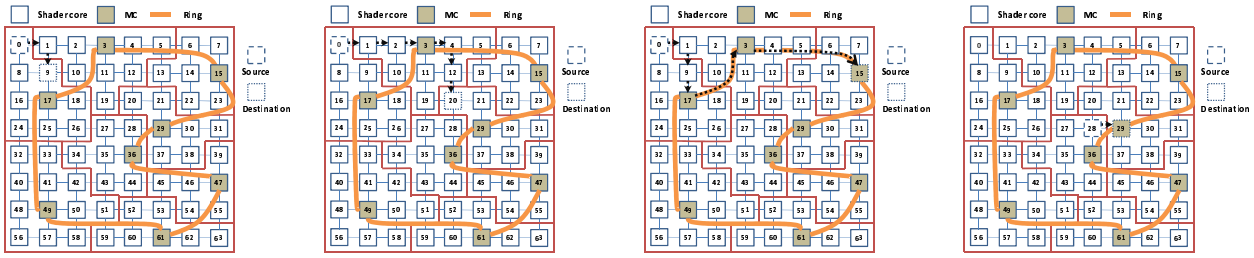
1) Clustering: As briefly explained in Section I, the Intra-Clustering architecture is built upon a 2D mesh topology with MCs connected using a nanophotonic ring. Figure 3 (a) shows an 8×8 mesh network with 56 Shader cores and 8 MCs (Node 3, Node 15, Node 17, Node 29, Node 36, Node 47, Node 49 and Node 61), which are distributed across the network. Since MCs become the hot spots in GPUs, besides being connected with the neighbor shader cores, MCs are hooked to a nanophotonic ring. According to its Manhattan distance from MCs, each shader core has at least one nearest MC. It is possible that more than two MCs have the same distance from a shader core, in such case the core randomly chooses one of those as its nearest MC. Multiple shader cores, which share the same nearest MC, are grouped into a cluster, and the corresponding MC is called the *host* MC. Each cluster will have one and at most one host

[1]This single communication media is composed of many separate channels, and each channel consists of a couple of waveguides.



(a) Dispersion (b) Row0_7 (c) Diagonal (d) Diamond

Figure 3. Clusterings with Different MC Placements.



(a) Internal Cluster (b) Crossing Cluster Core-to-Core (c) Core-to-MC (d) A Special Case

Figure 4. Communication Walk-through Examples in the Intra-Clustering Architecture.

MC. However, each cluster may have different numbers of nodes. Different placements of MCs can generate different clusterings of the network, as shown in Figure 3 (b), (c) and (d). With even distribution of MCs, each cluster can have similar number of nodes, and the hop count from a compute core to its host MC is one or two. In this work, we choose the placement in Figure 3 (a) as our default case, and different MC placements are explored in Section III.

2) Routing: Routing in the Intra-Clustering architecture falls into two categories: internal routing and external routing that include Core-to-MC, MC-to-Core, Core-to-Core and MC-to-MC.

Internal Routing: Internal routing handles the communication inside a cluster. If the source and destination nodes come from the same cluster, no matter what they are (shader cores or MCs), Dimension Order Routing (DOR) is employed. Since in a cluster the distance between two nodes is normally one or two hops, DOR can be very efficient.

External Routing: External routing takes care of the communication crossing the clusters. If the source and destination nodes are located in different clusters, selecting a path becomes more complicated.

- Core-to-MC: Packets are first routed to the host MC with DOR through electrical links; Then the host MC forwards the packets to the destination MC through the nanophotonic ring.
- MC-to-Core: The MC first sends packets to the host MC of the destination core through the nanophotonic ring; Then the host MC forwards the packets to the destination core using DOR.
- Core-to-Core: Considering that threads running on different compute cores have minimal intercommunication, to relieve the workload in the nanophotonic ring, communication between different compute cores is handled with DOR through electrical links.
- MC-to-MC: Since all the MCs are connected through the nanophotonic ring, MC-to-MC communication does not rely on electrical links and becomes pure optical communication.
- Special Cases: Considering such a case that a compute node wants to communicate with a MC located in another cluster but the distance between them is less than 2 hops, for example Node 28 and Node 29 in Figure 3 (a), external routing is not an optimal method. Therefore, before using external routing, source node should first calculate distance between source and destination. If it is less than a predefined threshold, set as two in this work, DOR is used. Otherwise, external routing is adopted.

