Photonic Interconnects Beyond High Bandwidth

Ke Wen

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Photonic Interconnects Beyond High Bandwidth

Ke Wen

The extraordinary growth of parallelism in high-performance computing requires efficient data communication for scaling compute performance. High-performance computing systems have been using photonic links for communication of large bandwidth-distance product during the last decade. Photonic interconnection networks, however, should not be a wire-for-wire replacement based on conventional electrical counterparts. Features of photonics beyond high bandwidth, such as transparent bandwidth steering, can implement important functionalities needed by applications. In another aspect, application characteristics can be exploited to design better photonic interconnects. Therefore, this thesis explores codesign opportunities at the intersection between photonic interconnect architectures and high-performance computing applications. The key accomplishments of this thesis, ranging from system level to node level, are as follows.

Chapter 2 presents a system-level architecture that leverages photonic switching to enable a reconfigurable interconnect. The architecture, called Flexfly, reconfigures the inter-group level of the widely-used Dragonfly topology using information about the application's communication pattern. It can steal additional direct bandwidth for communication-intensive group pairs. Simulations with applications such as GTC, Nekbone and LULESH show up to 1.8x speedup over Dragonfly paired with UGAL routing, along with halved hop count and latency for cross-group messages. To demonstrate the effectiveness of our approach, we built a 32-node Flexfly prototype using a silicon photonic switch connecting four groups and demonstrated 820 ns interconnect reconfiguration time. This is the first demonstration of silicon photonic switching and bandwidth steering in a high-performance computing cluster.

Chapter 3 extends photonic switching to the node level and presents a recon-

figurable silicon photonic memory interconnect for many-core architectures. The interconnect targets at important memory access issues, such as network-on-chip hot-spots and non-uniform memory access. Integrated with the processor through 2.5D/3D stacking, a fast-tunable silicon photonic memory tunnel can transparently direct traffic from any off-chip memory to any on-chip interface – thus alleviating the hot-spot and non-uniform access effects. We demonstrated the operation of our proposed architecture using a tunable laser, a 4-port silicon photonic switch (four wavelength-routed memory channels) and a 4x4 mesh network-on-chip synthesized by FPGA. The emulated system achieves a 15-ns channel switching time. Simulations based on a 12-core 4-memory model show that for such switching speeds the interconnect system can realize a 2x speedup for the STREAM benchmark in the hot-spot scenario and a reduction of execution time for data-intensive applications such as 3D stencil and K-means clustering by 23% and 17%, respectively.

Chapters 4 explores application-level characteristics that can be exploited to hide photonic path setup delays. In view of the frequent reuse of optical circuits by many applications, we proposed a circuit-cached scheme that amortizes the setup overhead by maximizing circuit reuses. In order to improve circuit "hit" rates, we developed a reuse-distance based replacement policy called "Farthest Next Use". We further investigated the tradeoffs between the realized hit rate and energy consumption. Finally, we experimentally demonstrated the feasibility of the proposed concept using silicon photonic devices in an FPGA-controlled network testbed.

Chapter 5 proceeds to develop an application-guided circuit-prefetch scheme. By learning temporal locality and communication patterns from upper-layer applications, the scheme not only caches a set of circuits for reuses, but also proactively prefetches circuits based on predictions. We applied this technique to communication patterns from a spectrum of science and engineering applications. The results show that setup delays via circuit misses are significantly reduced, showing how the proposed

technique can improve circuit switching in photonic interconnects.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor, Professor Keren Bergman, for advising my PhD study. Without her support and guidance, I would not be able to work in such an interesting field.

I also want to acknowledge US Department of Energy, Sandia Nation Labs and Lawrence Berkeley Lab for sponsoring my research. Special thanks to Jeremiah Wilke and John Shalf who led me into the core of high performance computing and provided valuable comments.

To my colleagues in Lightwave Research Lab, I owe them a lot for the generous help in all aspects. I would like to thank Sébastien Rumley, David Calhoun, Payman Samadi, Christine Chen, Hang Guan, Meisam Bahadori, Yiwen Shen and many others for their collaboration and contribution in my research projects.

I would also like to thank all the members in the dissertation committee, Professor Keren Bergman, Professor Luca Carloni, Professor Michal Lipson, Professor Vishal Misra, and John Shalf.

Chapter 1

Introduction

1.1 Communication Requirement within HPC

The performance of extreme-scale high-performance computing (HPC) systems relies heavily on the interconnection network as concurrency increase results in massive data exchange between network endpoints [1]–[4]. Designing networks that properly balance compute capability is challenging: over-provisioning the network incurs unnecessary cost [5], while under-provisioning negates the benefits of extra concurrency.

A relatively simple way to denote the computer capability is the number of floating point operations per second (FLOP/s), measured by the LINPACK benchmark. Since the stall of clock frequency around 2004 [6] due to the end of Dennard Scaling [7], increasing the number of cores has been the new driving force for performance improvement. By decomposing a problem to a larger number of "workers", reduction of total execution time is expected. The reduction, however, is often disproportional to the core increase, as parallelism implies communication. With the extraordinary growth in parallelism at all system scales, the performance of today's systems is increasingly determined by how data is communicated among the numerous compute resources rather than by the total raw computation resources available. Assuming

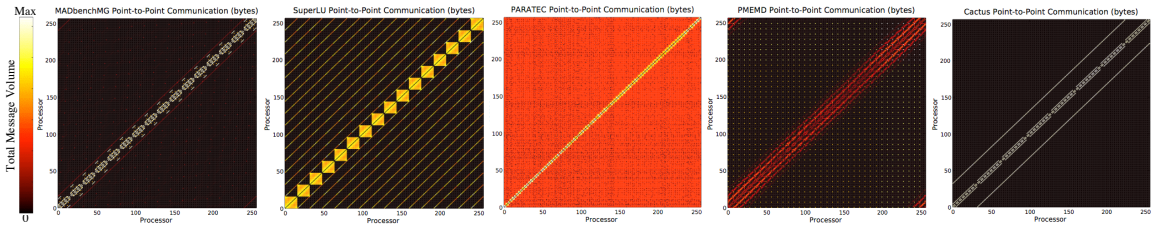


Figure 1.1: Diverse communication patterns of the "Seven Dwarf" applications identified by Phillip Colella [8] and a group of experts from Berkeley [9]. Figure compiled from [9], [10].

a 0.01 byte/FLOP node verbosity, i.e. each node injects 0.01 byte of data into the network per FLOP, a future exascale supercomputer (10^{18} FLOP/s) will require 10 PB/s data movement capability. This number is about three orders of magnitude higher than today's global Internet bandwidth usage (about 28 TB/s, 2015).

The need for high bandwidth is not the only aspect. In HPC, the application communication pattern can be highly diverse. In 2004, Phillip Colella [8] identified "seven dwarfs" – seven numerical methods important for scientific computing – which have distinctive communication patterns [9], [10] (see Fig. 1.1). In production runs, the patterns can further diverge due to different domain decomposition strategies, mapping strategies, communication algorithms, etc. Such diversity poses a big challenge to interconnect design, since having a single topology that works well for all applications is almost impossible. Over the last decade, the interconnect topology has seen a transition from multidimensional torus (as in IBM BlueGene/P and BlueGene/Q) to Dragonfly (as in Cray XC series). While each generation works best for a certain class of applications – for example, the BlueGene 5D torus for neighbor intensive applications and the Cray Dragonfly for all-to-all applications like FFT – they often fail to do so for another class. It is thus desirable to have a *reconfigurable network* that can create different topologies for different applications – just as how *reconfigurable computing* (like FPGA) adapt for heterogeneous compute needs.

1.2 The Memory Wall in Post Moore's Law and Many-core Era

The critical role of memory in computing roots in the long-live von Neumann model, where the memory acts as the source and sink of the processing unit. When the flow injection or egression gets congested, the processing unit will have to stall. Memories in exascale have an even higher bandwidth demand than the system interconnect for its higher byte-per-FLOP ratio (0.1 byte/FLOP in general). That is, an exascale machine will need 100 PB/s memory access bandwidth in total.

Due to the lag-behind of DRAM performance scaling as compared to transistor performance scaling, there has been a "memory wall" in the last two decades. A major remedy for this has been putting more caches on the processor chip, which is enabled by Moore's Law scaling. However, Moore's Law is ending as we enter the last years of shrinking transistors. Chip designers will thus have to use the available transistors more effectively. One may interpret a few already signs of this trend. As shown in Fig. 1.2, area devoted to cache on a supercomputer processor chip is decreasing, both in terms of (a) MB per FLOP/s and (b) normalized chip area – cache size (MB) $\times$ features size (nm^2) / die size (mm^2). Especially, a sharp fall is clear as the industry gets into the many-core era (around 2013). Interestingly, this cache cliff matches the time when Moore's Law was said to be dead in the economic sense – starting from 2013, the number of transistors bought per dollar has stayed flat. The fact that chipmakers are willingly trading the cache area for more FLOPs, along with the rise of data-centric throughput computing, calls for much higher off-chip memory bandwidth. Fig. 1.3 shows this trend: the sharp increase of the off-chip memory bandwidth matches the cache cliff of Fig. 1.2. This increase, however, is still not enough to balance the FLOPS increase as the bytes per flop ratio continues to drift away from the ideal point.

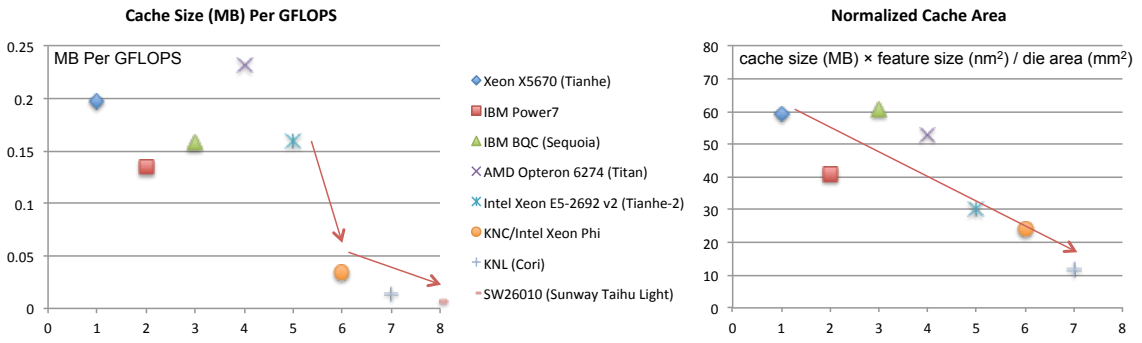


Figure 1.2: (left) Cache size normalized by GFLOPS. (right) Normalized cache area.

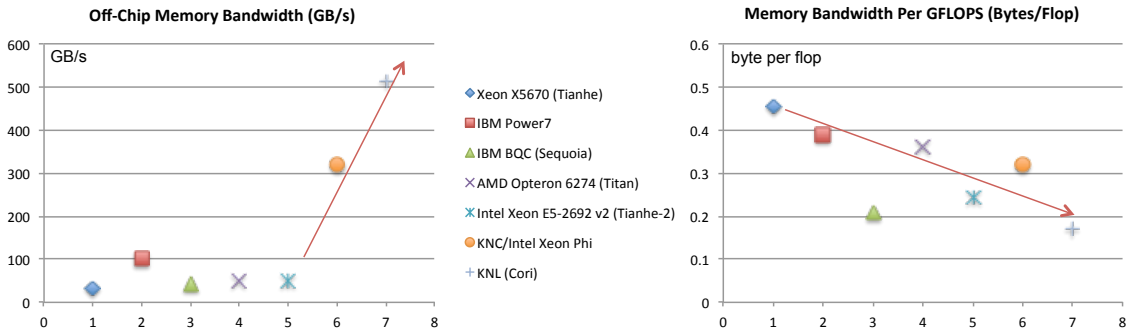


Figure 1.3: (left) Absolute off-chip memory bandwidth. (right) off-chip byte per FLOP ratio.

The memory bandwidth increase is also rapidly stressing the pin count limit of the processor package. For example, KNL requires 3647 pins in the socket, plus 1024 pins in the interposer for each of the eight on-package memory stacks. The pin density of standard chip package, however, cannot scale indefinitely. The ever-increasing bandwidth demand thus requires a more efficient chip I/O technology for processors beyond the Moore's Law.

There is more to this grim description. As many-core architectures start to get widely adopted, effects such as hotspots and non-uniform-memory-access (NUMA) emerge as a result of using network-on-chip to access memory [11]. These effects, in addition to the bandwidth bottleneck, place a detrimental scaling barrier to data-intensive computing. Moreover, they require careful designs that can accurately de-

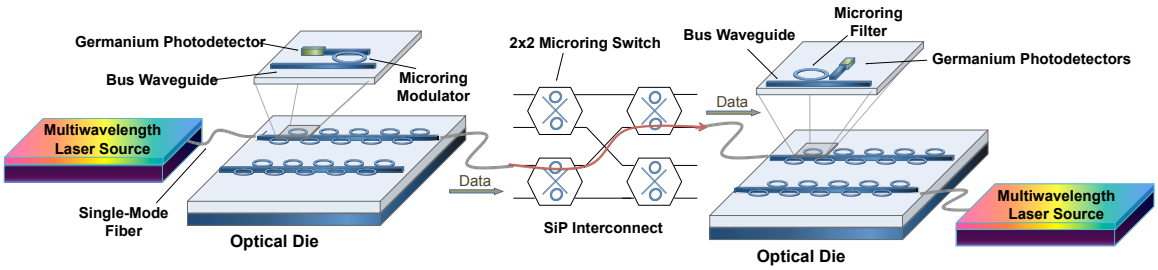


Figure 1.4: Microring-based silicon photonic link.

liver the bandwidth to where needed, rather than purely increasing the memory bandwidth in a brute-force way.

1.3 Photonic Interconnects

Optical interconnects have emerged as a promising technology for large-scale systems as they can support transmission of very high data rates in an energy-efficient manner. By using wavelength division multiplexing (WDM), a single optical fiber or waveguide can transmit tens of independent channels achieving terabit/s aggregate bandwidth. In addition, due to the low loss of the optical medium, the high bandwidth data movement can scale over warehouse distances and provide end-to-end connectivity in a distance-agnostic manner. Photonic interconnects thus have the potential to satisfy the bandwidth demand for exascale.

Silicon photonic technology, in particular, presents a potential platform for realizing photonic components in a cost-effective manner. The technology uses silicon rather than conventional materials as the core transmission medium. It thus allows the formation of optical communication devices (including modulators, switches, filters, detectors, etc.) in the same materials and process platform as used in the complementary metal oxide semiconductor (CMOS) industry [12]. This capability brings many advantages, including small device footprints, low power consumption and the potential for low-cost large-scale integration.

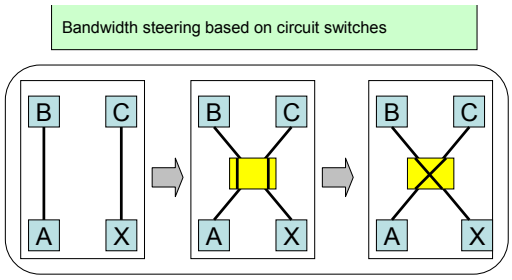
Fig. 1.4 shows a silicon photonics link. It consists of modulators, switches, wavelength demultiplexer and photo detectors. Modulators are usually based on a ring structure. By tuning the ring's resonance on or off with respect to the laser wavelength, on-off keying can be implemented. For fast modulation, mechanisms that change the carrier density within the optical mode are often employed, including carrier injection, depletion and accumulation [13]. The carrier density changes the refractive index of the waveguide and hence the resonance wavelength of the ring. By cascading multiple rings on the same waveguide working on different resonance, WDM transmission is possible. In this case, each ring modulates a different wavelength.

At the reception side, a wavelength demultiplexer is required to separate the wavelengths, i.e. dropping them at different ports. The demux can be also based on the ring structure. Different from modulation, the tuning of the demux (as well as any other switching devices) might not need to operate as fast. Thermal tuning methods are thus widely employed. Heating or cooling the ring will change its dimension, hence the distance experienced by light and hence the resonance.

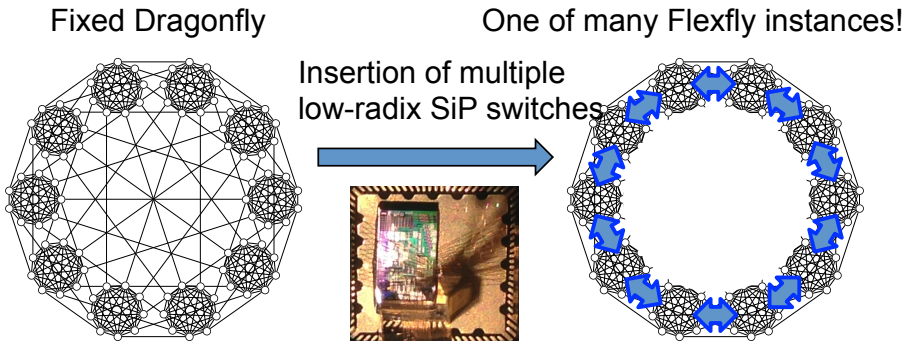
1.4 Beyond Just High Bandwidth

Due to the lack of practical buffers in the optical domain, current high-performance interconnection networks employ optical interconnects to transmit data while electronic routers are used for switching and routing. While these designs resolve the bandwidth $\times$ distance problem, they require multiple optical-electronic-optical (O/E/O) conversions at the interfaces. As the power consumption of the electronic switching and the O/E/O interfaces scale poorly with the increased serial data rate, such switch designs might not be able to scale in a power efficient manner to address the growing communication needs of HPC.

As an alternative solution, electronically controlled optical switching (Fig. 1.4),



(a)



(b)

Figure 1.5: (a) Rewiring links or steering bandwidth using a circuit switch. (b) Photonics enabled flexible topologies.

in which the data signals remain primarily in the optical domain, have been proposed and demonstrated [14]–[18]. In these photonic switches, the energy consumption is essentially decoupled from the data rate [19] and can therefore scale in an energy efficient manner. Moreover, the switches can provide bandwidth steering (i.e. dynamic rewiring) functionality for their electrical clients. See Fig. 1.5a for a simple example based on a 2-by-2 switch. We can hence utilize this bandwidth steering/rewiring feature to reconfigure network topologies for the diverse communication requirements, covering, for example, both all-to-all and neighbor-intensive patterns (Fig. 1.5b). Part I of this thesis ("Leveraging Photonic Switching") will explore this direction in the context of system interconnect (Chapter 2) and memory interconnect (Chapter 3), respectively.

Another feature of photonic interconnects beyond just bandwidth is its end-to-end connectivity. In a network with node-to-node optical connections, energy dissipation

due to electrical packet switching can be eliminated, and network architecture flattened. However, due to the circuit switched nature, such network may suffer from path setup time. Part II of this thesis ("Avoiding Path Setup Overhead") develops remedy approaches for this aspect by observing and leveraging application-level characteristics. Chapters 4 and 5 present passive and active latency-hiding techniques, respectively.

Part I:

Leveraging Photonic Switching

Chapter 2

Photonics Enabled Flexible Topology – A Dragonfly Example

2.1 Motivation

The Dragonfly network [20], [21] has emerged as a low-diameter, high-radix solution for HPC interconnects. A Dragonfly has two levels: in the lower level, intra-group routers are connected in local networks called groups, usually in an all-to-all or 2D-flattened butterfly (2D-FB) topology [22]; in the upper level, groups are connected through an all-to-all or high-degree topology.

With its high-connectivity at both levels, Dragonfly can greatly reduce the network diameter over topologies such as multidimensional torus or fat tree. A Dragonfly can connect any two routers within a distance of 5 if the intra-group network is a 2D-FB or a distance of 3 if an all-to-all network. A price for the high-connectivity, however, is diluted per-link bandwidth. In particular, the bandwidth of inter-group (global) links, carrying all the traffic between two large sets of routers, becomes the most scarce resource and can become bottleneck for the entire network [23]. Many scientific applications on a Dragonfly platform, unfortunately, tend to concentrate

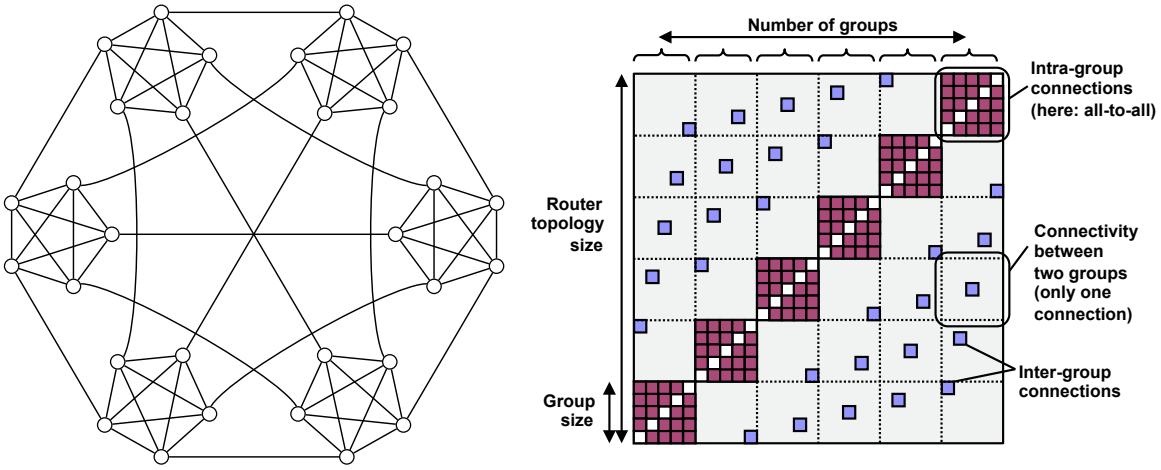


Figure 2.1: (a) A six-group Dragonfly, all-to-all both inter- and intra- group. (b) Connectivity matrix.

traffic on only a few of these links. +1/-1 neighbor-group based communication pattern, for example, is prevalent for many applications (detailed later). Moreover, inter-group traffic matrix is often sparse, leading to bandwidth allocated for idle pairs.