To support the different routing methods, in each shader core, positions of its host MC and cores belonging to the same cluster should be recorded. The whole network clustering information needs to be stored in each MC, because the MC is used as an intermediate forwarding node. Since there are limited number of clusters, we believe that storing this extra information will cause negligible hardware

overhead.

3) Walk-through Examples: In this section, some communication examples shown in Figure 4 are detailed. We highlight the path from source to destination in each case.

(a) Source is Node 0, and destination is Node 9. Both of them are in the same cluster. Simple XY routing is used.
(b) Source is Node 0, and destination is Node 20. Two nodes are located in different clusters but both are shader cores. Still XY routing is used.
(c) Source is Node 0, and destination is Node 15. Node 0 is a shader core while Node 15 is a MC. Packets are first routed to Node 17, which is the host MC of Node 0, using XY routing. From Node 17 to Node 15, nanophotonic ring is used.
(d) Source is Node 28, and destination is Node 29. Node 28 is a shader core while Node 29 is a MC. Since the distance between two nodes is less than two hops, XY routing is used.

III. Experimental Results

In this section, we evaluate the proposed Intra-Clustering architecture. First, we describe our evaluation methodology. Then, the performance of Intra-Clustering is analyzed, and compared with the traditional design. Finally, we explore the architecture's sensitivity to different network parameters.

A. Methodology

We use a modified version of GPGPU-sim [14], a detailed cycle accurate simulator modeling a contemporary GPU running compute accelerator workloads. The modification is to model a nanophotonic ring network, which connects all the MCs, in the 2D mesh network. We choose a traditional 2D mesh electrical interconnection network as our baseline design. Since the nanophotonic ring is only connected with a small number of nodes, we assume the round trip delay for the ring is eight cycles[2]. The transmission delay from one MC to another on the ring is proportional to the distance between them. Table I describes the GPGPU-Sim simulator configuration we simulated in the experiments and the configuration of the interconnection network. We use six CUDA benchmarks in our evaluation. The benchmarks used were taken from the benchmark suite provided in [14]. To explore the effect of different network parameters on the Intra-Clustering architecture, we use synthetic workloads and a cycle accurate network simulator. Power consumption of the NoC is surely a major concern. Since the major goal of this work is to show that the overall performance of the GPU system can be improved by alleviating the memory controller bottleneck in the interconnection network, we rationalize that we would be better served by emphasizing the performance evaluation.

[2]In Corona [11], the round trip delay for a ring is 8 cycles

B. Performance with CUDA Benchmarks

In this section, we evaluate the performance of the Intra-Clustering architecture with CUDA benchmarks. We simulate all benchmarks to completion to capture all the distinct phases of each kernel in the benchmark.

Interconnection Latency: We first analyze the average communication latencies of the interconnection network, as shown in Figure 5. It is observed that the Intra-Clustering architecture dramatically improves the performance of the network. For BFS, MUM and STO, the Intra-Clustering architecture achieves 27%, 32% and 18% performance improvement, respectively. The major reason for the improvement is the nanophotonic ring, which go through all the MCs. The dominant communication in data parallel architectures is between shader cores and MCs. With the nanophotonic ring, each shader core, no matter where it is located, can reach every MC in a small number of cycles. It is similar to providing a crossbar connection between shader cores and MCs but with better scalability. However, in a traditional 2D mesh topology, packets should be routed hop by hop, which incurs a lot of intermediate network congestion.

Impact on IPC: To study the impact of our design on overall system performance, we evaluate the system IPC, which is estimated by dividing the dynamic instruction count measured in simulation by the product of measured runtime on hardware and the shader core clock frequency. Figure 6 shows the normalized IPC values. Due to the reduction of the communication delay with the nanophotonic ring, Intra-Clustering obtains improvement in system performance. Simulation results demonstrate that the Intra-Clustering design improves system performance by 5% on average and up to 11.5% (MUM).

C. Sensitivity study

In this section, we explore our design with different network parameters. Since real applications have varied dynamic behavior, in this study we use synthetic workloads, in which each node has stable packet injection rates.

First, we vary the ratio of the Core-to-MC communication among the overall on-chip communication, as shown in Figure 7. 90% indicates that 90% of the injected packets go to MC, while the rest 10% are sent to other cores. It is observed that the network throughput degrades as the ratio increases. With more Core-to-MC traffic, more packets use external-routing, which generates more competition for the ring resources, making the nanophotonic ring easily congested. It is suggested that adaptive routing can be applied to balance the workloads between electrical links and the nanophotonic ring, which is left for future work.

Secondly, we explore the Intra-Clustering design with different MC placements. We fix the injection rate and ratio of Core-to-MC traffic. In Figure 8, it is observed that Dispersion, Diagonal and Diamond deliver better performance than

Table I
HARDWARE CONFIGURATION.

Number of Shader Cores	56
Warp Size	32
SIMD Pipeline Width	8
Number of Threads / Core	1024
Number of CTAs / Core	8
Shared Memory / Core (KB)	16 (16 banks, 1 access/cycle/bank)
Constant Cache Size / Core (KB)	8 (2-way set associate. 64B lines LRU)
Texture Cache Size / Core (KB)	64 (2-way set associate. 64B lines LRU)
Number of Memory Channels	8
Bandwidth per Memory Module	8 (Bytes/Cycle)
Average Memory Fetch Latency	16
DRAM Request Queue Capacity	32
Memory Controller	Out of Order (FR-FCFS)
Branch Divergence Method	Immediate Post Dominator
Warp Scheduling Policy	Round-Robin among Ready Warps

Topology	Intra-Clustering
Routing Algorithm	Dimension Order
Virtual Channels	4
Virtual Channel buffers	8
SW Allocator	iSLIP
Allocation Iteration	1
Input Speedup	1
Flit Size (Bytes)	16

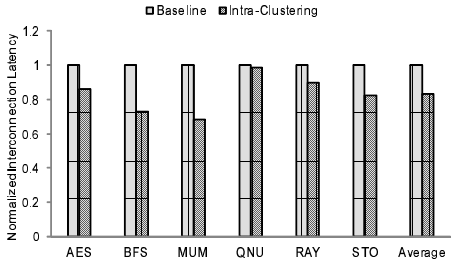


Figure 5. Normalized Interconnection Latencies with CUDA Benchmarks.

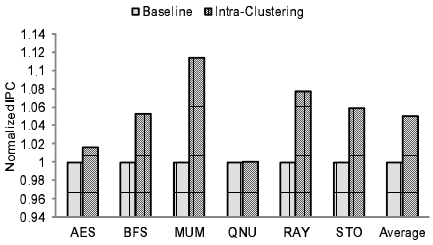


Figure 6. Normalized IPC.

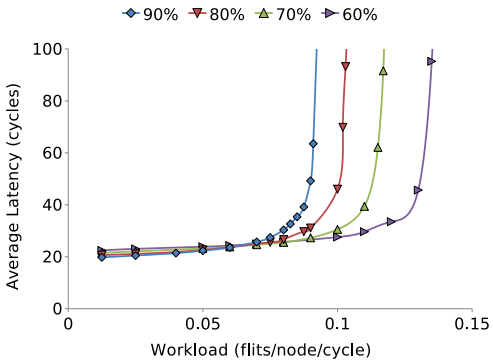


Figure 7. Performance of the Intra-Clustering Architecture With Different Ratios of Core-to-MC Traffic.

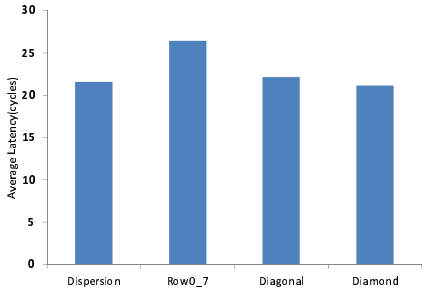


Figure 8. Average Communication Latencies with Different MC Placements.

Row0_7. Because in Dispersion, Diagonal and Diamond, every cluster has a similar number of components. The distance from a shader core to a host MC is one or two hops. However, within Row0_7, MCs are located on the edge of the chip, making the top and bottom clusters to have a large number of shader cores. Although the nanophotonic ring can provide low latency, internal delay increases in Row0_7 which degrades the overall network performance. Therefore, to benefit from the Intra-Clustering design, even distribution of MCs is preferred.

Finally, we research the performance of the proposed design with different network sizes. We fix the injection rate and maintain the same number of MCs in all the networks.Figure 9 shows that from 6×6 to 10×10, the average communication delay increases by 30% in the our design, while increasing by 39% in the baseline. We can conclude that compared with the baseline design, the Intra-Clustering architecture is more scalable with network sizes.