Ideally, total available bandwidth should be allocated where most needed. Being a fixed topology, however, Dragonfly can only try to achieve this through global or adaptive routing. Numerous routing strategies [20], [24]–[29] have been proposed, but they result in longer-distance paths and cross-group interference [30]–[32].

2.2 The Dragonfly Topology

Dragonfly is a direct topology that partitions S routers into G groups. Figure 2.1a illustrates a six-group Dragonfly. Each router is connected to C compute nodes (not shown in the figure), enabling a direct network of $N = C * S$ total compute nodes.

Links in Dragonfly are segregated into two *tiers*. Bottom-tier links connect routers of the same group, they are thus *local*, *intra-group* links. Top-tier links go over group boundaries and are thus *global*, *inter-group* links. There are no strict rules for the *intra-group* topology. For groups composed of a small number of routers

(e.g. $n < 20$), an all-to-all topology is practical. For larger groups, a 2D-Flattened Butterfly (2D-FB) [33] is generally retained, meaning that routers form a 2D lattice and are fully-connected in each row and column. As for the upper tier, the groups are usually fully connected. That is, the global links are disposed such that any pair of groups are connected by k links. Each group thus has $k(G-1)$ outgoing and incoming links. To allow a balanced distribution of the links over the n routers within a group, $k(G-1)$ should be a multiple of n. In the simplest ("canonical" Dragonfly), $k = 1$ and $n = G - 1$, and each router within a group is a gateway to one particular remote group. The total number of routers is then $S = G(G-1)$, while the number of global links is the same. If an all-to-all intra-group topology is used, there are $G(G-1)(G-2)$ intra-group links; for a 2D-FB intra-group topology, the number is $2G(G-1)(\sqrt{G-1}-1)$. The number of *intra-group* links exceeds the number of *inter-group* links by a factor of $O(G)$ for all-to-all or $O(\sqrt{G})$ for 2D-FB.

A key feature of Dragonfly topologies is their strictly bounded *diameter*. In the worst case, a message towards another group must i) route to the corresponding gateway within the source group, ii) traverse the global link, and iii) route to the destination router within the destination group. The diameter D of Dragonfly thus equals $1 + 2D_{intra}$, where D_{intra} is the diameter of the intra-group topology. Hence, if the intra-group network is all-to-all, then $D = 3$; if the intra-group network is 2D-FB, then $D = 5$.

2.3 Traffic Characterization on Dragonfly

Although Dragonfly has many advantages as mentioned above, its bandwidth allocation at the inter-group level can consistently mismatch the executed traffic pattern. In Fig. 2.2(b-f), *group-to-group* (G2G) traffic matrices are collected from a set of representative HPC applications. The figures show that G2G traffic matrices are far from

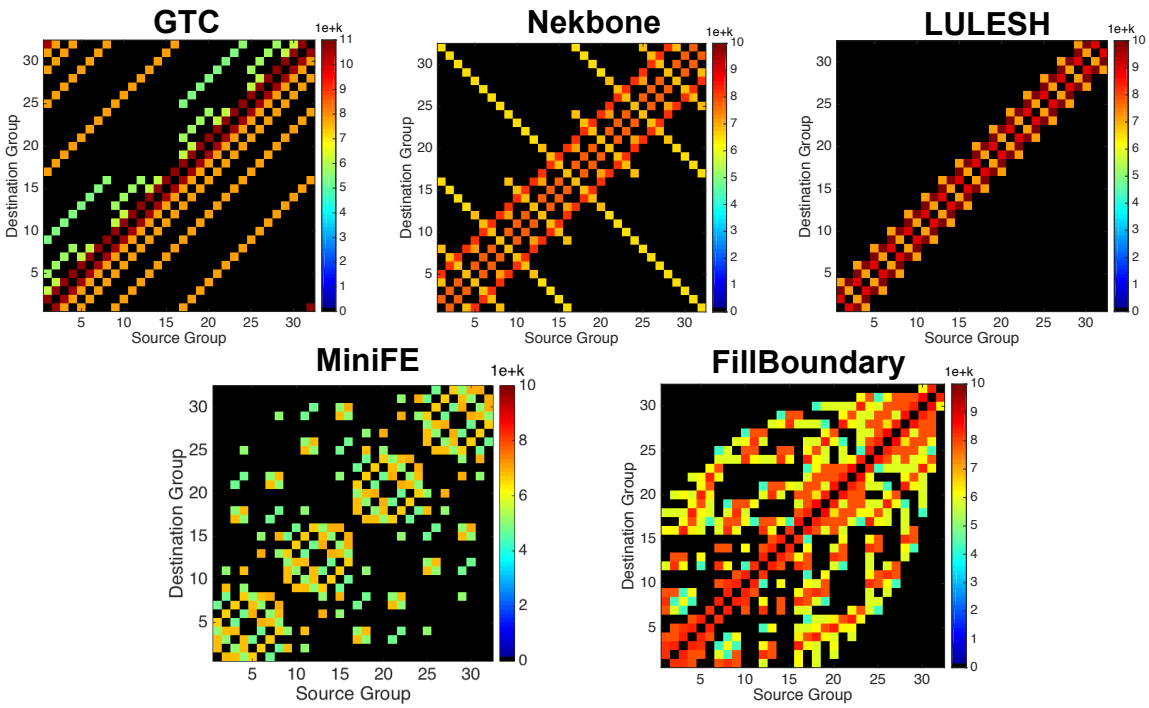


Figure 2.2: ***Group-to-group*** traffic matrices of applications run on a 32-group Dragonfly.

being uniform and many pairs of groups do not communicate at all (black regions). Also apparent: a large portion of traffic concentrates on just a few source-destination pairs, e.g. +1/-1 neigbors.

Among these evaluated applications, GTC (Gyrokinetic Toroidal Code) [34], [35] is a particle-in-cell simulation for plasma microturbulence and features neighbor-intensive point-to-point communications. The most distinguishing feature of GTC is (cyclic-shifted) diagonal lines with the heaviest traffic on the +1/-1 neighbor groups, which is also referred to as the "worst-case" adversarial traffic pattern for Dragonfly. Nekbone represents the main computational kernel of Nek5000 [36], an application for large eddy simulations and direct numerical turbulence simulations based on spectral element methods. The Nekbone kernel studied in this paper solves a Poisson equation using conjugate gradient iteration with no preconditioner. In addition to intensive traffic along the diagonal, Nekbone also has a complement (anti-diagonal) pattern

due to collectives. LULESH [37] is a proxy-app representing typical hydrocodes. In this paper, LULESH partitions the problem into a collection of volumetric elements defined by a 3D mesh and features 3D neighbor based communications. LULESH has a thick ribbon-shape pattern along the diagonal. MiniFE is a Finite Element mini-app from the Mantevo suite [38]. It assembles and solves a sparse linear system from a conduction equation using a conjugate-gradient algorithm. Its traffic pattern is characterized by an absence of diagonal lines. Lastly, FillBoundary is a simple code designed to profile communication patterns associated with ghost cell exchanges. We simulated the halo update of a BoxLib-based [39] production partial differential equation (PDE) solver. FillBoundary has an irregular traffic matrix and induces traffic between about half of the group pairs (with the heaviest traffic concentrated on a few pairs).

For many applications like the above ones, the traffic pattern is known a-priori. This information can be used for configuring the interconnect at, for example, job launch time. For other applications that are more dynamic, an iterative aspect usually exists in traffic patterns, and the runtime middleware should easily be able to characterize network traffic based on the first few iterations and then perform the reconfiguration.

2.4 Flexfly: A Reconfigurable Dragonfly

We propose Flexfly, a silicon photonics (SiPh) based architecture that enables flexible allocation of Dragonfly global links. In Flexfly, the global links initially composing the all-to-all topology can be "stolen" from their original destination groups, and reassigned to traffic-intensive ones. By trading the global links in this way, Flexfly creates additional *direct* bandwidth for intensively-communicating group pairs, matching the topology to the application traffic. Flexfly achieves such reconfigurability through

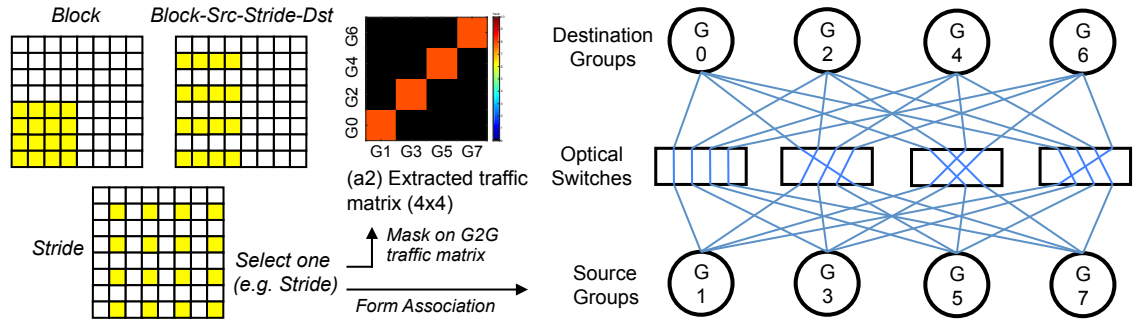


Figure 2.3: Construction flow of the Flexfly architecture.

transparent SiPh circuit switching. Unlike previous optical switching solutions which rely on large port counts (equal to the number of nodes, racks, routers [40] or groups [41]), Flexfly is designed in a way to support the use of low-radix optical switches, independently of the global system scale. Such low radices enable massive low-cost fabrication through current SiPh technologies and realizable SiPh switch designs. As compared to adaptive routing, Flexfly creates multiple minimal paths and mitigates the need for indirect routing.

In this section, we describe how Dragonfly groups, in topologies of any size, can be interconnected with arbitrary-radix SiPh switches and how the link re-allocation is performed.

A construction flow for the Flexfly architecture is shown in Fig. 2.4. A *group association mask* is used to extract a sub traffic matrix from the *group-to-group* traffic matrix. Examples of mask for $G = 8, r = 4$ (r being the optical switch radix), and the extracted traffic matrix (ETM) corresponding to the *Stride* mask, are shown in Fig. 2.4. Also corresponding to the mask is a set of source and destination groups, which we call an *Association*. These groups are originally fully-connected in Dragonfly. In Flexfly, r optical switches each of radix r are inserted in the middle of the r^2 optical connections owned by each *Association*, as shown in Fig. 2.4(3). An *Association* is thus composed of r source groups and r destination groups. The source

and destination groups do not have to be the same. The ETM size is also $r \times r$.

In Fig. 2.4(3), the state of the switches ensures an all-to-all connectivity among the *Association*. This corresponds to the "original" Dragonfly connectivity. In presence of non-uniform traffic, this all-to-all connectivity can be reconfigured to re-allocate global bandwidth to traffic intensive source-destination group pairs. Considering, for example, the ETM in Fig. 2.42, switches can be configured such that all four links originated at $G1$, $G3$, $G5$ and $G7$ are destined to $G0$, $G2$, $G4$ and $G6$, respectively. In that case, the bandwidth *concentration* within the *Association* is maximal ($r = 4$-fold). It should be noted that associating the groups will not lead to network isolations, since a group can appear in multiple *Associations*.

Group Association Strategies

Multiple strategies exist when creating *Associations* contained in a Dragonfly, corresponding to different group association masks. Each strategy results in different sets of ETMs, which may exhibit variable potential for link *concentration*. In this set of possibilities, we are interested in the strategies that a) distribute traffic evenly across associations, b) provide flexibility that can be exploited by major HPC traffic patterns, and c) allow easy cabling across the supercomputer cabinets. With respect to b), remark that in Fig. 2.4(2), all the "dark" (unused) links can be re-allocated to the four active group pairs. We therefore look for strategies favoring such situations. Here we propose three major strategies for comparison: *Stride* association, *Block* association, and *Block-Source-Stride-Destination (BSSD)* association. Their respective masks are shown in Fig. 2.4(1). We will evaluate the "quality of association" of these strategies based on the applications shown in Section 2.

The *Stride* strategy selects groups with a stride of $k = G/r$ in both source and destination dimensions. This strategies generally allows the distribution of heavy traffic spots into different associations. The stride k, however, should avoid coupling

Table 2.1: Quality of association with $G = 32, r = 8$

	std of number of nonzeros (σ_{nz})	std of traffic percentage (σ_{pct})	max number of nonzeros in a line (n_{NZL})
Block	13.12	6.72	2.86
Stride	10.23	6.09	**2.56**
Block-Src-Stride-Dst	**7.14**	**0.93**	4.38

with the distance between two intensive destinations. By contrast, the *Block* strategy selects r continuous groups in both source and destination dimensions. *Block* relies on an assumption that neighboring sources tend to "favor" the same configuration when connecting to destinations – for example, in GTC each source group prefers connections to their +1 neighbors. Such "harmony" allows for skewing the original source-destination (s-d) connections within a switch by the same amount, thus simplifying the determination of the switch state. The *Block* strategy, however, fails to efficiently distribute the traffic across *Associations* in presence of traffic concentrated along the diagonal (for neighbors-intensive communication patterns as in LULESH). Hence, it results in higher chance of creating fully occupied ETMs on one side, and purely dark ETM's on the other side. The third strategy, *BSSD*, is a mixture of the previous two for comparison purpose. Fig. 2.4 show how the different strategies generate distinct ETMs.

Evaluation of Association Strategies

We evaluate which association strategy provides flexibility in the most relevant way for HPC applications. We use three metrics from the ETM pool as a measure of "quality of association": 1) standard deviation of the number of nonzeros across ETM's (σ_{nz}), 2) standard deviation of the traffic percentage across ETM's (σ_{pct}) and 3) average of maximum number of nonzeros in a line (row or column) of an ETM

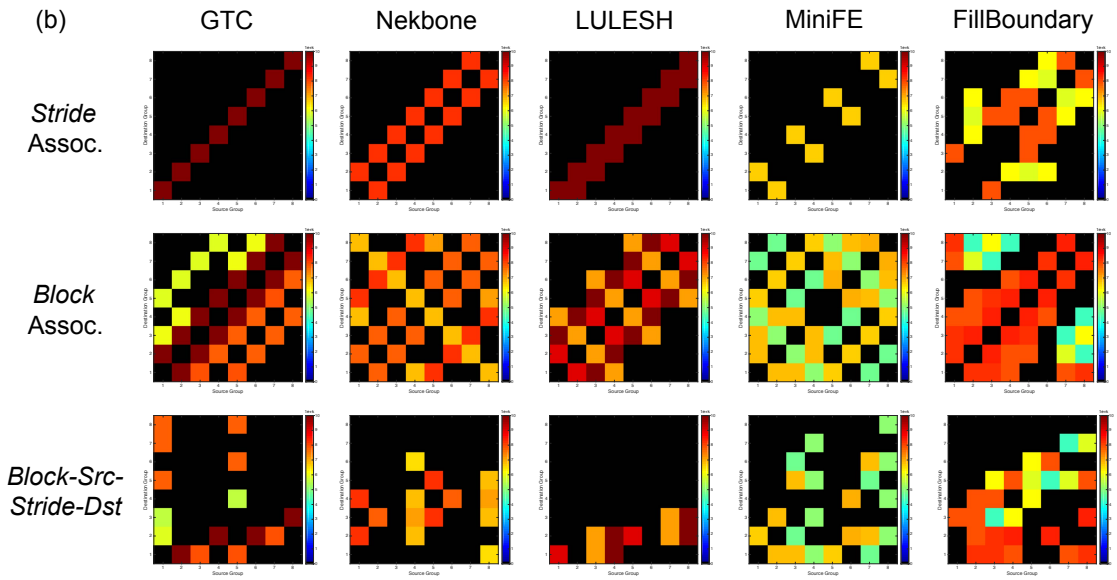


Figure 2.4: Representative ETM's from applications using the three association strategies; here switch radix $r = 8$, hence ETM size is 8×8.

(n_{NZL}). The first two metrics are intended to indicate how evenly the G2G traffic is distributed across different ETMs/associations. The third metric n_{NZL}, by contrast, indicates the maximum communication degree of a source or destination group within an association, i.e. how many links are necessary for a source or destination to cover its non-zero traffic. The higher this value, the lower the flexibility, as "necessary" links cannot be stolen (described later in Sec. 2.5).

Since it is impossible to find a single association strategy that works best for every application, we use a statistical approach to evaluate the above strategies based on the application set described in Sec. 2.3. The statistics are hence a pool of ETM's extracted from the G2G traffic matrices of those applications, using respectively the three association strategies.

The evaluation result is shown in Table 2.1. *BSSD* achieves the smallest σ_{nz} and σ_{pct}, i.e. it distributes traffic most evenly among the ETM's. However, for n_{NZL}, the performance of *BSSD* is the worst; indeed, the n_{NZL} of *BSSD* is much higher than that resulting from the *Stride* strategy or *Block* strategy. This means that *BSSD*

requires more "necessary" switching modes to cover the nonzeros in ETM, reducing the amount of free links (flexibility) useable for intensive traffic. The *Stride* strategy, in comparison, achieves the lowest n_{NZL} and hence the highest flexibility. Therefore, we use it in later application-based simulations.

2.5 Link Stealing Algorithm

After forming the associations, Flexfly applies a link stealing algorithm to each of the associations for allocating global bandwidth based on the ETM (note that in practice, associations are formed once for all at design time, whereas to link stealing algorithm is applied for each new application). The task of the link stealing algorithm is to find a set of switch states for the r optical switches owned by an association. Since each association is independent, the link stealing algorithm can be applied in parallel. Furthermore, limited radixes r ensures that each problem remains small in scale.

A decomposition example for $r = 8$ is shown in Fig. 2.5. Rather than using the 8 switches for forming an all-to-all topology, Flexfly picks permutation matrices (each corresponding to a switch state) to cover the heavy traffic, resulting in a 2 to 3 times bandwidth increase for each nonzero s-d pair.

Here we formulate the link stealing problem with a min-max rule: given a non-negative $r \times r$ ETM $T = (t_{ij})$, find a set of r permutation matrices $\{P_1, P_2, ..., P_r\}$, such that

$$\max_{i,j} \frac{T_{ij}}{(\sum_{k=1}^{r} P_k)_{ij}} \rightarrow \min$$

It is already known that a *decomposition problem* with a min-max optimization rule is NP-hard [42]. Here we give a heuristic algorithm yielding an efficient decomposition.

In the algorithm, a nonzero entry T will be given a weight close to infinity if that entry has not been allocated a link. In this way, the algorithm makes sure that

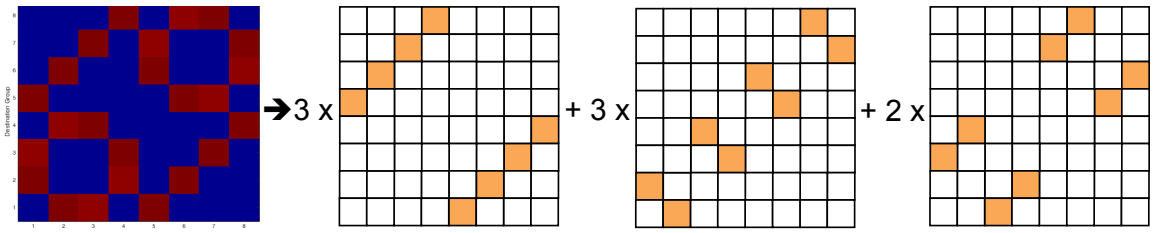


Figure 2.5: Decomposition of an example ETM into three switching modes ($r = 8$).

Algorithm 1 Min-Max Decomposition

Require: $T_{r\times r}$

1: $inf \leftarrow$ a large number
2: $D_{r\times r} \leftarrow 0$
3: Define a bipartite graph $G = (V, E)$, where $(i, j) \in E$ has weight:

$$w_{ij} = \begin{cases} T_{ij}/(D_{ij}+1), & \text{if } D_{ij} \geq 1 \\ T_{ij} \cdot inf, & \text{otherwise} \end{cases}$$

4: Find a maximum weight matching P_k in G
5: $D \leftarrow D + P_k$
6: If $k = r$, STOP; else go to 3.

an uncovered nonzero entry will have higher priority in getting picked than those already covered, even if the latter has much more traffic to send. Due to the nature of permutation matrices, whereby the line sum of each row and column is always 1, the final allocation matrix D will have a line sum of r for each row and column. This means that the number of transmitters needed by a source group for that association, i.e. row sum of D, does not exceed r; and similarly, the number of receivers needed per destination group, i.e. column sum of D, does not exceed r either. Each permutation matrix in $\{P_1, ..., P_r\}$, which corresponds to an input-output mapping, can be carried out by one optical switch in that association.

2.6 Routing in Flexfly

In this section, we discuss the routing schemes for Flexfly. Since in Flexfly there can be multiple *direct* links to the same destination group, the routing scheme must be

able to efficiently utilize these links.

Minimal Valiant Routing

In Flexfly, to load-balance the traffic across multiple minimal paths, the Valiant algorithm approach can be used to randomizes the path selection. This algorithm, which we call Min-Val, takes three steps for a source router R_s in group G_s:

1. If $G_s \neq G_d$, R_s randomly selects a router R_a in G_s that has a global channel to G_d (R_a can be R_s itself) and route within G_s from R_s to R_a;
2. traverse the global channel from R_a to reach router R_b in G_d;
3. If $R_b \neq R_d$, route within G_d from R_b to R_d.

Different from the Valiant routing in Dragonfly, MIN-VAL in Flexfly does not involve intermediate groups, i.e., the path selection is always among minimal paths. This ensures exactly one global hop for all cross-group packets, enabling more than 50% throughput in global level.

Minimal UGAL Routing

The Min-Val algorithm works well in load-balancing traffic evenly across multiple minimal paths. However, it does not consider different congestion states of different paths. Minimal UGAL (Min-UGAL) overcomes this drawback by using queue lengths to estimate network delay and choose the path with minimal delay. Similar to Min-Val, Min-UGAL does not involves intermediate group either. For the UGAL-L version [43], which uses *local* queue information at the current router, the algorithm is described as follows: if $q_m H_m \leq q_{nm} H_{nm}$, route minimally; else route non-minimally. Here, the minimal and non-minimal paths both take one global hop, and differ only in the number of hops within source/destination groups.

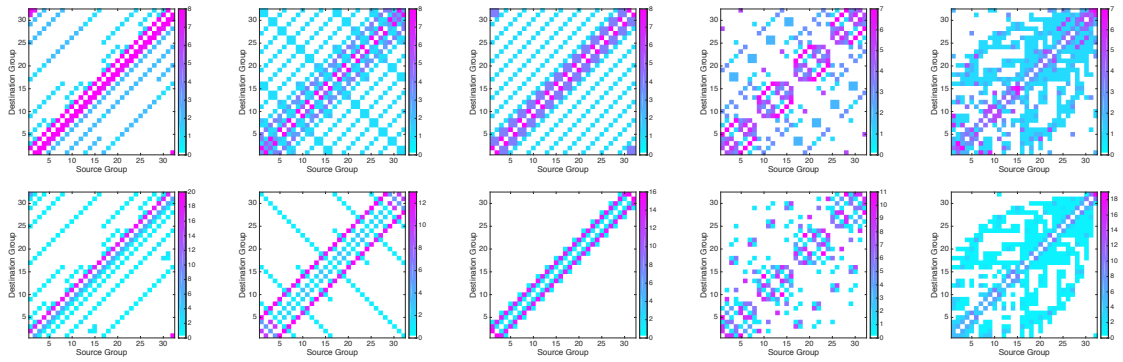


Figure 2.6: Global link allocation with switch radix $r = 8$ (upper) and $r = 32$ (lower). Note that the color scales are differnet across sub-figures.

2.7 Simulation

In this section, we use the HPC applications from Sec. 2.3 to evaluate Flexfly performance relative to minimal and UGAL routing for Dragonfly. The application configurations are shown in Table. 2.2. In the simulation, we assume the Dragonfly network has an all-to-all topology in the upper level and a 2D-flattened Butterfly in the lower level. We assume 3 GB/s for both intra- and inter- group links and an injection bandwidth of 8 GB/s. Switch hop latencies are 40 ns. The applications are simulated either in form of mini-apps (GTC, MiniFE and LULESH) or by replaying timestamped MPI traces using the DUMPI tracing tool (Nekbone and FillBoundary). Both forms are simulated through the SST simulator (SST/macro) [44]. Trace replay uses a coarse-grained packet simulator that models routing and control-flow at the level of packets [45]. Traces were collected in a MPI-only fashion on CPU-only machines. Here we examine "optimistic" scenarios for on-node parallelism, assuming a compute time speedup factor of 1/1152 leveraging abundant on-node parallelism (multi-threading with accelerators) in future applications. The traces were obtained from a NERSC web portal on characterization of DoE mini-apps [46].

Fig. 2.6 shows the global link allocation generated by the link stealing algorithm

for different applications. With a modest radix $r = 8$, the allocation matrices already become very similar to the G2G traffic matrices in Sec. 2.3. As r increases, more "unused" links can be stolen for intensive traffic pairs. When $r = G = 32$ (full reconfigurability) every unused link can be stolen for traffic and the allocation matrix pattern matches the G2G traffic matrix.

Table 2.2: Application size and parameters

Application	*Size*	*X Y G*	*Concentration*	*Parameter*
GTC	1024	4 8 32	1	*npartdom*=32
Nekbone	1024	4 8 32	1	DUMPI trace
LULESH	4096	4 8 32	4	$nx = 256$
MiniFE	1024	4 8 32	1	$nx = 256$
FillBoundary	1000	4 8 32	1	DUMPI trace

Application Speedup

Fig. 2.7 (top chart) shows the speedup of Flexfly (Ffly) with different switch radices as compared to Dragonfly (Dfly). Here we use minimal, Valiant and UGAL routing for Dragonfly, and minimal, Min-Val and Min-UGAL routing for Flexfly. For comparison purpose, the speed of Dfly-Min is normalized to 1.0x. When minimal routing is used, Flexfly achieves at most 7.1x speedup over Dfly-Min and 1.8x speedup over Dfly-UGAL in the case of GTC (the most neigbor intensive application). When Min-UGAL routing is used, these two speedup numbers are 8x and 2x. In the case of LULESH, Flexfly with $r = 32$ and any routing algorithm achieves 5x speedup over Dfly-Min and 1.7x over Dfly-UGAL. For MiniFE and FillBoundary, Flexfly also shows better performance than both Dfly-Min and Dfly-UGAL, but with a modest speedup (< 1.5x). This limited speedup is due to the less sparse traffic matrix of these two application, which results in reduced stealing possibility. Another reason for MiniFE is that its communication is not very intensive, rendering the performance

more compute-bounded. Within Flexfly, the performance also improves in general as the switch radix r increases (due to more flexibility).

From the above results, one can see that the sparser the traffic matrix is, the more free links Flexfly can steal, and hence the better the performance. In light of this, a more aggressive approach would be to sparsify the traffic matrix *before* applying the link stealing algorithm. This is done by eliminating the light traffic dots that are not "worth" allocating one full link of bandwidth. A possible approach is to use two links that would be allocated for two heavy traffic dots to provide a two-hop path for the light traffic dot (similar to indirect routing). In this way, the link that would have been allocated to the light traffic dot can be freed for servicing heavy traffic. Although indirect paths are used, it is used for the light traffic, creating little impact on the network load. By contrast, in UGAL or other indirect routing approaches for Dragonfly, the indirect paths are often used by heavy traffic, which can quickly consume the capacity of the network.

Hop Counts and Latency

A closer look of the architecture performance can be found in the middle and bottom charts of Fig. 2.7. On the hop count experienced by global messages, Valiant and UGAL routing result in much more hops than minimal routing in Dragonfly due to the traversal of intermediate groups (about 40-50% increase). By contrast, Flexfly with any switch radix and any of Min, Min-Val and Min-UGAL routing, requires even less hops than Dfly-Min. There are two reasons for that: (1) the three Flexfly routing algorithms all rely on direct global links rather than intermediate groups, and (2) the increased distribution of wanted global links in the group plane reduces the distance to/from cross-group gateways. Specifically, Ffly-Min halves the hop count compared to Dfly-UGAL in cases of GTC and LULESH. With respect to the latency of global messages, the same trend is observed, as a combined result of reduced hop counts

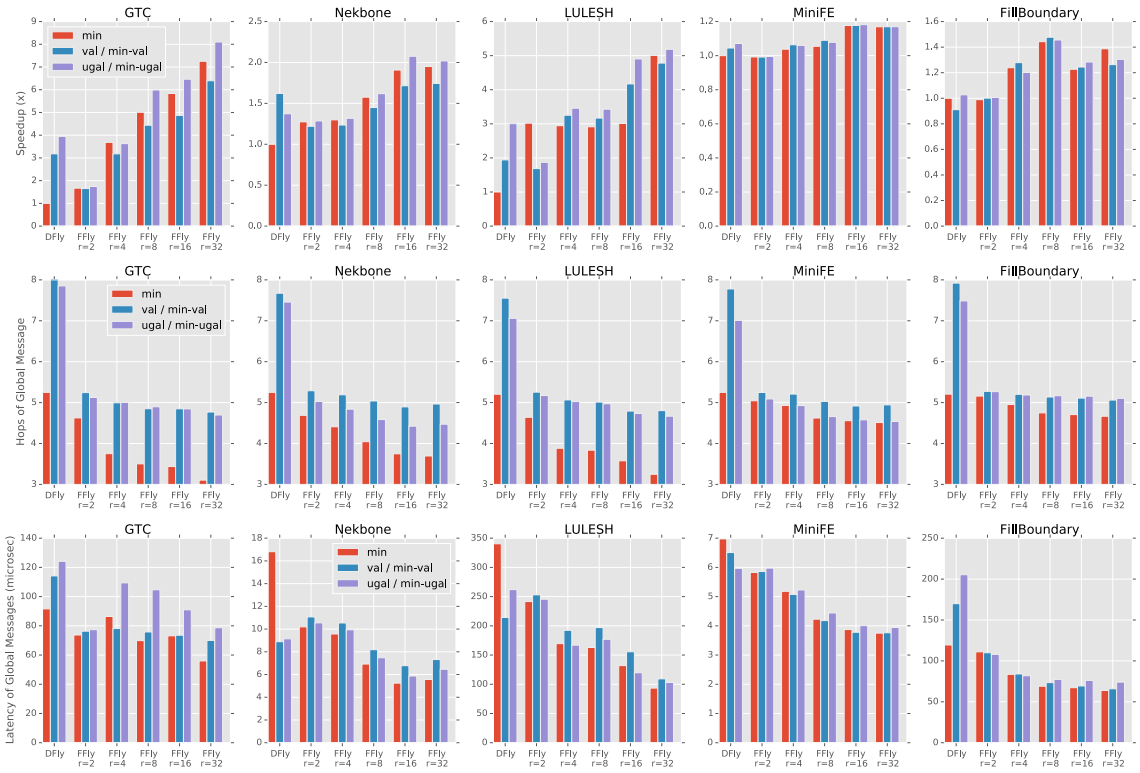
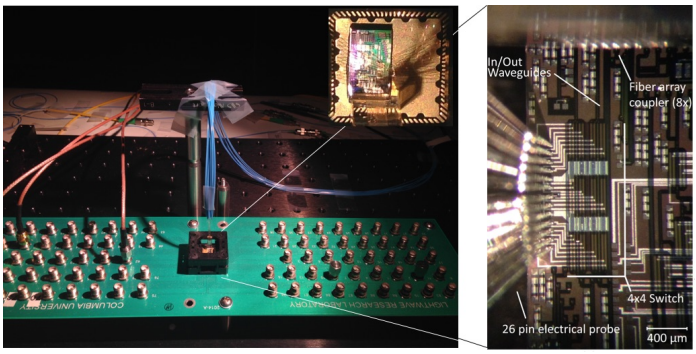


Figure 2.7: (Top) Speedup of Flexfly ($r = 2, 4, 8, 16, 32$) with various routing mechanisms over Dragonfly with minimal routing (normalized to 1.0x). (Middle and bottom) Hops and latencies of cross-group messages. Val and UGAL are used on Dragonfly. Min-Val and Min-UGAL are used on Flexfly.

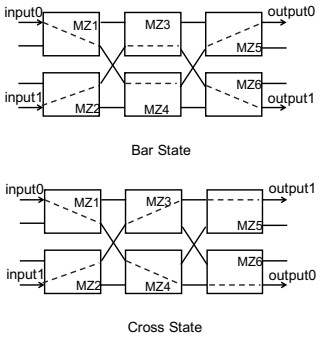
and increased cross-group bandwidth. In cases of GTC, LULESH and FillBoundary, Ffly-Min ($r = 32$) achieves 2x, 2.5x and 3.2x reduction in global message latency over Dfly-UGAL. The difference in message latencies across application is mainly due to different message sizes.

2.8 Experimental Demonstration

We built a 32-node Flexfly prototype using a 2-by-2 SiPh switch to evaluate the implementation complexity and performance improvements compared to Dragonfly.



(a)



(b)

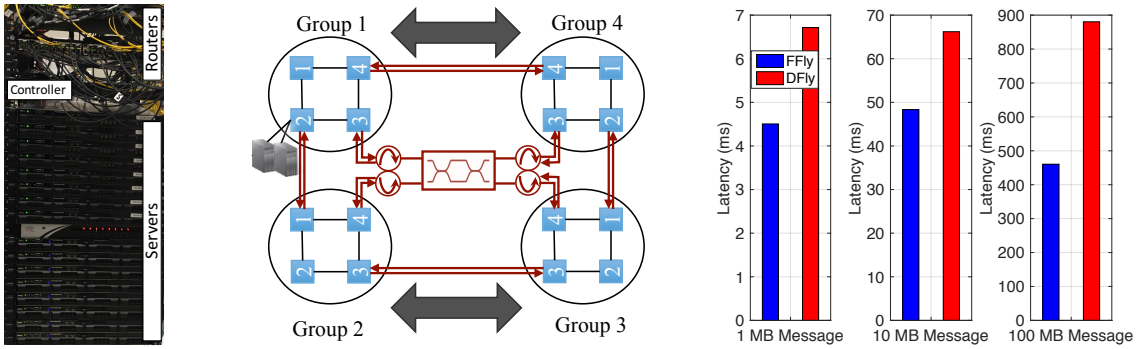
Figure 2.8: (a) Socketed switch chip mounted on a PCB; (b) Bar and cross configurations for ports 1 and 4 in the six-MZI Benes switch.

Silicon Photonic Switch

The SiPh switch (Figure 2.8a) embedded in the HPC testbed was manufactured through the OpSIS [47] foundry. The chip was mounted in a plastic leaded chip carrier socket that was soldered on a printed circuit board (PCB). The SMA input/output from the board is connected to the bias control units (Fig. 2.8a). The switch is a rearrangeably non-blocking 4×4 Beneš topology [48], comprised of three stages and six 2×2 Mach-Zehnder interferometer (MZI) elements. Further characterization of the device is reported in [49]. For this particular demonstration, the switch is biased to either a *bar* or a *cross* state, turning it as a 2×2 as illustrated in Fig. 2.8b. Dynamic extinction ratios are maximally 15 dB and minimally 7 dB in the *bar* state, while in the *cross* state it ranges from 19 to 24 dB. A Data Acquisition (DAQ) unit was employed to generate the voltages applied to the device. A C program controls the DAQ and is directly callable by a *controller server*.

Flexfly Prototype

The 32-node prototype comprises 16 routers divided in four groups. This structure, shown in Fig. 2.9b, corresponds to a $G = 5$ Dragonfly whose fifth group has been



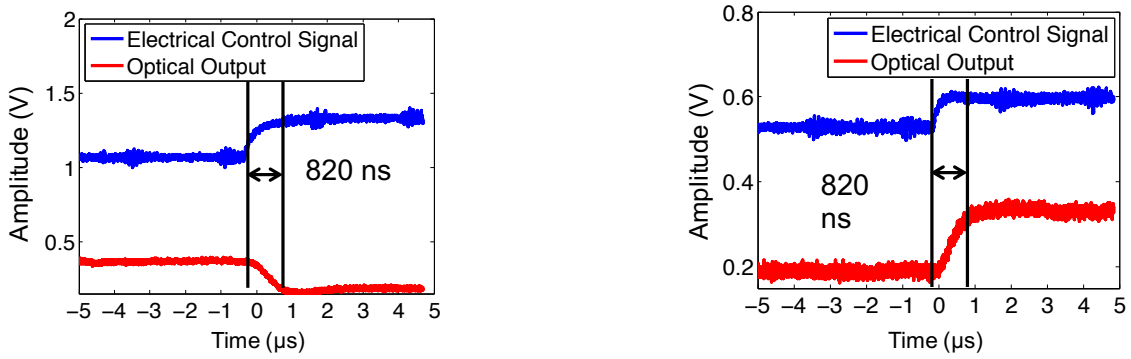
(a) (b) (c)

Figure 2.9: (a) 32-node Flexfly prototype and (b) corresponding topology diagram (2×2×4); (c) Comparing round trip time over 8 pairs of servers (each pair in different groups), in Dragonfly or Flexfly configurations.

removed (due to Ethernet switch limited port count). Two servers are connected to each router ($C = 2$), as shown in Fig. 2.9b and Fig. 2.15. Each server is equipped with a dual-core Intel Xeon processor, 10 GB RAM, and a 10Gbps Network Interface Card (NIC). Routers are 10G OpenFlow [50] Ethernet switches. The intra group links are 10G Direct-Attached Cables. For inter-group links, 10G SFP+ transceivers are connected to a 3-meter long Single-Mode Fiber (SMF). One of the servers (the *controller server*) sends reconfiguration commands to the DAQ for switching between the *bar* and *cross* states and to the routers for updating the flow table rules.

Experimental Results

We began by measuring the switching time of the silicon photonic switch, controlled by the DAQ. Figures 2.10a and 2.10b show the switching time of the *bar* and *cross* states. We were able to achieve 820 ns switching time using a DAQ supporting mega-sample per second analog output. Note that faster switching times could be reached using an RF driver rated for GHz range instead (the intrinsic switching time of the MZI-based switch with P-N junctions being in nanoseconds-range [49]). This measurement shows that switch reconfigurations can be applied rapidly.

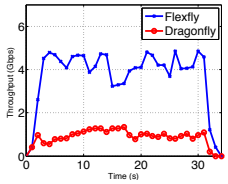


(a) (b)

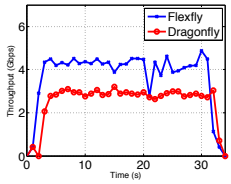
Figure 2.10: Silicon photonic switch setup time for (a) *bar* and (b) *cross* states.

By injecting traffic using the Hpcbench MPI benchmark [51], we measured the message round-trip delay between 8 pairs of servers located in two groups. For the traffic pattern, we had each server in G1 communicating with a corresponding server in G4. In the *cross* state, which corresponds to the original Dragonfly topology, there is one global link between the two groups. Setting the switch in the *bar* state grants an additional link to this G1–G4 group pair. We measured the message round-trip delay between each pair of servers using messages of 1MB, 10MB and 100MB. The experiment was averaged over 10 runs. Fig. 2.9c shows the average latency over the 8 pairs of servers. In the Flexfly-adapted configuration, the round trip delay reduces by 33%, 35% and 47% compared to Dragonfly, for 1, 10, and 100MB messages respectively.

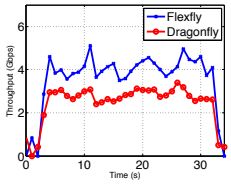
We then generated four point-to-point traffic flows between routers 3 and 4 of Group 1 and Group 4, and four other flows between routers 3 and 4 of Groups 2 and 3 (as indicated by the gray arrows in Fig. 2.9b). The routing scheme utilizes minimal routing, i.e., the direct global link between the groups are used for inter-group transmission. The traffic is generated by iperf [52], a network measurement tool. We measured the throughput between servers. In Dragonfly configuration (*cross* state of the SiPh switch), there is an optical global link between each two groups. In this



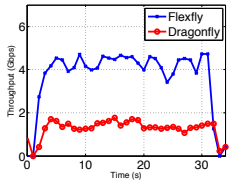
(a) (G1, R3, S2) to (G4, R3, S2)



(b) (G1, R4, S1) to (G4, R4, S1)

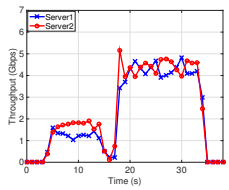


(c) (G2, R3, S1) to (G3, R3, S1)

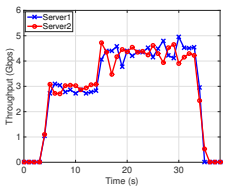


(d) (G2, R4, S2) to (G3, R3, S2)

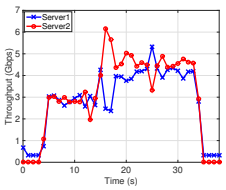
Figure 2.11: (a-d) Measured throughput of servers in a data transfer between G1 to G4 and G2 to G3. G: group, R: router, S: server. Throughput is higher in the Flexfly-adapted configuration (*bar* state) as two global links are awarded to each of the above group pairs.



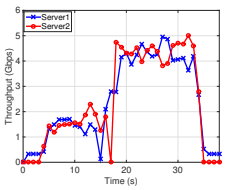
(a) Group 1, Router 3



(b) Group 1, Router 4



(c) Group 2, Router 3



(d) Group 2, Router 4

Figure 2.12: Demonstration of on-the-fly reconfiguration across the Flexfly prototype. Reconfiguration is performed at second 15 approximately and one extra global link is provided between the two groups.

case, all G1–G4 traffic is routed over the global link attached to routers 4. Similarly, all G2–G3 traffic goes over the global link attached to routers 3. That is, these two global links are each shared by four flows. In contrast, the global links between G1–G3 and G2–G4 are unused. By configuring the switch from *cross* to *bar* (Flexfly), these unused global links are allocated to the G1–G4 and G2–G3 pairs, resulting in superior throughput improvement (about 2x as shown in Fig. 2.11). Note that in the Dragonfly configuration, flows that have to take extra intra-group hops (Figs. 2.11a and 2.11d) retains a lower share of the global bandwidth. These extra hops, which increases the round-trip time, can potentially affect the flow control, resulting in lowered bandwidth. In contrast, in the Flexfly configuration, all flows can be routed over minimal hops, leading to a fairer bandwidth access in that particular case.

Finally, we performed on-the-fly reconfiguration on the Flexfly prototype. The

setup is the same as the previous experiment except that the transmission starts in the cross configuration (i.e. Dragonfly) and is then switched to bar (i.e. Flexfly). This requires i) the *controller server* to send a command to the DAQ to modify the driving voltages, and ii) updating the flow tables of the routers to add the new route enabled by the extra link. To improve flow update performance, we avoid flow deletion and simply add new rules with higher priority. Previous rules are automatically deleted after an idle time. The script uses OpenVSwitch [53] to update the flows. In our implementation flow rules are added sequentially. This process could, however, be expedited by leveraging a standard controller such as OpenDayLight (ODL) [54]. ODL can insert hundred flow rules in milisecond range [55]. We used clusterSSH [56] in the controller server and applied the updates on the routers and the DAQ simultaneously. Fig. 2.12 shows the results. Each plot represents two servers that are connected to the respective router/groups. Servers on G1, R3 and G2, R4 had lower initial throughput and a dip in the throughput during the reconfiguration. This is due to the change in their routing from one global link to the other and as well as sequential update of the flow rules. We expect that improving the flow rule insert speed will reduce the overall reconfiguration time.