IV. CONCLUSIONS

Advances in technology have made it possible to accommodate an increasing number of transistors on a die,

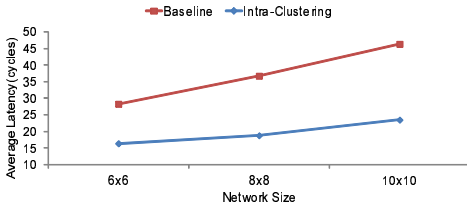


Figure 9. Average Communication Latencies with Different Network Sizes.

enabling designers to integrate a vast number of diverse components on a single chip, which facilitates modern Graphic Processing Units (GPUs). Data Level Parallelism dominates a variety of modern computation workloads. With parallel programming models, such as CUDA and OpenCL, GPUs become easily to be programmed for non-graphics applications, and therefore obtain attention as a cost-effective approach for data parallel architectures. Limited by the number of available pins, fewer MCs can be located on a chip compared with the massive number of processor cores. The large quantity of parallelism places a heavy stress on the memory system. In this work, we propose the *Intra-Clustering* on-chip network for data parallel architectures, in which MCs are connected with a nanophotonic ring, and compute cores are grouped into different clusters. Our evaluations with CUDA benchmarks show that the Intra-Clustering architecture can improve communication delay by average 17% (up to 32%) and IPC by average 5% (up to 11.5%). From the sensitivity studies, it is suggested, that to relieve the high workload in the nanophotonic ring, adaptive routing can be applied to the Intra-Clustering architecture, which is left as our future work.

REFERENCES

[1] E. Lindholm, J. Nickolls, S. F. Oberman, and J. Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture," *IEEE Micro*, vol. 28, no. 2, pp. 39–55, 2008.

[2] NVIDIA, "NVIDIA's Next Generation CUDA Compute Architecture: Fermi," September 2009.

[3] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," *ACM Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[4] K. Group, "OPENCL-The Open Standard for Parallel Programming of Heterogeneous Systems." http://www.khronos.org/opencl/.

[5] J. H. Kelm, D. R. Johnson, S. S. Lumetta, S. J. Patel, and M. I. Frank, "A Task-Centric Memory Model for Scalable Accelerator Architectures," *IEEE Micro*, vol. 30, no. 1, pp. 29–39, 2010.

[6] International Technology Roadmap for Semiconductors. 2008 Update., http://www.spec.org/omp/.

[7] D. Abts, N. D. E. Jerger, J. Kim, D. Gibson, and M. H. Lipasti, "Achieving Predictable Performance Through Better Memory Controller Placement in Many-Core CMPs," in *ISCA*, 2009, pp. 451–461.

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

[9] A. Shacham, K. Bergman, and L. P. Carloni, "On the Design of a Photonic Network-On-Chip," in *NOCS*, 2007, pp. 53–64.

[10] H. Gu, J. Xu, and Z. Wang, "ODOR: A Microresonator-Based High-Performance Low-Cost Router for Optical Networks-On-Chip," in *CODES+ISSS*, 2008, pp. 203–208.

[11] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. L. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System Implications of Emerging Nanophotonic Technology," in *ISCA*, 2008, pp. 153–164.

[12] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. N. Choudhary, "Firefly: Illuminating Future Network-On-Chip with Nanophotonics," in *ISCA*, 2009, pp. 429–440.

[13] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh, "Research Challenges for On-Chip Interconnection Networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, 2007.

[14] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," in *ISPASS*, 2009, pp. 163–174.

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

[16] M. Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER chip," in *Proceedings of Hot Interconnect 4*, 2009, pp. 141–146.

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

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

[19] X. Zhang and A. Louri, "A Multilayer Nanophotonic Interconnection Network for On-Chip Many-Core Communications," in *DAC*, 2010, pp. 156–161.

[20] L. Zhang, M. Song, T. Wu, L. Zou, R. G. Beausoleil, and A. E. Willner, "Embedded Ring Resonators for Microphotonic Applications," *Optics Letters*, vol. 33.

[21] D. Vantrease, N. L. Binkert, R. Schreiber, and M. H. Lipasti, "Light Speed Arbitration and Flow Control for Nanophotonic Interconnects," in *MICRO*, 2009, pp. 304–315.

[22] Y. Pan, J. Kim, and G. Memik, "FlexiShare: Channel Sharing for an Energy-Efficient Nanophotonic Crossbar," in *HPCA*, 2010, pp. 1–12.