2.9 Large-Scale Implementation and Cost

In this section we show how Flexfly could be integrated in a practical supercomputer. Refering to Cray XC40 [57], a group typically spans over one or two cabinets. We therefore assume that r groups on the same row forms a Flexfly *supergroup* , resulting in $\frac{G}{r}$ *supergroups* . A Flexfly switch blade is introduced to each *supergroup* . This blade contains all the SiPh switches corresponding to the inter-group links originating from the *supergroup* . Specifically, a Flexfly switch blade will have $G-1$ switches. Hence, with $G-1$ links per group and r groups per *supergroup* , a *supergroup* has

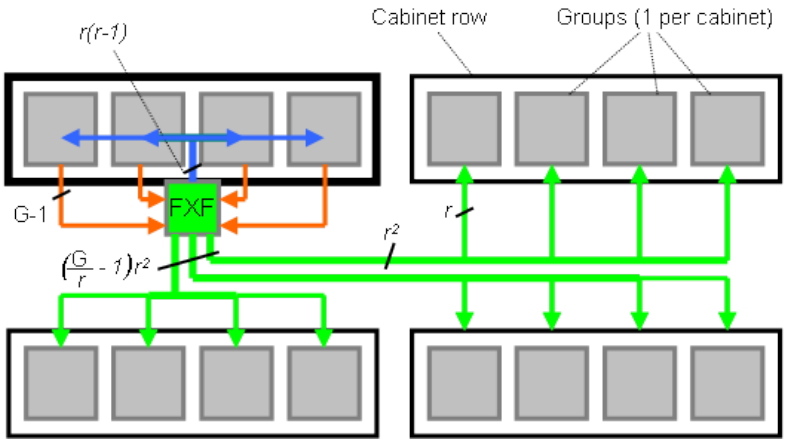


Figure 2.13: (a) Organization of cables within Flexfly. For clarity only the links originated in the highlighted set of groups are shown. These $r(G-1)$ links first connect to a Flexfly blade located in the middle of the row (in orange). From there, $r(r-1)$ connect back to the same row (in blue), while $(\frac{G}{r}-1)r^2$ links are distributed across the remaining groups (in green), (b) 32-node Flexfly prototype snapshot.

Table 2.3: Content of Flexfly racks

G	r	# of *supergroups*	# of switches	# of connectors
8	4	2	7	56
16	4	4	15	120
16	8	2	15	240
32	4	8	31	248
32	8	4	31	496

$r(G-1)$ links that fully connect to the $G-1$ switches, each with r ports. Each Flexfly switch blade also has $2r(G-1)$ fiber connectors. Table 2.3 provides the number of switches and connectors per blade for various G and r values. Due to the compatibility of SiPh with CMOS foundry, a large number of switches can be put on a single chip, reducing the packaging cost and space. Next to the SiPh chip containing the optical switches, each Flexfly switch blade also includes a controller for accepting reconfiguration commands and actuating on the switches. Since only G/r such blades are required and SiPh chips can be massively produced via CMOS fabrication, the cost and space needed for incorporating Flexfly in a Dragonfly-based supercomputer are deemed minimal.

The impact of Flexfly on cabling lengths is also minimal compared to conventional Dragonfly-based systems. Flexfly simply requires every cable to first reach the Flexfly blade of the *supergroup* . In the worst case, this results in an cable length increase by the distance of r cabinets (to the blade and back); in the best case, there is no penalty at all (if the blade is on the way of the fiber to a remote group).

2.10 Power Penalty Analysis

With optical switches inserted, optical *power penalty* is accrued as the signal experiences losses and receives crosstalk [58], [59]. In an effort to alleviate concerns about power penalty, especially as the size of switch radix scales, we perform power penalty analysis of SiPh switch designs required by our platform. We consider Mach-Zehnder interferometers (MZI) based designs [60] exclusively, MZI being far less sensitive to thermal changes than ring resonators [61].

The structure of a 2×2 MZI based switch is drawn in Fig. 2.14(a). Propagation loss and phase shift introduced by each phase shifter arm of the MZI are the main parameters used in our analysis. Applying a voltage on the arm induces a change of material refractive index. This in turn modifies light propagation speed, and the optical loss. Fig. 2.14(b) shows the dependence of the power at the output ports as a function of phase shifter loss when a single signal is presented at one input port. The Cross state shown in Fig. 2.14(c) refers to the case where no voltage is applied to the phase shifter. In that case, signals traverse the switch with minimal perturbations, they also minimally "leak" in the non-desired output port (Fig. 2.14(d)). This translates into a power penalty of 0.25 dB. In the *Thru* state, in contrast, attenuation and leakage are more important, even when applying the ideal shift, as shown in Fig. 2.14(d). The power penalty raises to 1.36 dB.

A non-blocking 4×4 Beneš switch can be constructed with the optimized 2×2

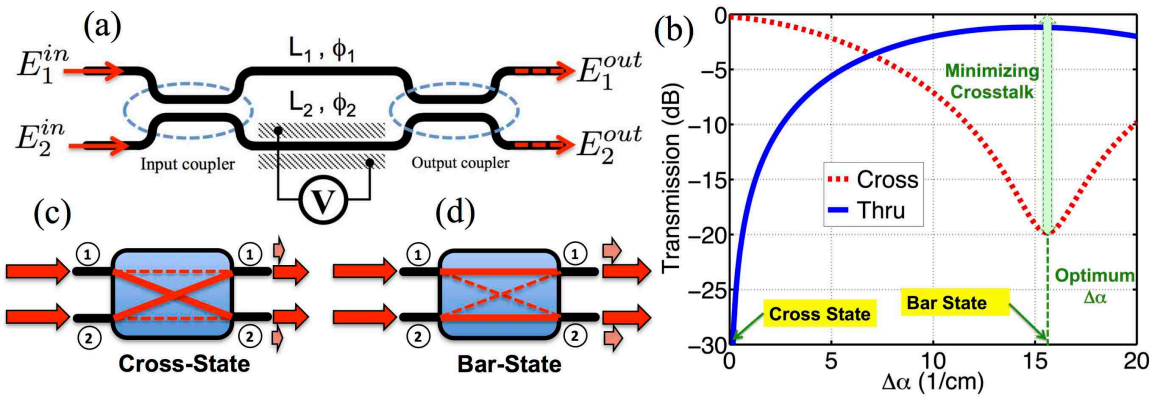
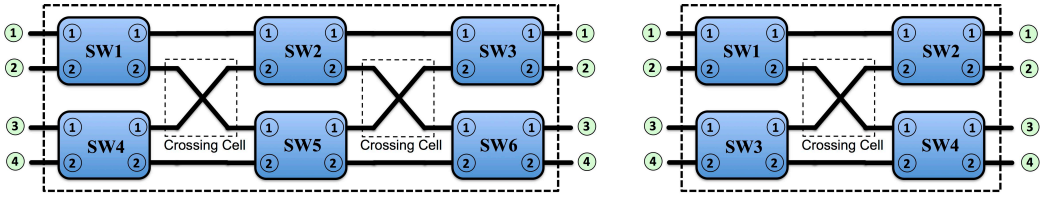


Figure 2.14: (a) Structure of a 2×2 MZI-based switch. (b) Thru and Cross transmission as a function of phase shifter loss. (c) and (d) are schematics of the switch in Cross and Bar states.



Topology	2x2 Switch			4x4 Benes			4x4 Omega		
Parameter	Value	State	Path	Value	State	Path	Value	State	Path
Worst-case Loss (dB)	1.2	B	1-1, 2-2	3.77	BBBBBB	2-2, 3-3	2.42	BBBB BBBC	2-3 3-2
Worst-case Xtalk (dB)	-19.8	B	1-2, 2-1	-17	CBBCCB BCCCBC	1-3 2-4	-20	CBCC CCBC	2-2 3-3
Penalty (dB)	1.36			4			2.6		

Figure 2.15: (top) 4×4 Beneš and Omega topologies based on optimized 2×2 switches. (bottom) Worst-case of insertion loss and crosstalk for all possible mappings from the input ports to the output ports.

switches [61]. The schematic of the 4×4 switch is presented in Fig. 2.15, which also summarizes the analysis of worst-case insertion loss and crosstalk levels for the 4×4 Beneš topology. The worst-case of insertion loss for the 4x4 Beneš topology is about 3.7 dB, the worst-case of crosstalk is about -17 dB and the worse-case overall power penalty is about 4 dB.

With power penalties $< 5dB$, the insertion of SiPh switches of radix $r = 4$ should remain transparent to optical transceivers, which generally have an optical budget in excess of 10dB. For higher radixes, or to reduce the power penalty of $r = 4$ switches, an alternative 4×4 Omega topology can be constructed from optimized 2×2 switches (schematic is shown in Fig. 2.15). Reducing one stage of the 2×2 switches will decrease the worst-case of insertion loss from 3.77 dB to 2.42, hence saving 1.35 dB of penalty. The crosstalk performance is also better for the Omega topology (improvement from -17 dB to -20 dB) due to the lower number of crossings. The worst-case penalty for the Omega structure is about 2.6 dB.

Omega switches do not provide the same flexibility as Beneš. They are, however, able to realize all circular shift permutations. Therefore, they can support all "diagonal" ETMs, which are frequent when the stride strategy is used (Fig. 2.4).

2.11 Related Work

Traffic Engineering Methods

Indirect adaptive routing (IAR) methods have been proposed to leverage underutilized, "dark" global links. Among them, Valiant (VAL) routing [20] scatters traffic hotspots by sending cross-group traffic to a randomly selected intermediate group, resulting in two global hops for most packets. Universal Globally-Adaptive Load-balanced (UGAL) routing [20] balances between minimal routing and valiant routing by comparing an estimated queuing delay of both paths. Due to the unavailibility of global queuing information, such estimation is embodied by a product of local queue lengths and hop counts (UGAL-L, assuming each hop has the same queue length as the local one). However, this often leads to misestimation and degraded performance. Jiang et al [24] explored four IAR methods to obtain non-local queuing information: credit round trip (CRT), progressive adaptive routing (PAR), piggyback routing (PB)

and reservation routing (RES). Still, due to the indirect routing nature, these IAR methods saturate before reaching 50% throughput for neighbor-group intensive traffic.

Optical Circuit Switching

Various approaches have been proposed to include optical circuit switching (OCS) in both on-chip and large-scale interconnection networks. The on-chip approaches [62]–[66], focusing on nanoscale SiPh-electronics integration, are mainly used for core-to-core packet switching. For the large-scale networks (data-centers or supercomputers), the proposed approaches can be classified into two groups. In the *parallel* group, OCS is often used for "elephant" flows for mitigating the load on the EPS counterpart, as described in Barker et al [67], Farrington et al [68], [69] and Wang et al [70]. In the *unparallel* group ([40], [41]), Shalf et al [71] proposed an architecture called HFAST that connects compute nodes to "active" packet switches. A common feature of these approaches is that they require a large-radix optical switch whose port count equals to the number of compute nodes, racks or Dragonfly groups, and has to increase as the system size scales.

Silicon photonics offers the capability of creating *photonic integrated circuits* through a CMOS-compatible process [72]. As the technology becomes mature, Silicon photonics is an attractive platform for integrated OCS because high-density devices, for example, ten of switches, can be integrated on a chip [73]. As for a single switch, however, the demonstrated port count is still at a modest level. This is because the majority of SiPh switches comprise multiple stages of small switching elements such as 2-by-2's, which, scales the optical insertion loss with the port count. To date, the largest SiPh switches of this kind are 8-by-8 [74], [75].

2.12 Summary

Most supercomputing platforms currently employ a fixed network topology which can match only a few traffic patterns. Thus, these interconnection networks may become a bottleneck for next-generation exa-flop platforms whose application may vary over a wide range. In this work, we propose the Flexfly architecture, which utilizes the circuit switching capability of Silicon photonics for a reconfigurable Dragonfly network. Our solution shows a way to increase the global-level bandwidth of Dragonfly by r fold given radix-r SiPh switches. We have analyzed 5 parallel workloads that represent a wide spectrum of scientific applications and associated communication patterns. We show that these patterns consistently mismatch the global link allocation of Dragonfly, and often resembles "worst-case" traffic scenario for Dragonfly. We show that Flexfly can solve the above problem by dynamically allocating links based on the G2G traffic demands. Numerical evaluation with the analyzed applications shows at most 7.1x and 1.8x speedup over Dragonflies that use minimal and UGAL routing, respectively. The network distance and latency for global messages are also significantly reduced. We have also built a 32-node prototype based on a SiPh switch connecting 4 groups and observed 2x improvement in cross-group throughput in a realistic SiPh-based platform. On-the-fly reconfiguration of the SiPh switch has also been demonstrated and exhibits 820 ns latency.

Chapter 3

Reconfigurable Memory Interconnect for Many-core Processors

Abstract

Off-chip memory accesses for manycore processors can incur considerable non-uniform-memory-access (NUMA) latency and points of bandwidth contention hot-spots as they traverse multiple network-on-chip (NoC) links. Data-intensive applications typically issue many concurrent memory requests from each core, creating further contention at the memory controller (MC) and across the NoC fabric. We address the NUMA issue and NoC-hotspot issue using fast-tunable silicon photonics (SiP) memory tunnel interconnects. Integrated with the processor through 2.5D/3D stacking, the SiP tunnels can transparently direct traffic from any off-chip memory to any on-chip MC, thus alleviating the above effects.

We demonstrate the operation of our proposed architecture using a tunable laser, a 4-port SiP switch (four wavelength-routed memory channels) and a 4x4 mesh NoC synthesized in FPGA. The emulated system achieves a 15-ns channel

switching time. Simulations based on a 12-core 4-memory model show that for such switching speeds the interconnect system can realize a 2x speedup for the STREAM benchmark in the hotspot scenario and a reduction of execution time for data-intensive applications such as 3D stencil and K-means clustering by 23% and 17%, respectively.

3.1 Motivation

As computational density for high-performance computing and big-data services continues to scale, performance scalability of next generation computing systems is becoming increasingly constrained by limitations in memory access, power dissipation and chip packaging. The processor-memory communication bottleneck, a major challenge in current multicore processors due to limited pin-out and power budget, presents a detrimental scaling barrier to data-intensive computing. These critical issues present two key challenges for next-generation many-core multi-memory systems: 1) What is the most cost-effective way to provide enough bandwidth for processor cores to interface with off-chip memory modules? and 2) How should we address the non-uniform memory access (NUMA) and hotspot effect as NoCs keep scaling up?

The first challenge is directly related to the fundamental bandwidth density limits of electrical packages as we approach the physical limits of information density over conventional electrical wires [76], [77], specifically:

- Pin limits and bandwidth density: Although Moore's Law continues to improve transistor density and consequent performance density, there is no "Moore's Law" for increasing the number of electrical connections for a standard chip package. Conventional Ball Grid Array (BGA) packages have reached peak density.
- Packaging costs: To overcome BGA limitations, manufacturers have moved

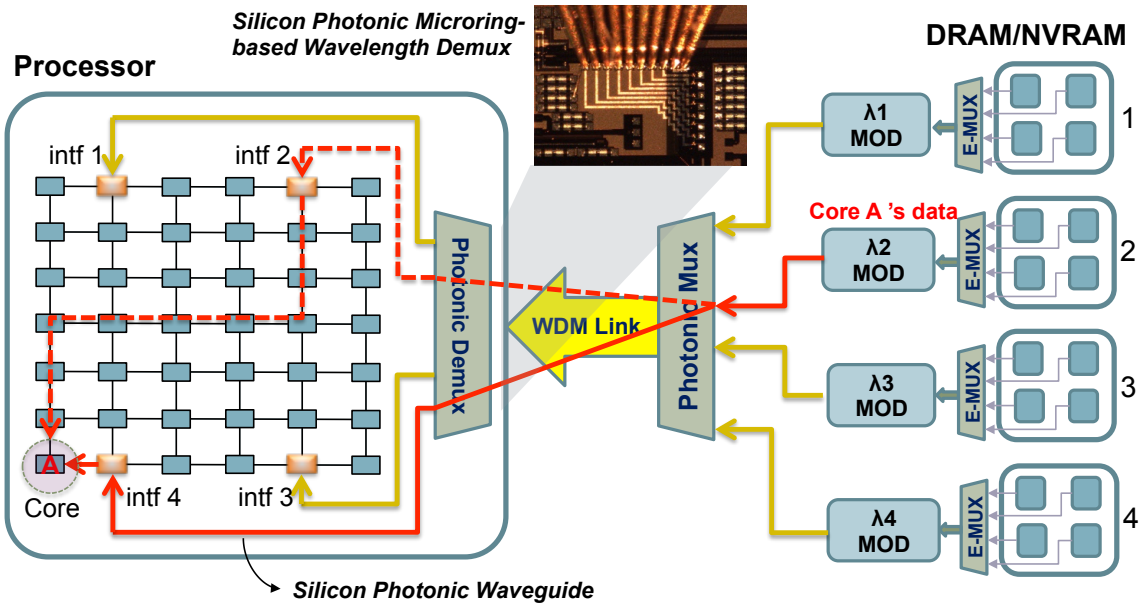


Figure 3.1: A many-core multi-memory architecture using a SiP demux as reconfigurable memory fabric. Dashed red path: 10 hops on NoC; solid red path: 1 hop only.

to memory stacking and silicon carrier technologies to provide a boost in pin density. The substantial packaging expense and limited memory capacity of these solutions create an opportunity for photonic approaches.

- Bandwidth vs. Capacity Exclusiveness: Because of area and bandwidth density limitations of current packaging technologies, memory technologies are consigned to delivering either bandwidth or capacity exclusiveness, but not both.

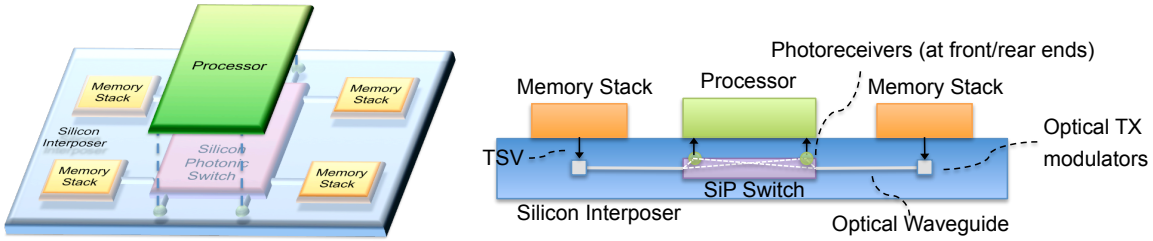
The second challenge is due to the fact that the memory interfaces in the NoC are "disseminated," and the mapping/physical layout of these memory interfaces to the off-chip memory stacks are fixed. Because the physical address space of a multicore system is globally shared, all cores transparently access all memory modules and access to remote memory modules must traverse the on-chip interconnect creating much longer latency than accessing local counterparts.

Optimizing data accesses in NoC-based many-core systems has recently attracted much research interest. Proposed strategies include careful design of cache ac-

cess/lookup strategies [78]–[80] and on-chip access localization [81], [82] to avoid very high costs of remote memory accesses. Other work [11] explores the congestion on interconnect links and in memory controllers, two factors that can dramatically hurt performance.

While these approaches have provided excellent advances in the optimization of data accesses, we propose a completely new way to simultaneously address the insufficient memory-access bandwidth, NUMA effect, and hotspot effect. Optical interconnects, especially cutting-edge silicon photonics, offers a path to a new generation of computing systems with interconnects that enable much higher bandwidth density through WDM and fast link reconfigurability. These features satisfy the requirements of emerging high-performance memory systems such as the HMC and HBM. With recent rapid progress in small-footprint CMOS-compatible Silicon Photonic (SiP) devices, a variety of optically connected memory systems [83]–[85] have been experimentally demonstrated. The cost can be further driven down when the chips go through mass production.

This chapter describes the architecture of a new photonics-enabled reconfigurable memory interconnect, our implementation of a technology demonstration platform to validate the concept, and the benchmark results from a synthetic data-intensive workload to demonstrate its performance potential. Our approach based on fast tunable SiP memory tunnels is a feasible and cost-effective solution to interconnect the many-core processor with the off-chip memory stacks.



(a) Plan View (b) Cross-section

Figure 3.2: A 2.5D stacking solution using a Silicon interposer with an embedded active Silicon photonic interconnect.

3.2 Integrated Photonic Memory Interconnect for Reconfigurability

2.5D Stacking based Implementation

The flexibility and transparency of photonic connections can provide an energy-efficient solution to NUMA and hotspot effect. The central piece in this solution is a Si-photonic interposer chip which contains active photonic components, specifically transmitters for each memory stack and the SiP switch (Fig. 3.2) . Electronic devices such as the processor and memory stacks, stacking on the interposer, connect to the photonic components through flip-chip bonding [86] or through-silicon vias (TSV) [87]–[89]. The SiP switch, capable of routing optical data without Optical-Electrical-Optical (OEO) conversions, allows the processor to communicate to all memory stacks or to selected memory stacks via any desired interfaces with no additional propagation latency.

The processor may also contain a control section to configure the photonic switch based on communication demand. This connectivity can be configured to achieve single-wavelength connection (if using wavelength demultiplexers) or multiple-wavelength connections (if using broadband spatial switches such as MZIs), depending on the bandwidth requirement of individual memory modules such as HBM and NVRAM. Control information can also be also used to adjust NoC routing rules.

Switching Implementation

In this work, we used a programmable tunable laser[90], [91] and a SiP demultiplexer to construct the optical link. Co-integration of tunable lasers with the silicon wafers has been demonstrated [92]–[94]. The 4-channel SiP demultiplexer is first tuned to the designated wavelengths and maintains wavelength stability during the experiment. The fast tunable laser is then triggered to switch among four channels in a circular order. Different wavelengths will be selectively dropped by the SiP demultiplexer to different output ports, thus reaching different MCs (Fig. 3.10c). Unlike electrical switching, SiP switching does not require per-flit latching, thus can be highly energy efficient. From the memory's perspective, the connections are transparently re-wired to a desired MC tile, bringing them closer to the requesting cores. The reason we chose to switch the tunable laser instead of the micro-rings is because the tunable laser, as shown later in section 4, can achieve WDM channel switching time of 15 ns, which is two orders of magnitude smaller than previously-shown conventional thermally-tuned experiments [95]. It is also shorter than the transaction latency of state-of-the-art memory modules (HBM and HMC, about 40 ns). Such short switching time thus enables dynamic optimization of memory-MC connections during application run time, with fine granularity.

3.3 Optimizing Core-Memory Affinity

We consider how individual CPU cores fundamentally communicate with the memory. For many-core processors, a 1D ring or a 2D mesh NoC is typically used in commercial chip designs such as Intel Knight's Corner (ring) and Knight's Landing (mesh). In these designs, the memory interface works as an endpoint on the NoC: instead of a NoC tile with a processor core, it is a NoC tile with a memory controller. The NoC therefore carries two types of traffic: core-to-core traffic (such as coherence messages)

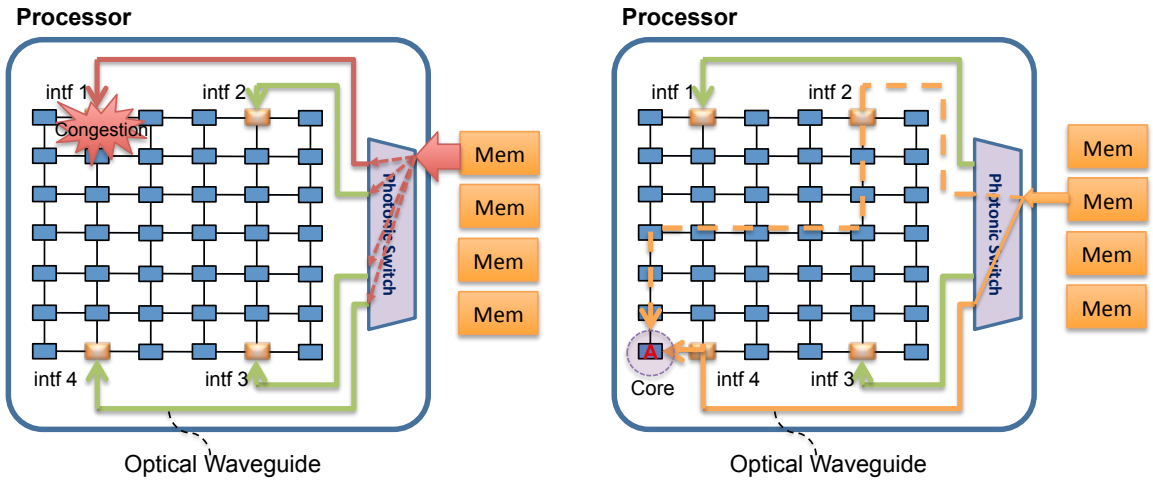


Figure 3.3: Architectural benefit of SiP switch: alleviating hotspots (left) and optimizing core-memory affinity (right).

and core-to-memory traffic, with the bandwidth shared between the two.

In an n-by-n mesh NoC, the hop distance between a core and a memory controller can vary between 1 and $2(n-1)$. When n is greater than 6, this range would span over one order of magnitude, severely limiting the scalability of the core count. In addition, when a core needs to access a remote interface, the throughput will taper as the hop distance will increase the amount of congestions encountered by the flow.

A reconfigurable SiP switch can dynamically change the injection interface of the memory data, enabling the use of an injection point near the destination core. In this way, the number of on-chip hops can be effectively reduced. For instance, in Fig. 3.3(right), the hop count decreases from 10 (dashed red, as in native connection) to one (solid red, after reconfiguration). The hop reduction can not only cut down the latency experienced by the memory traffic, but also reduce the load of on-chip links and routers. The latter effect makes additional bandwidth available for other on-chip traffic such as core-to-core messaging and cache coherence communication.

3.4 Alleviating Hotspots

While the NoC-based many-core architecture requires well-balanced workload and memory assignment where each core accesses each memory stack uniformly, real application usually fails to show this uniformity. For example, in Fig. 3.3(left), the cores on the left half may simultaneously access the memory interfaces on the right, creating a bisectional bandwidth requirement of 256 GB/s (assuming each memory stack supports a bandwidth of 128 GB/s). If the NoC link is 64-bit wide and runs at a 2.1 GHz clock, then the 7 bisection links will provide a bisection bandwidth of 117.6 GB/s, less than half of the 256 GB/s requirement. To make matters worse, the cores may also simultaneously access a single memory module, creating congestion in the NoC around that single MC tile.

For this hotspot scenario, the SiP switch can TDM shuffle the response traffic (memory to cores) through multiple injection interfaces of the chip. This unique capability of photonics thus distributes the memory traffic over different sections of the NoC, thereby alleviating the load of the congested MC tile by k times, where k is the total MC number. For efficiency of data transmission, the system may switch the memory channel with coarse time granularity, rather than on a per-packet basis. The choice of time granularity may depend on the buffer size at the on-chip memory interface. One example is to switch to the next interface when the buffer of current one fills up.

It should be noted that for the read (injection) traffic, no change to the NoC or the message header is needed, because the NoC can always route the message to correct core based on the core ID, no matter from which interface the message enters the NoC. On the other hand, for the write traffic, if the interface-memory mapping has changed from the natural mapping, some engineering of the destination tag may be needed to direct the packet to the correct interface. One lightweight approach is to apply mapping from a logical tag to a physical tag based on the switching state of

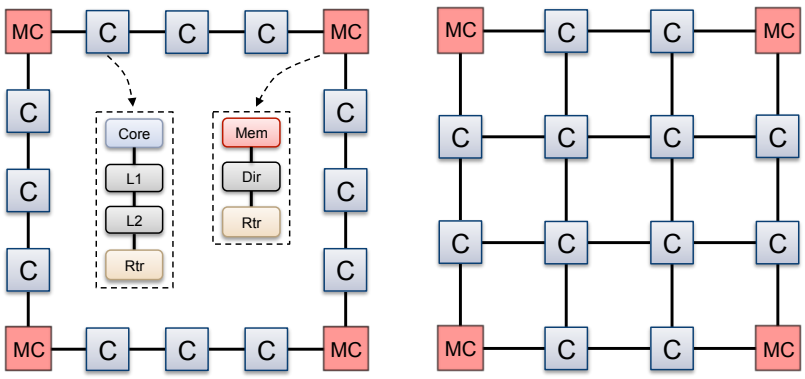
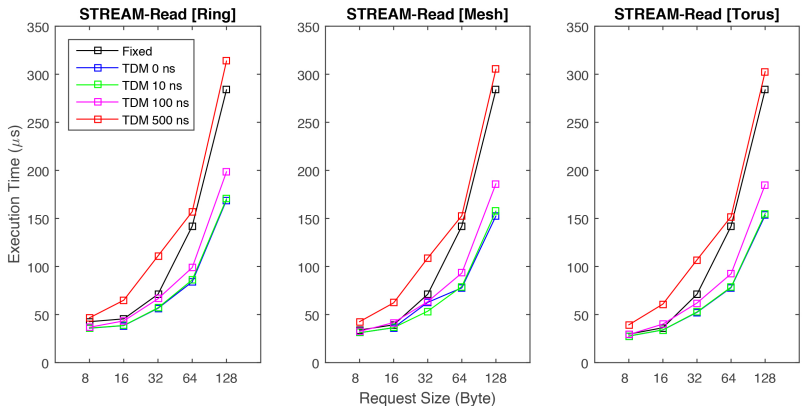


Figure 3.4: Simulated 12-core 4-MC architecture in ring and mesh configuration.

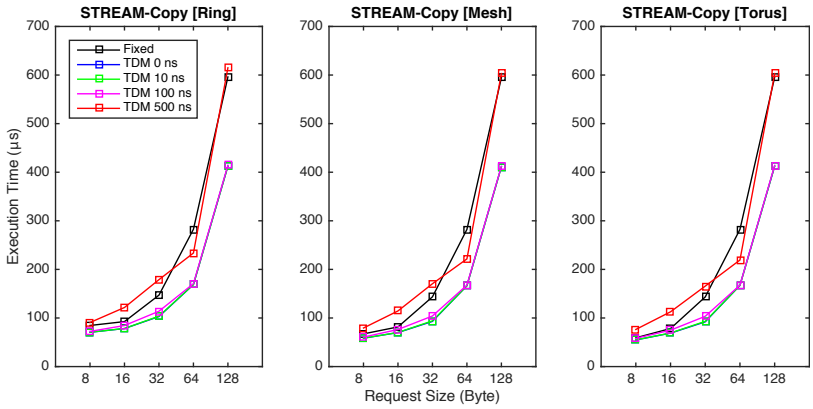
the SiP switch.

3.5 Performance Evaluation

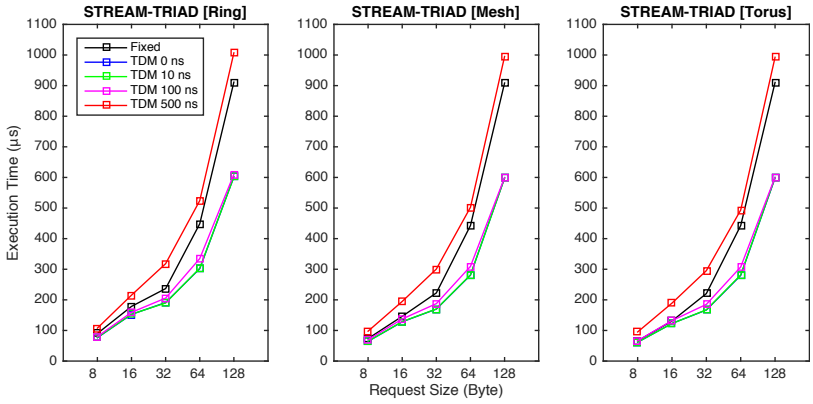
We use three data-intensive applications – STREAM , 3D stencil, and K-mean clustering benchmarks to compare system performance under both fixed-connection mode and photonic-switching mode. The simulated architecture consists of 12 cores and 4 memory controllers connected in a ring, mesh, or torus topology (Fig. 3.4). In the ring case, there is a memory controller every three cores. In the mesh and torus cases, the memory controllers are located at the four corners of the NoC. Each core has 32 KB of L1 cache and 256 KB of L2 cache. Each memory interface includes a directory. Each memory stack has 96 GB/s bandwidth, and an access latency of 30 ns. The links on the ring are bi-directional, with a width of 8 B for each direction. The clock frequency of the processor is 2.1 GHz. Considering the communication overhead, we assume that a NoC link has a bandwidth of 12.6 GB/s (6B $\times$ 2.1 GHz). In this study, we assume only the read (injection) direction uses the switching functionality of SiP devices, while the write direction still uses fixed connections. This complies with the fact that a SiP switching plane is usually used for one direction of communication.



(a) STREAM-Read



(b) STREAM-Copy



(c) STREAM-TRIAD

Figure 3.5: Execution time of STREAM with different kernels (read, copy and TRIAD), under fixed or TDM-switched memory connection modes.

STREAM Benchmark

The `STREAM` benchmark is a simple synthetic benchmark program that measures sustainable memory bandwidth and corresponding computation rate for simple vector

kernels. We evaluate three kernels for `STREAM` :

- READ: Each core reads its part of an array from the main memory; no other operation is performed.
- COPY: Each core reads its part of the array and writes the data to a second array, i.e. $a(i) = b(i)$.
- TRIAD: Each core reads its part of two input arrays and performs a MULT-ADD operation, i.e. $a(i) = b(i) + q * c(i)$, and writes the result to a third array.

The READ kernel evaluates the pure data injection capability of the system, while the COPY kernel evaluates patterns with balanced reads and writes (the write operation still takes the fixed memory interface). The TRIAD kernel evaluates actual compute performance by further mixing memory accesses with FLOP operations.

Hotspot Scenario

To mimic the hotspot effect, the address space to be accessed is set to reside in the first memory module. This arrangement creates high traffic load, including both memory read requests and responses, around the first MC tile. The execution time of `STREAM` kernels with fixed injection points vs. TDM-switched injection points is shown in Fig. 3.5. For the TDM case, channel switching latencies ranging from 0 ns to 500 ns are considered. On the X-axis, we vary the burst length of each memory request from 8 bytes to 32 bytes. For switching latencies less than 100 ns, the TDM-switched injection achieves lower execution time than the fixed injection in all three tests. And the fixed injection has similar performance to that of TDM with 500 ns switching latency. Additionally, the speedup increases as the request size increases. For the READ kernel, which maximizes the benefit of switched injection over fixed injection, the speedup reaches about 2x when the request size approaches 64 bytes – the cache line size of most modern processors. The reason for this increase in speedup

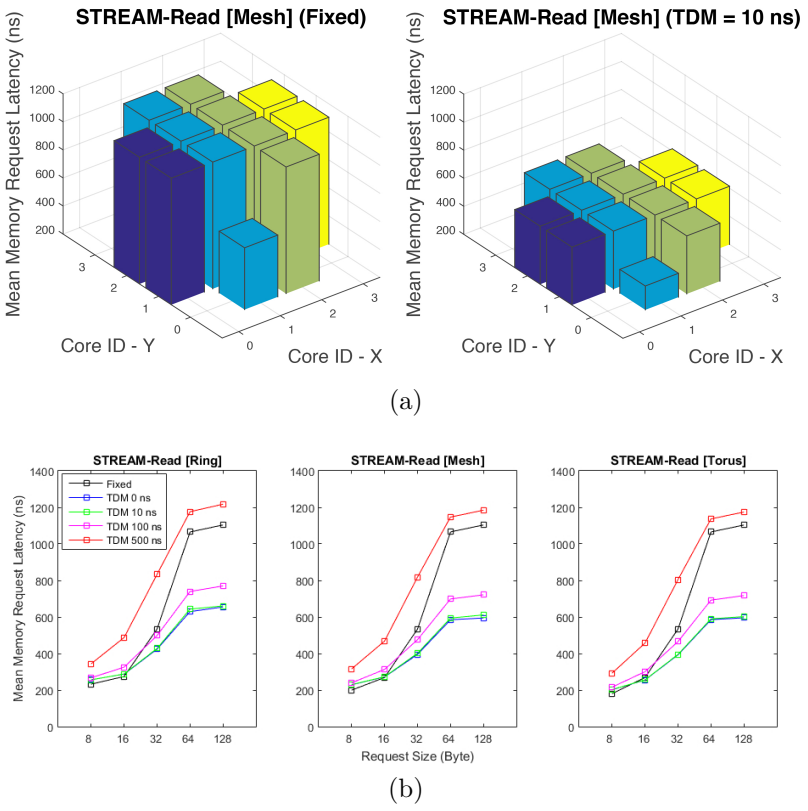


Figure 3.6: (a) Respective memory request latencies of 12 cores (in mesh topology) under fixed vs. TDM modes, request size = 64B; (b) Mean memory request latency across all 12 cores.

is that bigger request size more easily congests the NoC in the fixed-injection mode. In comparison, the switched-injection mode can evenly distribute the memory response traffic over all available injection points, alleviating the hotspot effect by a factor of 4x.

A comparison of memory request latencies (MRL) under the fixed mode versus the TDM mode is presented in Fig. 3.6 for the READ kernel. Fig. 3.6a shows the MRL of respective cores of the 4x4 mesh topology, with a request size of 64 bytes (size of a cache line). The four corner tiles are not shown since they are memory controllers. It is clear that the TDM mode, with 10 ns switching latency in this case, cuts down the MRL by half compared to the fixed case. Another observed effect is that Core (1,0), being relatively closer to the first MC, has lower MRL than the other

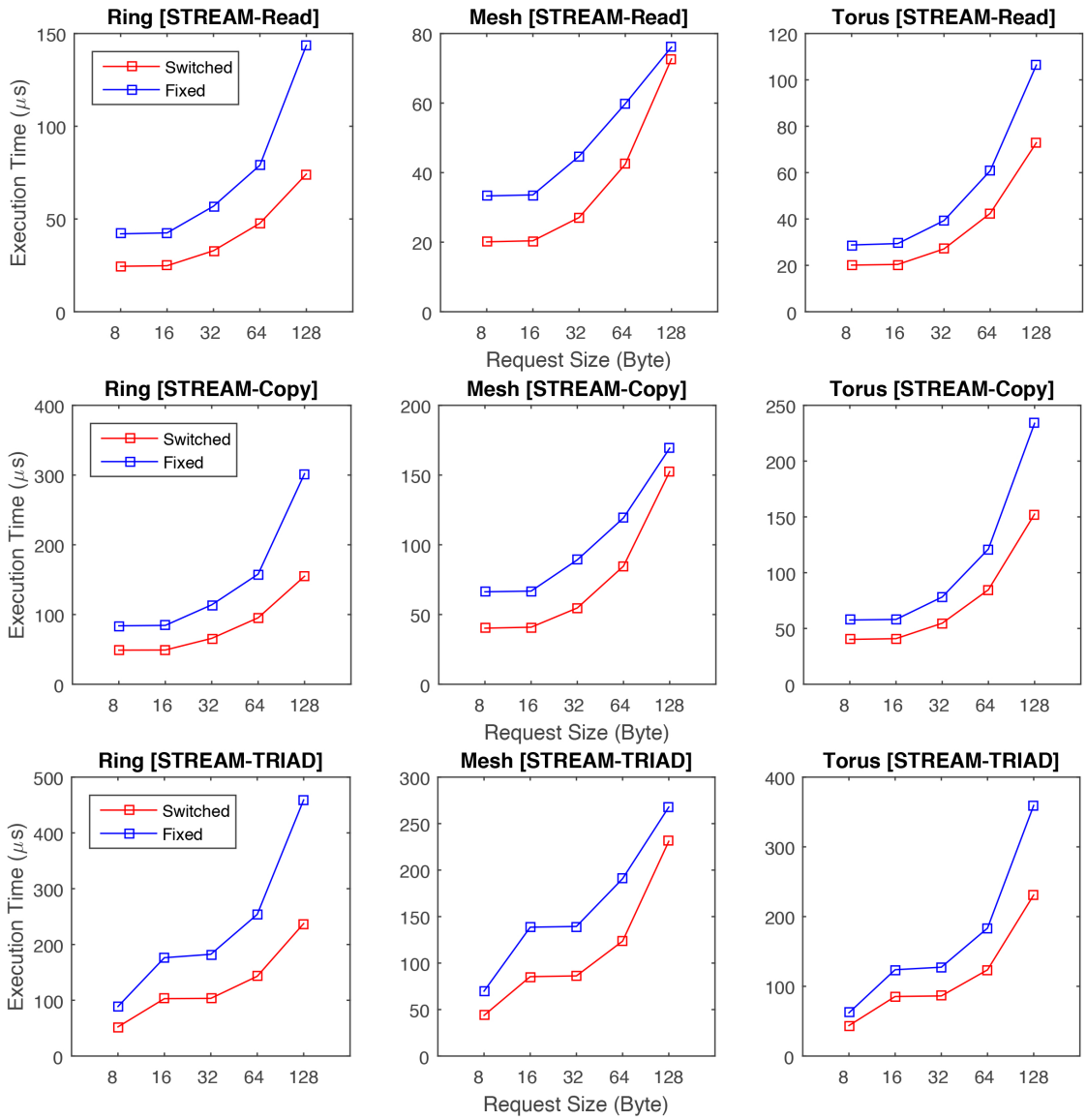


Figure 3.7: Execution time of various STREAM kernels using native or affinity-optimized memory interfaces.

cores, which is in line with the NUMA effect. Fig. 3.6b shows the average MRLs across all 12 cores under different topologies and request sizes. The TDM mode also significantly reduces the average MRLs by avoiding NoC congestions.

Affinity Scenario

To demonstrate the benefit of photonic switch enabled core-memory affinity, we assume that data is stored in the memory in a non-local fashion. While the cores still perform jobs in parallel, each core must access its data via a distant memory. Specifically, we assume that the mismatch distance is half a ring. With a SiP switch, one can change the mapping between the interfaces and memory modules and compensate for this mismatch, thereby eliminating the non-local efficent.

Fig. 3.7 shows the execution time of the three kernels in both fixed-connection mode (affinity-suboptimized) and switched mode (affinity-optimized). For the READ and COPY kernels, the switchable mode can achieve a constant speedup of 2x, regardless of the request size. For TRIAD, the speedup increases as the request size increases.

K-means Clustering

We include K-means clustering in our evaluation because of the prevalence of data mining and machine learning algorithms. Specifically, we use `streamcluster` from the PARSEC benchmark suite [96] to emulate a K-means clustering method with streaming characteristics. `streamcluster` solves an online clustering problem faced by many machine learning applications [97]: For a stream of input points, find a predetermined number of medians such that each point is assigned to its nearest cluster center. `streamcluster` has an intensive memory-to-FLOPS instruction ratio of 9.48 : 11.6 as reported in [96]. The `streamcluster` kernel simulated here implements a data-parallel pthreads model and has a medium-level parallelization granularity.

The memory access of `streamcluster` has a typical hotspot pattern. As reported in [11], 97% of the memory accesses are towards the first memory module, creating much congestion in the section of the NoC near the first MC.

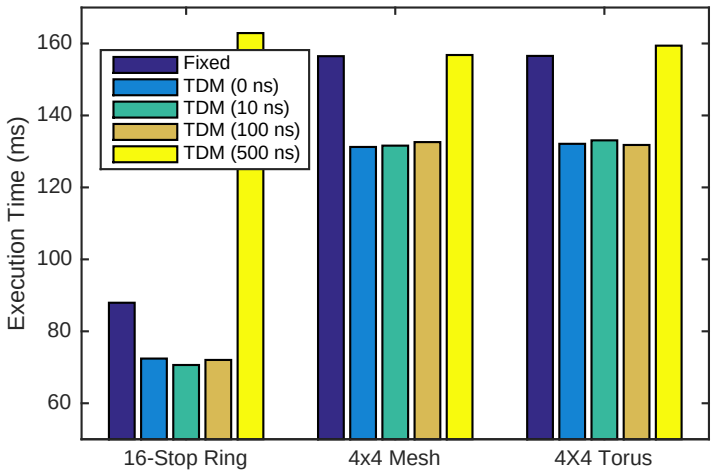


Figure 3.8: Run time of `streamcluster` with fixed and reconfigurable (TDM) memory connections

Fig. 3.8 shows the simulation result comparing the fixed memory connection versus the TDM memory connection. With the topologies evaluated, maximum runtime reductions (under 0ns TDM switching) are 17% for ring, 15% for mesh, and 15% for torus. Another trend seen in the data is that even when TDM switching delay increases (up to 100 ns), the run time is not significantly affected, which means that some real applications like `streamcluster` can be very tolerant to optical switching latency.

3D Stencil

`3D stencil` is employed by a large fraction of scientific applications including heat diffusion, electromagnetics, and fluid dynamics. The operation sweeps over a spatial grid and performs nearest neighbor-based computation (called *stencil*). In this work, we set the stencil size to 27, 3 points in each of the XYZ dimensions. Since the data structure swept by the stencil operation is typically much larger than the available cache space, `3D stencil` is often intensive in off-chip memory access. To address this problem, software strategies such as tiling [98] and auto-tuning [99]–[101] have

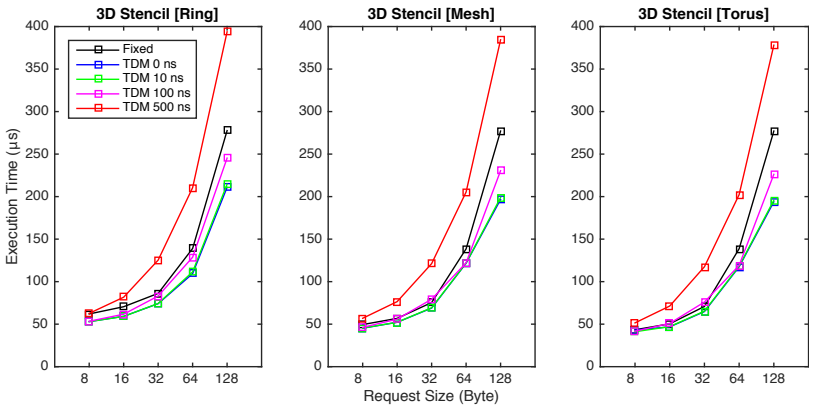


Figure 3.9: Execution time of 3D-stencil, under fixed or TDM-switched memory connection modes.

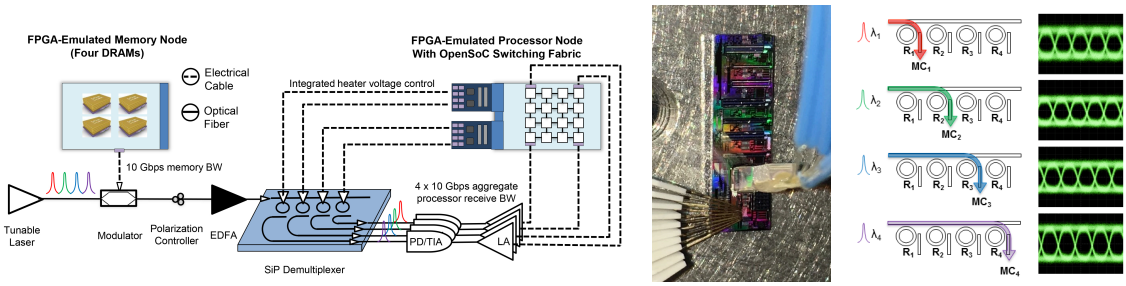
been proposed to improve cache reuse (i.e. temporal locality). Effective core-memory bandwidth, however, is still identified as a major performance bottleneck [100], [102].

In our simulation, we set the stencil problem size to $nx = ny = 10, nz = 20$, which allows data to be stored continuously in memory. We iterate over the data width space (i.e. size of each grid point) from 8B to 128B, and show the corresponding simulation result in Fig. 3.9. Mesh topology shows maximum performance difference between the fixed and TDM cases, where TDM with 10ns switching reduces the run time by 23% over the fixed case under 128B grid point size.

3.6 Hardware Demonstration

Experimental Setup

Fig. 3.10a shows the implemented experimental configuration. Two Altera Stratix V GX FPGAs with 4x10-Gbps transceivers were used to emulate, respectively, a many-core processor and four memories. The four memories were mapped to a single transceiver channel, with the transmitter coupled to a $LiNbO_3$ commercial modulator via a QSFP-to-SMA interface to drive on-off keying (OOK) non-return-to-zero (NRZ) data onto the tunable laser output. The tunable laser was programmed to switch



(a) Memory connected to OpenSoC NoC via a SiP demux

(b) SiP demux chip

(c) Eye diagram

Figure 3.10: (a) Many-core multi-memory testbed based on a fast tunable laser, a SiP wavelength demux and an OpenSoC NoC. (b) Chip image with probes landed and fiber array attached. (c) Measured eye diagrams.

among four channels (λ_1=1548.1 nm; λ_2=1551.3 nm; λ_3=1554.5 nm; λ_4=1557.7 nm) in a round-robin fashion. The optically modulated channels were amplified with an erbium-doped fiber amplifier (EDFA), and a single WDM lightpath at 14 dBm was injected into the SiP demultiplexer. Four rings in this device were used to selectively drop the four data channels to different output ports. At each drop port, a 12.5 GHz PIN/TIA optical-to-electrical converter and limiting amplifier (LA) were used to receive and condition the signal injecting electrical data into the processor's memory interface.

VHSIC Hardware Description Language (VHDL) was used to code synthesizable hardware for a communication backend to inject and collect user-defined data. Other VHDL was implemented to control four digital-to-analog converters (DACs), used to tune the four rings' resonances via integrated heaters, providing programmable reconfiguration of the SiP demux. VHDL-coded, state-based logic was written to reconfigure and synchronize the four transceiver channels together with reconfiguration of the SiP demux.

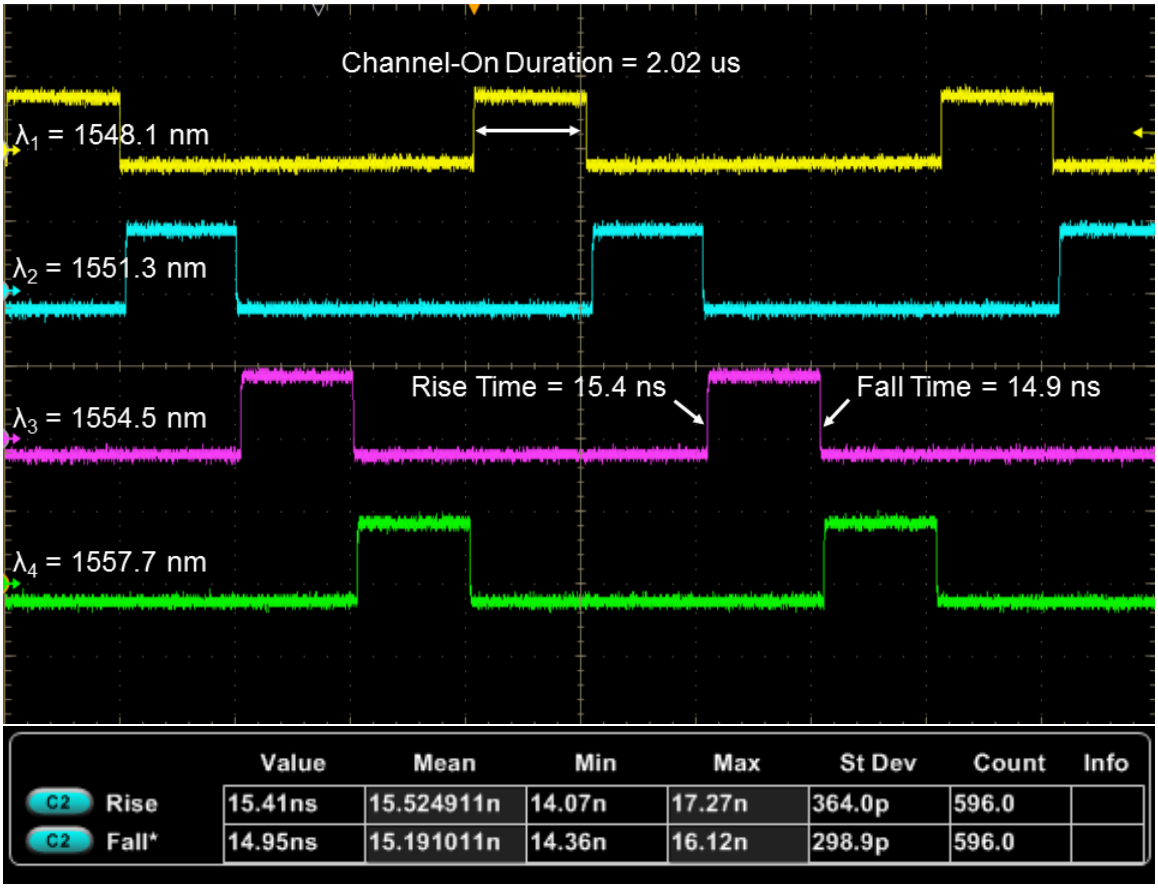


Figure 3.11: Experimental demonstration of time-sequenced switching diagrams with a channel switching time of 15 ns.

Characterization of Optical Switching

Physical layer network performance was evaluated by collecting 10-Gbps eye patterns generated by FPGA-driven 2^{31}-1 pseudorandom data on the four optical channels, routed in various static configurations of a 1-by-4 network (Fig. 3.10c). Clear and open eyes indicate that the physical layer of the network is capable of error-free data transmission.

Fig. 3.11 depicts the measured responses at the four drop ports of the SiP demux using a four-channel oscilloscope. It shows that the four channels are turned on/off in a cyclic order due to the switching of the fast tunable laser. The duration of the channel-on state is set to 2.02 microseconds, for transmitting 2 KB data. The

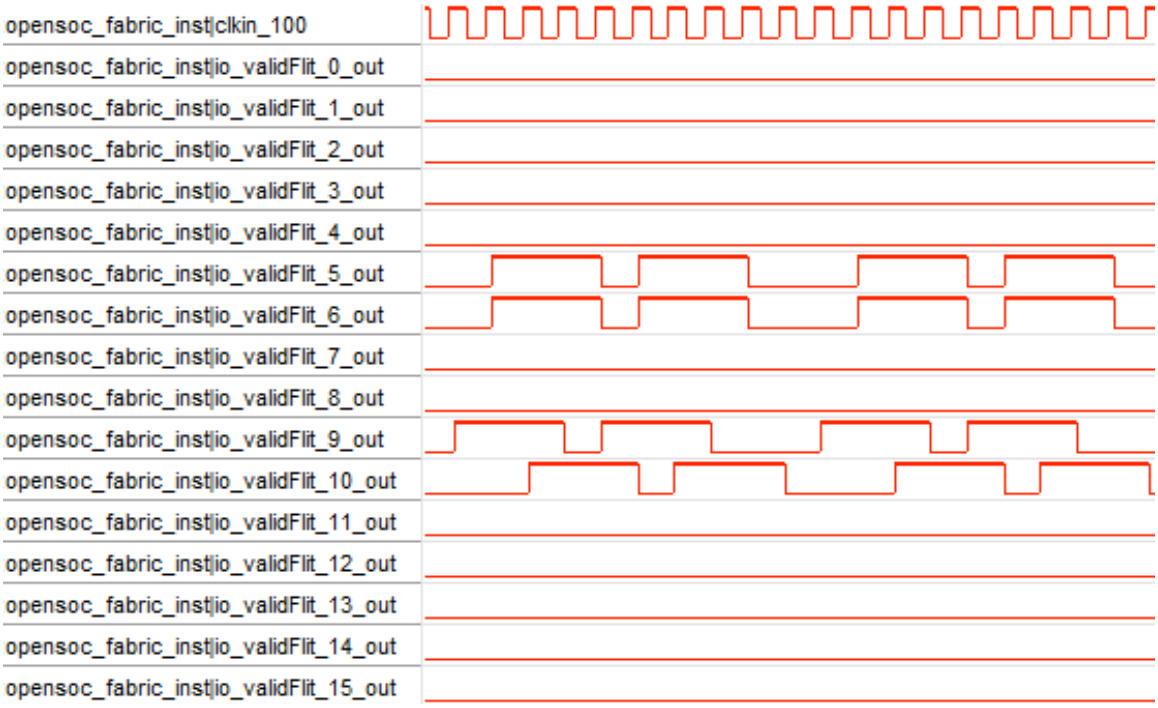


Figure 3.12: OpenSoC switch fabric outputs showing four cores simulatenously receiving memory data from four memory interfaces

measured switching time (defined by the rise/fall time) of the fast tunable laser is 15 ns. Based on the above-described simulation results, this reconfigurable memory interconnect with a switching time of 15 ns can bring significant speedup compared to the conventional fixed case.

OpenSoC Fabric for NoC

We take a step further by implementing a 4-by-4 mesh NoC in the processor-side FPGA. The RTL of the NoC is generated by *OpenSoC Fabric* [103], a Chisel based on-chip network generator. This fabric allows the flits to be routed from any memory interface to any destination core. It adopts the backpressure-based flow-control schemes, and generates ready and valid signals at the I/O ports of each tile. In the second experiment, the control logic at the processor side initiates the data transmission. As a response, the four memory modules send response data using separate

Table 3.1: Routing latency (clocks) from port (0, 0) to port (X, Y) in unit of clock cycles

	X=0	X=1	X=2	X=3
Y=0	9	14	19	24
Y=1	14	19	24	29
Y=2	19	24	29	34
Y=3	24	29	34	39

Table 3.2: Breakdown table of latency, and scaling the emulator by a factor of 10

Component	FPGA-Emulated	Actual System
Channel Switching	15 ns	15ns
Clock Data Recovery	200 ns	20 ns
OpenSoC Routing	50 ns	5 ns
Total Latency	255 ns	40 ns

wavelength channels. The control logic tunes the four SiP rings to drop the four data streams at their respective memory interfaces – four corners of the OpenSoC mesh (tile ID 0, 3, 12, and 15). The flits are then routed by the synthesized routers to their respective destination core.

Fig. 3.12 depicts the output timing diagram captured using SignalTap tool offered by Altera's Quartus-II development environment. The top-most signal is a 100 MHz clock signal that drives the OpenSoC fabric. The following 16 signals are the output valid flags of the 16 NoC tiles from the router to the core. Output ports 5, 6, 9, 10, corresponding to the four requesting cores, have flits coming out successfully, while other ports remain quiet.

Table 3.1 shows the latency of the OpenSoC fabric from input port 0 to different output ports. Latency increases by 5 clock cycles as the hop distance increases by 1. When the average hop distance is reduced by 5 (equal to 25 clock cycles) by using the reconfigurable multi-core to many-memory interconnects, the average latency is reduced by 25 ns with a clock speed of 1 GHz.

Table 3.2 shows the breakdown of the read latency from the memory to the core. The values listed in the "FPGA-Emulated" column are the measured results in the

experiment using FPGA at a clock of 100 MHz. The values listed in the "Actual System" column are the results we think our proposed scheme can achieve in a real CPU system when using an application-specific integrated circuit (ASIC) with a clock of 1 GHz. The overall latency of the "Actual system", with the optical switching operation, is 40 ns. This latency is comparable to memory transaction latencies of state-of-the-art memories like HMC and HBM [104].

3.7 Summary

We developed a method to deliver memory traffic to the requesting cores in a network-distance-efficient manner using fast tunable lasers and SiP demultiplexers, at nanosecond-level memory channel switching latency. Performance evaluation based on a many-core processor model verified that this fast tunability has the potential to significantly improve the performance of data-intensive benchmarks. We believe that the proposed reconfigurable SiP memory interconnect, in conjunction with the ongoing switching time improvement from both industry and research, can be a promising solution to memory access issues such as NUMA and NoC hotspots in the many-core era.

Part II:

Avoiding Path Setup Overhead

Chapter 4

Reusing Optical Circuits in HPC Applications

Abstract

Optical interconnects can support high-bandwidth, end-to-end connectivity over warehouse-scale distance. However, due to the circuit switching nature and additional peculiarities, optical links generally show longer path setup delays. These delays are a major obstacle in exploiting the high bandwidth of optics for application speedups, especially when low-latency communication is required. These limitations can be overcome by maintaining a set of frequently used optical circuits based on the temporal locality of the application and by maximizing the number of reuses to amortize the overheads. However, since circuits cannot be simultaneously maintained between all source-destination pairs, the set of selected circuits must be carefully managed. This chapter applies techniques inspired by cache optimizations to intelligently manage circuit resources with the goal of maximizing the circuit "hit rate". We propose the concept of "circuit reuse distance" and design circuit replacement policies based on this metric.

4.1 Motivation

Silicon photonic (SiP) interconnects, which have been shown to provide large bandwidth densities at high energy efficiencies, can scale over warehouse distances and provide end-to-end connectivity across HPC platforms [105]. Despite this reach advantage, SiP interconnects also have a set of peculiarities and special operation requirements. For example, resonance based devices such as microring resonators will require wavelength tuning to reach the designed operating wavelengths [106]. Resonator devices are also sensitive to surrounding environmental temperature due to the high thermal-optic constant of silicon, and thus require thermal stabilization or re-initialization [107]. These requirements add up to longer link initialization delays compared to electronic links. The link initialization delays further increase with cascaded switching components.

The optical interconnect system delays directly add to the execution time of HPC applications and are a major obstacle in exploiting the high bandwidth of optics for application speedup. In particular, the latency penalty could be especially detrimental in scenarios when remote direct memory access (RDMA) [108] is enabled or when small messages are used.

Compared with traditional two-sided communications, RDMA mitigates synchronization overheads such as tag matching between the sending and receiving processes [109]. Instead, communication can be initiated by only one side using tools such as MPI One-sided [110], [111], OpenSHMEM [112], PGAS [113], etc. RDMA enables a process to directly access remote memory space of another without involvement of the latter. To maximize this advantage, physical layer communication with small initialization overhead is desirable. Increasing setup delays can easily negate the advantages of RDMA, making the link similar in performance to having two sided synchronization delay. Instead, latencies for remote and local memory access latencies should be unified as in a flattened memory architecture. Thus it is critical for

SiP circuit-switched networks to provide circuits in such way that an RDMA request can immediately find a corresponding circuit upon arrival (we call this a circuit hit). Otherwise, the request sees a circuit miss and has to suffer from the setup penalty. The role of circuits in a circuit-supported RDMA system resembles that of caches in microarchitectures.

Although the initialization delay of SiP devices is hard to minimize presently due to the limitation of the silicon thermal constant [114], such penalty can be overcome through careful architectural design. One such method is to explore the temporal locality in an application's communication pattern, where a node could reference a remote memory space multiple times within a short period. If the circuit corresponding to the requested end point already exists, messages can be immediately transmitted, avoiding the circuit setup penalties. Taking advantage of temporal locality, a set of optical circuits can be maintained for the frequently accessed neighbors, significantly increasing the circuit hit rate and hence the application performance. Maximizing the number of reuses of these circuits also helps amortize their initialization overheads.

A requested circuit, of course, will not always exist to be reused. This is because optical connections cannot be maintained for all source-destination pairs, and because application communication patterns can change over time. The challenge then becomes to carefully select and update the set of circuits to maintain.

We apply techniques inspired by cache optimizations to intelligently manage circuit resources with the goal of maximizing the circuit hit rate. We propose the concept of "circuit reuse distance" and design circuit replacement policies based on this metric in order to avoid circuit setup penalty. Since our work focuses on optimizing replacement performance at runtime, an estimation of the next reuse distance of a circuit is needed. We propose two predictors for predicting the circuit reuse distance and show that a novel Transition Matrix Based Predictor (TMBP) can provide up to 40% accuracy gain compared to the traditional Maximum Likelihood Based

Predictor (MLBP). Two reuse distance-based replacement policies are also studied: the Farthest Next Use (FNU) policy and the Minimum Reuse Score (MRS) policy. Simulations based on scientific benchmarks show that both policies have the potential to achieve much higher hit rates than the Least Recently Used policy. Considering the distinction between circuits and caches, we also investigate the tradeoff between the hit rate and energy consumption. Finally, we collect data on the circuit setup delay using an FPGA-controlled network testbed containing the latest SiP devices.

4.2 Source of Delays in Silicon Photonic Links

The reliance on resonance makes microring devices highly sensitive to environmental temperature and fabrication variation. The high thermal optic coefficient of silicon means that device resonance varies strongly with temperature, which must be maintained within sub-kelvin accuracy for normal system operation. To overcome this thermal dependence a variety of methods have been demonstrated [107], with very promising results using active control systems to drive a local integrated heater to maintain temperature stabilization. Active control systems increase the circuit setup latency because they require time to stabilize [114].

Fabrication variation causes an inherent offset between realized microring resonances and the laser wavelengths in the system. The overall silicon photonic control system must be also capable of initializing microring elements to their operating wavelengths in an operation called wavelength locking. The time required to do so is on the order of tens to hundreds of microseconds and is limited by the thermal time constant of the device [114]. It should be noted that most currently demonstrated control systems require optical power going into the circuit to maintain stability. When the path is turned off device temperature will no longer be stable and can require re-initialization. This is another motivation to selectively maintain active

circuits and maximize reuse.

4.3 A Reuse Distance Based Approach

Efforts have been made to explore the use of optical circuits at system scale [71], [115] or for memory access [116]–[118]. The author in [119] proposed an "asynchronous circuit programming" model, which explicitly setup circuits before communication. However, none of these works considers minimizing setup penalties, especially, in view of the temporal use patterns. In this work, we investigate the reuse patterns of circuit communications and optimize the circuit management based on the metric of *reuse distance*.

Reuse distance has been an important metric in cache performance optimizations [120]. In this work we consider reuse distance from the circuit perspective and specifically, in view of a source node. A reuse distance of a circuit is defined as the number of circuit requests from its source node S between two consecutive calls to this circuit. For example, if a sequence of circuit requests made by node S is C, A, B, F, E, C (labeled by destination nodes), then the reuse distance of circuit C is 4. We also call the outgoing circuits maintained by a source node its circuit set. Despite this specific perspective, the proposed techniques can also apply in view of destination nodes or the entire network.

The reuse distance of a circuit can be also measured in time. However, the count-based distance as described above (short as "reuse distance" hereafter) and time-based distance (short as "time distance" hereafter) may play different roles in different optimization problems. Similar to cache optimization, we rely on the former for circuit replacement design. However, a difference between circuits and cachelines is that circuits consume static power due to use of lasers. While flushing a cacheline before a miss makes little sense, turning off a circuit that is not likely to be used in

near future could save energy. We hence also repy on the *time distance* for optimizing the tradeoff between hit rate and energy consumption.

4.4 Profiling Circuit Reuse Distance

For measuring the temporal locality in scientific applications, we start by analyzing the distribution of reuse distances based on a group of representative HPC applications (whose description is in Appendix A). This will guide us in designing prediction and replacement policies. As a node progresses through its workload, it issues communication requests. The node counts the number of circuit requests it makes and maintains a table to keep track of the last request index for each of its circuits. Upon a new circuit request, the entry corresponding to the circuit is consulted, and the difference between the current and the last request index of the circuit is a sample of the reuse distance of the circuit. By collecting such samples along program execution, an estimation of the reuse distance distribution is obtained. A fine-grained estimation of the distribution is not necessary. Instead, we cover a wide distance range and use power-of-two based bin divisions, i.e. $[0], [2^0], [2^1, 2^2), [2^2, 2^3)$, etc. The time distance can be similarly collected based real time elapsed on the node.

The resulting distance histograms are shown in Figs. 4.1-4.3. Each application leads to a different reuse pattern. Applications such as miniMD show very nonuniform distributions, while some others (e.g. GTC) are more uniform. Such difference is related to the application's communication degree (i.e. the number of nodes toward which a given node issues most of its traffic), as well as irregularity of the communication pattern. The results show a high probability that a source node will reuse its circuits within a small distance. For instance, reuse distances in miniMD with a value smaller than 8 comprise 90% of the samples, while this percentage is of 43%, 70% and 60% for miniFE, GTC and HPCCG, respectively. Applications such as miniMD,

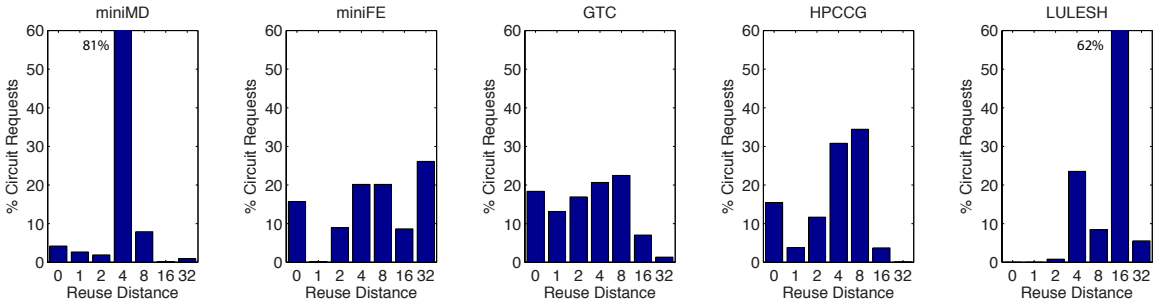


Figure 4.1: Distribution of reuse distances for HPC benchmarks (64 nodes). Each bin corresponds to a range between its own label (included) and the next label (excluded), same below.

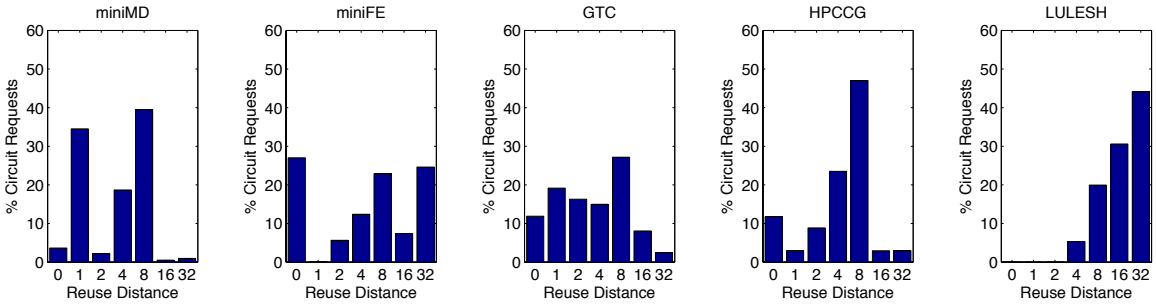


Figure 4.2: Distribution of reuse distances for HPC benchmarks (256 nodes); 512 nodes for LULESH.

miniFE and GTC even show a high percentage for reuse distances from 0 to 2. Only LULESH shows relatively longer distances, which is due to its higher communication degree. Fig. 4.3 presents the time distance distribution and show that a large portion of the circuits are reused within tens of microseconds. Applications such as miniMD, GTC and HPCCG, even show a high percentage for time distances less than 16 μs.

These results provide evidence that there is a high potential to reuse a circuit for multiple near requests, thereby amortizing setup delays. While analyzing a posteriori communication patterns is helpful, using such information for better runtime optimization is yet another thing. In particular, we need to determine which circuit should be maintained to seize the reuse opportunities, given that the size of the circuit set is limited. We hence explore online techniques that utilize the observed circuit-use history at runtime to optimize circuit replacements.

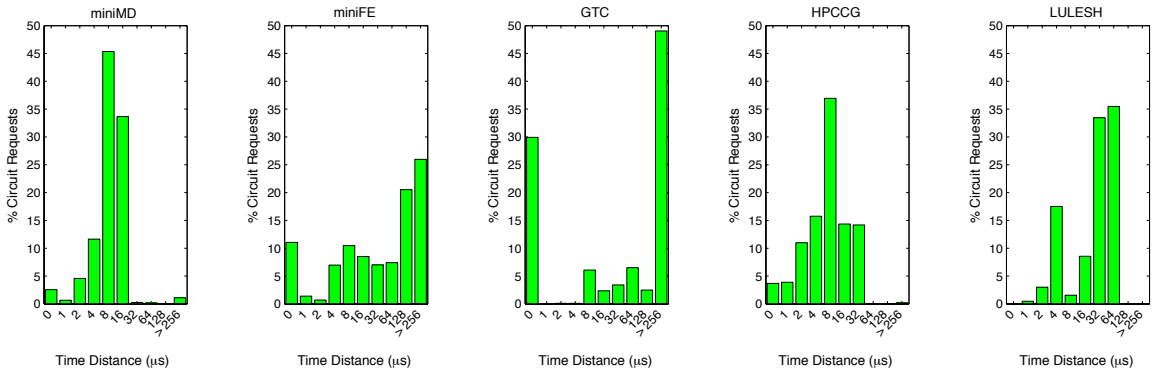


Figure 4.3: Distribution of time-based reuse distances for HPC benchmarks (64 nodes). For miniMD, GTC and HPCCG, a high percentage of circuit reuses are within 16 μs.

4.5 Predicting Circuit Reuse Distance

One key step of utilizing observed reuse history for circuit replacement is to predict the reuse distance for replacement candidates when a circuit miss occurs. In this way, the circuit that is the least likely used in the near future can be removed. In this section, we describe two techniques for predicting the reuse distance. Our online methods presented here differ from previous works that considered offline cases. We also compare the prediction accuracy of the two predictors.

Maximum Likelihood Based Predictor

The Maximum Likelihood Based Predictor (MLBP) looks at the currently-collected reuse distance distribution of a circuit and selects the bin with the highest frequency as the prediction. Although the prediction has the maximum likelihood, MLBP suffers from two major drawbacks: 1) its prediction accuracy largely depends on distribution pattern: if the distribution has one or more bins with comparable frequency to the highest bin, the prediction accuracy is hindered; 2) MLBP neglects the temporal pattern of the reuse distance sequence collected.

Transition Matrix Based Predictor

The Transition Matrix Based Predictor (TMBP) avoids the drawbacks of MLBP. It explores the temporal aspect of the reuse distance sequence observed for a circuit and offers prediction based on transition patterns in the sequence. To extract the pattern, TMBP models the transition of reuse distance using a Markov chain (Fig. 4.4). The states of the Markov chain correspond to the histogram bins, while the transition matrix represents the probability of the reuse distance transiting from one bin to another. Each time a reuse distance sample is collected, the matrix element corresponding to the transition from the last bin to the current bin increments by 1. Upon predicting the next reuse distance, TMBP finds the bin to which the current bin has the greatest transition probability. Such Markov chain is maintained per circuit.

Prediction Performance

Fig. 4.5 shows how our two prediction techniques lead to different prediction accuracies across the applications and problem sizes. Each time a circuit is used, its distance until the next use is predicted. If this prediction has the same log2 value as the reuse distance observed (at the next use), then the prediction is considered accurate; otherwise, it is considered not accurate. In the case of miniMD, MLBP sees a severe accuracy drop when the number of nodes increases from 64 to 128 and 256. The reason lies in Figs. 4.1 and 4.2, where the distribution of miniMD transforms from a single-tower shape into a two-tower one. In comparison, the accuracy of TMBP remains at high, with a gain of 40% and 36% over MLBP observed in the cases of miniMD and HPCCG, respectively.

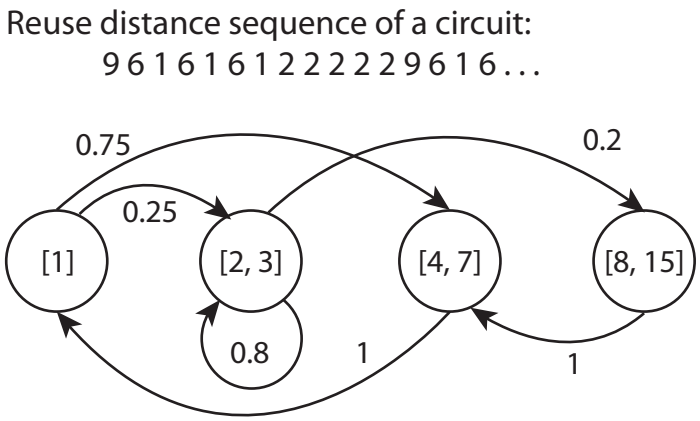


Figure 4.4: Example for Transition Matrix Based Preditor. Upper: reuse distance sequence of a circuit. Lower: modeling of the sequence transition using a Markov chain. Each state of the Markov chain corresponds to a bin in the distribution histogram.

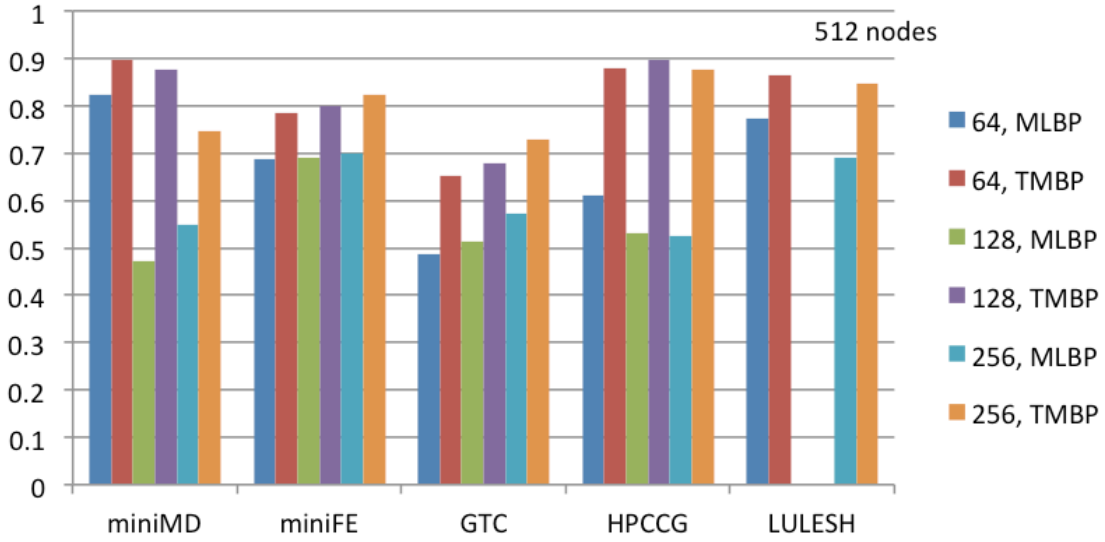


Figure 4.5: Reuse distance prediction accuracy of Transition Matrix Based Preditor (TMBP) versus Maximum Likelihood Based Predictor (MLBP), across different benchmarks and different numbers of nodes. TMBP shows as much as 40% and 36% higher accuracy than MLBP in cases of miniMD and HPCCG, respectively.

4.6 Optimizing Circuit Replacements

Prediction of circuit reuse distances allows us to approximate an optimal replacement algorithm because we can attempt to preempt future communication patterns with

appropriate circuit configurations. The circuit that is the least likely used in the near future is "sacrificed" when a circuit miss occurs. In this section, we describe two ways of using the reuse distance information for circuit replacement.

Farthest Next Use (FNU)

The FNU policy selects for replacement the circuit that is going to be reused in the farthest future. Each time a circuit miss occurs, circuits that are currently maintained but not in data transmission become replacement candidates. Similar to [120], the estimated time to access (ETA) a circuit can be calculated by adding the predicted reuse distance to the circuit's last use time minus the current time. However, not every circuit has a positive ETA, some may have a negative value due to the passing of its expected access. In this case, the decay time is used, i.e. how much time a circuit has not been used. Different from [120], we also use the decay time if credibility of the ETA prediction is not high. The circuit with the largest value for ETA or decay time will be replaced.

Minimum Reuse Score (MRS)

In MRS, each circuit is associated with a score regarding its frequency of reuse. Instead of granting every reuse with equal weight, reuses within smaller distances retain higher "values". Each time a circuit is used, its score increases by (2^{maxbin} - reuse distance). Each time a replacement is needed, the vacant circuit with the lowest score is replaced.

Replacement Performance

Performance of the two aforementioned replacement policies is compared with the Least Recently Used (LRU) policy via simulation. Our simulation assumes a fully-

connected network topology and that the destination node has adequate receivers (slightly greater than its communication degree) to receive incoming circuits. These assumptions make sure that network contention and receiver contention will not affect the state and replacement of the circuit set at source nodes. Global network-based or destination-based replacement can be also investigated with our proposed techniques and will be included in our future work.

As Fig. 4.6 and 4.7 show, in most cases FNU (based on the prediction result of TMBP) and MRS lead to much better or comparable hit rate than the LRU policy, and hence the setup penalty due to circuit misses is minimized. It is worth noting that FNU and MRS perform better than the other in different cases. The reason is that the two policies account circuit history differently. MRS collects scores from the beginning of an application; a circuit's score acquired during an early phase could still secure its position in the circuit set in a later phase even if the circuit is not frequently used in the latter. Such effect could keep dead circuits that have long been vacant from exiting the circuit set. In the case of FNU, if a circuit has long passed its expected access time, the increased decay time will flush it out of the circuit set. Hence, the performance of FNU is better than MRS in many cases, except for LULESH. From the distribution, we know that LULESH shows more likelihood towards long reuse distances. FNU replaces these long-distance circuits, which however, contribute most reuse opportunities.

4.7 Energy Consumption Tradeoff

Although circuit set and cache share many similarities in shaping the hit/miss characteristics of data accesses, several notable differences persist. An obvious difference is that maintaining a circuit explicitly costs time-proportional energy consumption (e.g. laser power), while maintaining a cacheline costs little. Maintaining circuits as

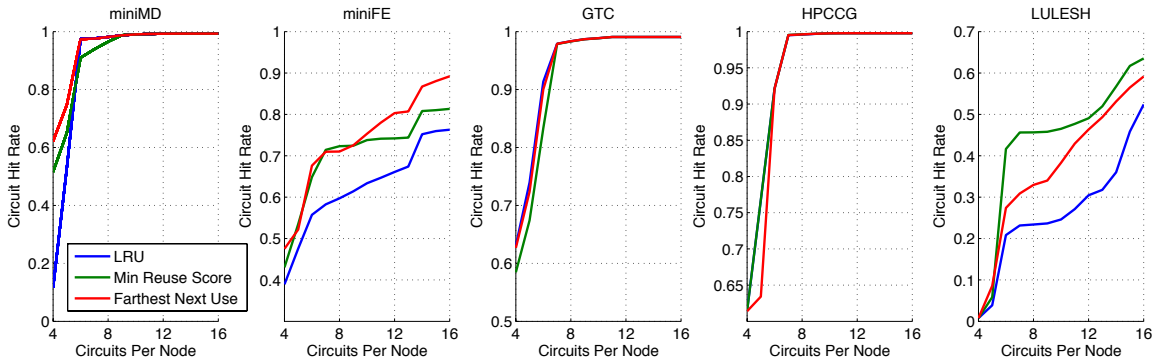


Figure 4.6: Circuit hit rates (64 nodes) for replacement policies: LRU, Farthest Next Use and Minimum Reuse Score, across different benchmarks.

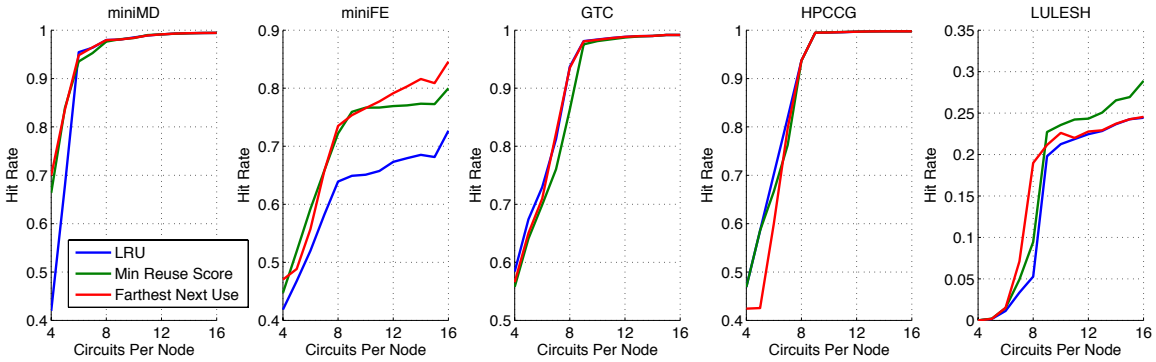


Figure 4.7: Circuit hit rate (256 nodes) for replacement policies: LRU, Farthest Next Use and Minimum Reuse Score. 512 nodes for LULESH.

long as possible can help further reduce the miss rate; however, such reduction comes at a price of excessive energy consumption and the reduction might not be proportional to the price paid. A long-time-no-use circuit can still remain in the circuit set if no replacement occurs. In this case, a mechanism is needed to actively turn off the circuits without the help of replacement. One such method is to predict the time distance of a circuit—if the circuit is not going to be used again until far in the future, it will be turned off. Note that such proactive turn-off (PTO) will not lead to additional penalty if the circuit is to be replaced by a miss before a reuse. However, PTO could indeed lead to the drop of hit rate if the time-distance prediction is not accurate and a circuit reuse does arrive. Instead, if the time-distance prediction is trustworthy, a circuit could be turned off right after its use if its predicted next

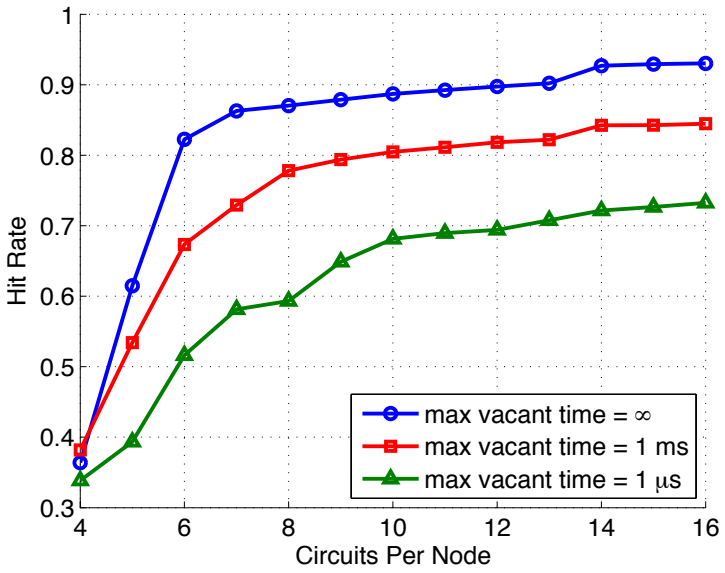


Figure 4.8: Circuit hit rate (geometric mean of all benchmarks except LULESH) when maximum vacant time is set to infinity, 1 ms and 1 μs.

distance is larger than an allowed maximum vacant time (MVT). Such a method can provide more energy savings than waiting until a circuit's decay time reaches MVT.

The change in hit rate with respect to MVT is presented in Fig. 4.8, where the MVT is set to infinity, 1 ms and 1 μs, respectively. The energy consumption of the circuit set, however, may not necessarily drop as MVT becomes tighter. If the circuit set size is small (e.g. 6 per node, left of Fig. 4.9), PTO could lead to counter effects on energy consumption. For example, the energy consumption of miniMD and HPCCG increases as MVT shrinks. The reason is that too eager PTO creates more misses, and more energy is consumed during miss penalty periods. However, if the circuit set size is relatively large (e.g. 16 per node, right of Fig. 4.9) – in which case the circuit resource might be overprovisioned – PTO adaptively shuts down the excessive resources and the energy consumption is reduced.

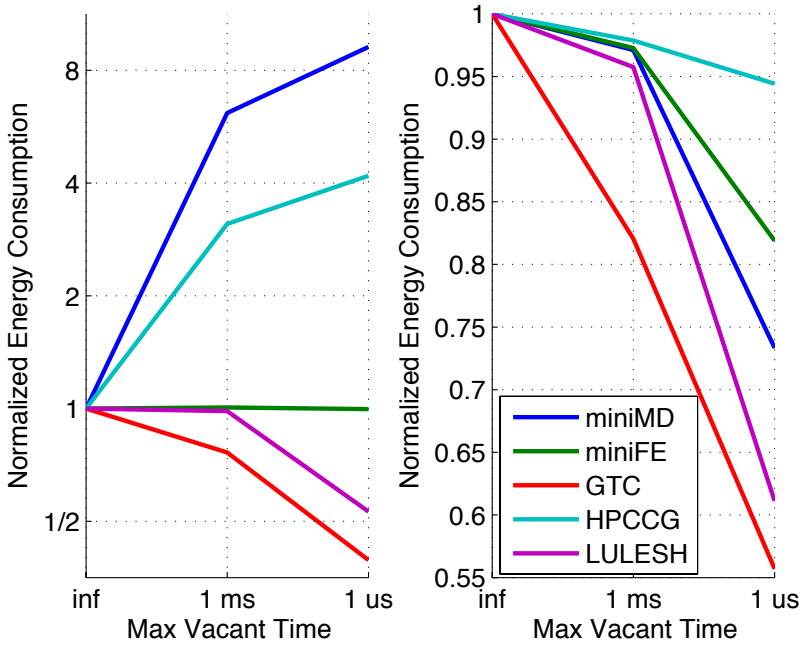


Figure 4.9: Energy consumption of circuits versus maximum vacant time. Left: max circuits per node = 6, Right: max circuits per node = 16. All energy values are normalized to the infinite-vacant-time case.

4.8 Experimental Demonstration

We demonstrate dynamic reconfiguration of optical circuits using state-of-the-art silicon photonic devices interfaced to high-speed FPGAs. We construct a 20 Gbps wavelength division multiplexed (WDM) optical network that can be rapidly reconfigured using a silicon photonic switch driven by the FPGA, and then wavelength filtered using a silicon photonic demultiplexing device. We characterize all latencies involved in network reconfiguration. The motivation is to provide parameters for the circuit reuse design optimization.

Experimental Setup

The experimental setup is shown in Fig. 4.10. An Altera Stratix V GT Signal Integrity Kit FPGA is used to generate PRBS $2^3 1 - 1$ data at 10 Gbps. The data

is modulated on two DFB laser outputs (1550 nm and 1552 nm) using commercial LiNbO3 modulators and combined using a 50:50 passive optical splitter. The data is then amplified and launched onto a silicon photonic Mach-Zehnder interferometer (MZI)-based 2x2 switch. A second Altera FPGA drives a 40 mVpp, 12 ns-period square wave having a DC offset of 92.2 mV to change the switch between the cross and bar states. The output of the switch is sent to a microring based demultiplexer (demux) for wavelength filtering. The filtered wavelengths are then amplified for data reception through PIN/TIA optical-to-electrical converters and 12.5 GHz limiting amplifiers interfaced directly to another Stratix V FPGA. The 2x2 MZI switch was fabricated through the OpSIS multi-project-wafer foundry service and features both thermal and fast P-I-N electrical switching functionality. The switch is capable of 15 dB cross-bar port extinction ratio and has a fast switch speed of 2 μs [121]. The demux device was fabricated at the Cornell Nanofabrication Facility on a standard silicon-on-insulator (SOI) platform and contains localized heaters for thermal tuning. The device has a measured extinction ratio of 15 dB and a thermal time constant of 4 μs.

Experimental Results

Fig. 4.11 show the temporal response of the MZI switch simultaneously switching two wavelengths. We show optical eye patterns for $\lambda 1$ and $\lambda 2$ passing through each output state and after filtering by the demultiplexing filter. This demonstration shows a 1.0 ns rise time and 2.2 ns fall time, which is the fastest time achieved yet for OpSIS MZI switch devices. This optical response shown in the figure is the result of the aforementioned 12 ns-period digital square wave used for switching. The electrical rise and fall times are 144 ps and 256 ps, respectively. Fig. 4.12(a-c) shows the optical switching properties of the OpSIS 2x2 MZI according to the aforementioned electrical control signal. Fig. 4.12d shows the thermal tuning and stabilization time

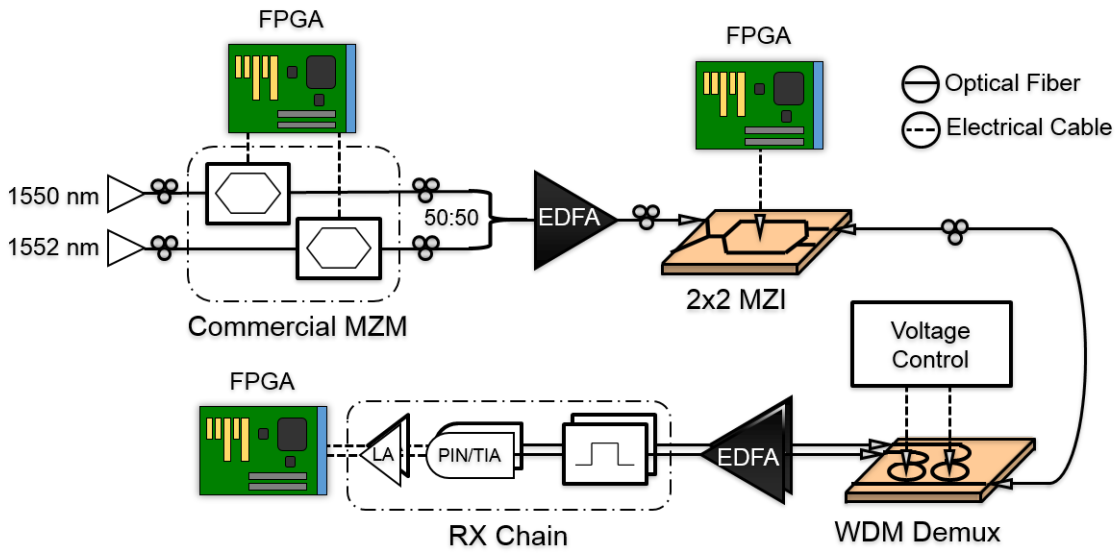


Figure 4.10: Experimental setup for dynamic WDM circuit reconfiguration (Only one switch to demultiplexer path is shown).

for the demux ring filter for a variety of wavelength offsets. The time is on the x-axis and shows stabilization time on the order of 200 μs for a large wavelength offset. This initialization time is added each time the device is not locked to its operation wavelengths and each time wavelength assignments change. For the temporal figures shown previously the thermal tuning is assumed to be complete and unchanged.

Hardware description language (HDL) was used to implement state-based logic, which counted execution times of essential steps in PHY initialization and synchronization logic. PHY initialization requires time to setup electrical transmit and receive component, such as phase-locked-loops and shift registers. We measure an average of 2.635 ms for the PHYs driving each optical datapath. Optical errors are not reflected in the ultimate data delivery due to adaptive equalization of receiver components in the PHY; however, optical errors are reflected in the word alignment process of PHY initialization. We implement a syncword-based word alignment that relies on successful delivery of 5 successive syncwords before the link is available for data transmission. We measure an average synchronization time of 1.2 μs – after ini-

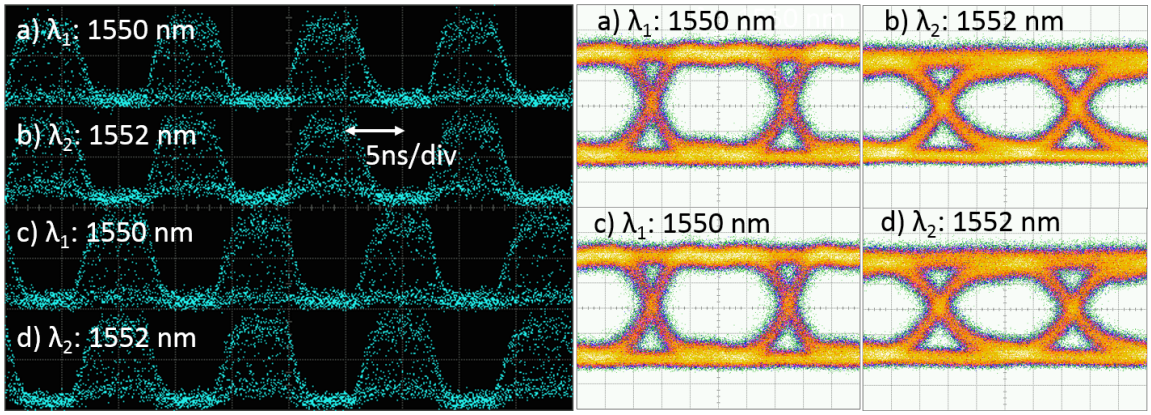


Figure 4.11: Left: optically-switched WDM data: (i) 1550 nm and (ii) 1552 nm through one path of the 2x2 MZI switch; (iii) 1550 nm and (iv) 1552 nm through the other path of the switch. Right: optical eye patterns of modulated data.

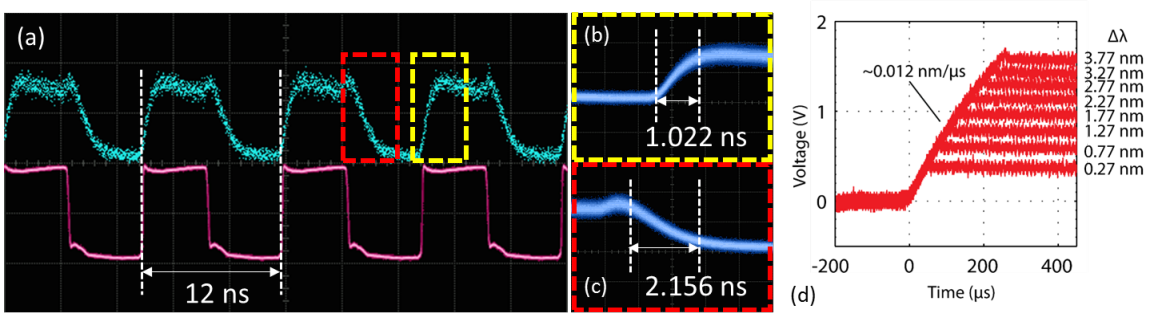


Figure 4.12: (a) Optical circuit switching latencies detected using a high speed digital communications analyser. The bottom waveform is the electrical driving signal, and the top waveform is the optical output of the switch. (b-c) Rise and fall times measrued from 10-90%, the fall time is slower because of free-carrier lifetime. (d) demux thermal wavelength locking latencies. Time is on the x-axis and a set of wavelength shifts is located on the right. The ramp waveform shows the time it takes for the output heater voltage to stabilize to the wavelength offsets.

tialization – for data delivery over each optical data path. We demonstrate successful delivery of 5×10^12 bits (5 Tb) of PRBS data consecutively on each optical circuit without error. The experimental results show combined latency characteristics of the link initialization process. Faster PHY initialization times are possible using commercial ASICs. Considering the SiP circuit setup latencies, however, we still see the need for circuit reuse design optimization.

4.9 Summary

In this work, we study architectural solutions for avoiding the setup penalty of silicon photonic circuits. The investigation of circuit reuse distances based on HPC benchmarks provides evidence for the temporal locality of circuit requests and the opportunity to amortize setup overheads. Inspired by previous cache optimization techniques, we investigate the performance of reuse distance based circuit replacement techniques. The proposed Transtion Matrix Based Predictor is shown to provide much higher prediction accuracy than previous maximum likelihood prediction for HPC communications. Based on the reuse distance prediction, the two replacement policies – Farthest Next Use and Minimum Reuse Score – also effectively increase the circuit hit rate compared to the LRU policy and hence avoid the setup penalty.

Our future work will include a comprehensive evaluation of application performance improvement and energy savings when using the proposed approach. Specifically, we will focus on how the improvement of circuit hit rates could translate into application runtime speedup. We will also study methods for determining the optimal maximum vacant time (MVT) in order to optimize the performance-energy trade-off.

APPENDIX

miniMD, miniFE and HPCCG are developed by the Mantevo project [38]. miniMD is a proxy application for molecular dynamics simulations. It has a single kernal with a few AllReduce collectives [122]. HPCCG is a simple conjugate gradient benchmark and "intended to be the best approximation to an unstructured implicit finite element or finite volume application in 800 lines or fewer" [38]. miniFE is also an proxy application for unstructured implicit finite element codes, but with complete computation steps. LULESH (Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics) intends to mimic computation of hydrodynamic simulations [123].

LULESH allows perfect weak scaling over distributed architectures [124]. GTC (Gyrokinetic Toroidal Code) solves a set of non-linear partial differential equations and is extensively used for fusion energy research. It has a "toroidal" communication pattern and is further described in [35].

Chapter 5

Prefetching Optical Circuit for Further Latency Avoidance

This chapter develops an application-guided circuit management technique that further hides path setup latency. By learning the temporal locality and communication patterns from upper-layer applications, the technique not only caches a set of circuits to maximize reuse, but also prefetches predicted circuits to actively hide the setup latency. We apply the technique to communication patterns from a spectrum of science and engineering applications. The results show that setup delays via circuit misses are significantly reduced, showing how the proposed technique can improve circuit switching in HPC optical interconnects.

5.1 Motivation

In last chapter, we describe a cache-cached technique that amortizes path setup delays by reusing optical circuits. This technique, however, relies on circuit misses to update the cached circuit set – that is, the architecture does not replace a circuit until a miss occurs. This reactive strategy thus constitutes a drawback: if a required circuit is not yet in the cache set, its setup latency will never be hidden.

To address the reactive replacement problem, this chapater further proposes an active circuit prefetching approach based on the circuit-cached scheme. By prefetching circuits before real requests arrive, the approach can further reduce or even eliminate the setup delays. Specifically, an application-specific predictor is proposed to learn characteristic *predecessor-follower* destination patterns in an application's communication behavior. Simulation based on a broad spectrum of benchmarks shows that the proposed caching-plus-prefetch scheme significantly enhance the hit rate performance compared to the previous caching-only scheme (by a rate increase as large as 95%).

5.2 Circuit Prefetch Using Application-specific Predictors

Accurate prediction of incoming circuit requests is critical to efficient circuit prefetching. The prediction can be made by learning the communication behavior of an upper-running application. In this paper, we consider the learning process in an online fashion and use as learning material the destination sequence generated by a network end point, for example, an application rank. In particular, we are interested in extracting characteristic *predecessor-follower* patterns from the destination sequence. These patterns, repeated due to workload iterations, can provide useful information regarding which circuit would be requested after the current one, thus enhancing the prediction accuracy.

Fig. 5.1, for example, shows a sequence of destinations addressed by one rank of a parallel *Adaptive Mesh Refinement* (AMR) application along the time axis. In this example, circuit requests towards destinations **84**, **55** and **114** often follow the request towards destination **90**. With knowledge of such a *follower* pattern, a circuit management runtime can prefetch the most probable *follower* circuits when still

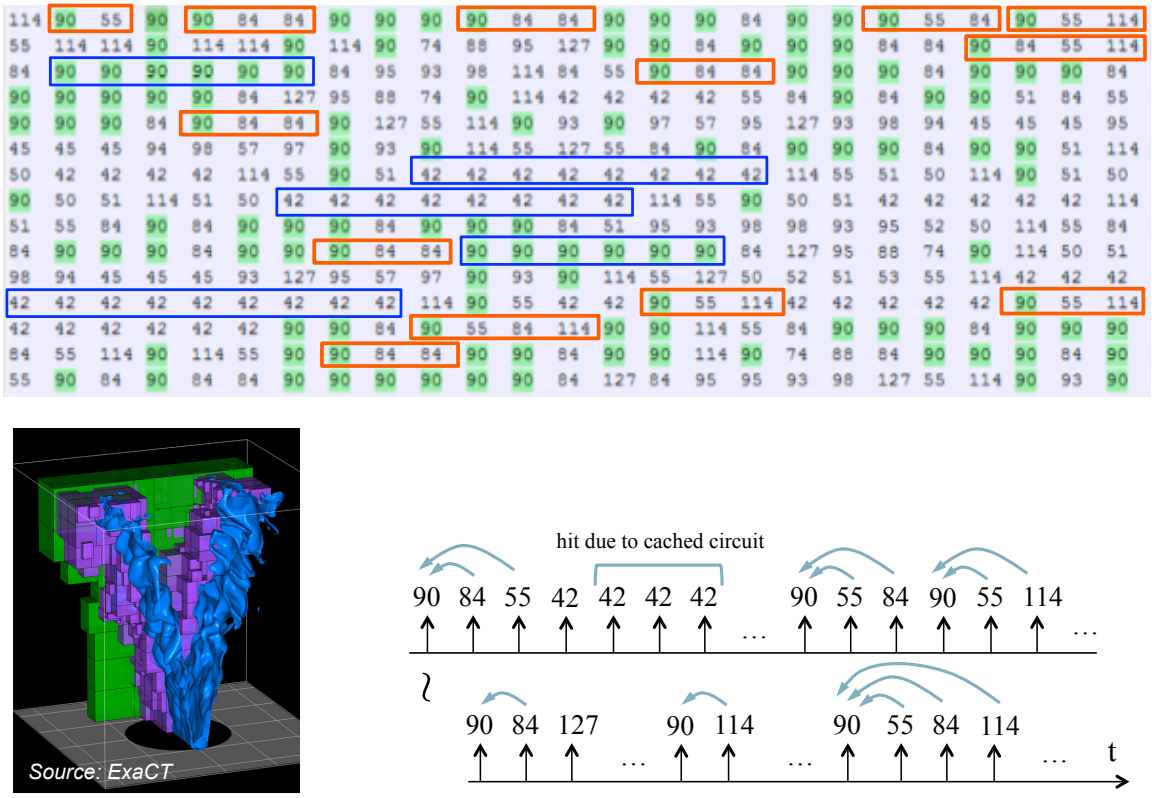


Figure 5.1: A sequence of destinations that a rank of a parallel *Adaptive Mesh Refinement* application communicates to. The *blue* arrows indicates characteristic *predecessor* (**90**) - *follower* (**84**, **55**, **114**) patterns.

communicating with the current destination, hiding the setup delay from the application. It should be noted that although a caching-only approach without prefetching can also achieve circuit hits when there are recurring communication requests – for example, the repeated requests to **42** in Fig. 5.1 – it misses the *follower* pattern that could further reduce the setup latency.

Predicting the follower circuits is an important step of effective prefetching. Several prediction techniques have been proposed in the realm of cache management [125]–[127]. These techniques, however, are designed for optimizing memory accesses, which often have constant access strides in the address space (e.g. when accessing an array in the memory). Hence, a prediction is often made by adding or subtracting a constant stride to or from the current address. The circuit communication considered

in this paper, by contrast, does not always have a constant "stride" in the destination ID sequence. Such a lack-of-stride feature is even more common for applications with irregular communication patterns. Therefore, conventional stride-based cache prefetch techniques may not work for the circuit communication scenario considered in this paper.

To solve the above problem, the circuit prefetch runtime proposed in this work uses a *lookup table* (LUT) to learn the characteristic follower patterns. The LUT, maintained by each node, uses a *predecessor* ID as an entry's key, and the corresponding *followers'* ID (with their repeat frequency) as the entry's value. By observing the local communication history of an application rank, the node constantly updates the LUT to record the k most frequent *followers* of each *predecessor*. Here, we call parameter k the *tail length*, the maximum number of *follower* circuits for a given *predecessor* circuit. In Fig. 5.1, the predecessor **90** has three followers **55**, **84**, **144**, giving a tail length $k = 3$. The tail length impacts the actuation costs induced by prefetching, including the power consumed by circuit setup. The tail length also determines the size of the LUT and the update complexity. Thus, the tail length k imposes a trade-off effect in circuit prefetching: increasing k may result in more prefetched circuits and hence a potentially higher hit rate, but it may also induce more actuation energy consumption, a larger LUT footprint and higher update complexity. We will analyze such trade-offs in the next section.

When a circuit request arrives and is recognized as a recorded *predecessor*, its correspondent follower circuits will be prefetched by the circuit management runtime, unless the follower circuit is already cached. In this paper, we assume that the replacement policy used for prefetching is the same as that used for circuit misses. We also assume that only vacant circuits that are not in data transmission can be considered as a replacement candidate. Furthermore, upon circuit misses (hard replacement), a prefetched circuit can be preempted.

Table 5.1: Description of benchmarks used in the simulation and their communication features (numbers mearsured at 256 ranks).

Application	Description	Neighbor-to-Rank Ratio	Reuse Distance [5]
HPCCG [38]	Conjugate gradient code for 3D chimney domain simulation	4.69%	[8,16)
miniFE [38]	Unstructured implicit finite element codes	10.04%	0, [8,16)
miniMD [38], [122]	Molecular dynamics for spatial-decomposition particle simulations	5.18%	1, [8,16)
LULESH [123], [124]	Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics	17.71%	[32,64)
Multigrid [128]	Differential equations solver using a hierarchy of discretizations	14.46%	[32,64)
CNS [128]	Compressible Navier Stokes equations with constant viscosity and thermal conductivity	17.30%	[32,64)

5.3 Performance Evaluation

Methodology

To evaluate the effectiveness of the proposed prefetch scheme, we compare its performance with the caching-only scheme in Ref. [5]. In both cases, a *Least Recently Used* (LRU) replacement policy is used. We use multiple mini-apps that cover a wide spectrum of scientific computations as upper-running benchmarks. These benchmarks, representing different communication patterns, are simulated based on a non-blocking circuit-switched network implementing one of the two latency-avoiding schemes above. The applications are simulated using trace replay from a library of DUMPI traces in conjunction with the macro-scale components of the SST simulator [129]. Time gaps between consecutive communication calls are derived from

timestamps within the traces. To reflect how the application behaviors impact the effectiveness of the schemes, we assume there is only one application rank per node. In this way, the destination sequence seen by every local circuit runtime is a reflection of single-rank communication behavior. The simulation assumes a 256-node non-blocking optical network, and hence 256 ranks for the applications, except LULESH, which consists of 125 ranks due to a three-dimension decomposition requirement of the problem. Although a non-blocking optical network [130], [131] represents an ideal network scenario, in this paper we only use it as a platform to facilitate an initial comparison of different circuit management strategies. Furthermore, the proposed application-guided prefetching methodology is migratable to blocking networks and our future work will include research in this direction.

Table 5.1 provides a brief description of the benchmarks as well as information regarding their *neighbor-to-rank ratios* and *reuse distances*. Here, the *neighbor-to-rank ratio* is a ratio of the average number of communication neighbors per rank to the number of ranks, indicating a diversity of communication destinations. The *reuse distance* represents how often a destination is re-addressed by a source [5]. Consider a destination sequence from a single source s. If the number of destinations interposed in the sequence between two requests for a destination t is d, then d is the reuse distance of destination t in view of source s. Each application has a "maximum-likelihood" set of reuse distances. These reuse distances are binned logarithmically by powers of 2 in Table 5.1, showing the most common reuse distances.

Hit Rate

Fig. 5.2 shows the hit rate performance of the caching-only and the caching-plus-prefetch schemes with different tail lengths ($k = 1, 2, 3$), against various numbers of circuits per node ($p = 2, 3, ..., 8$). It should be noted that in case p is smaller than k, the circuit management runtime would fetch no more than p most probable circuits

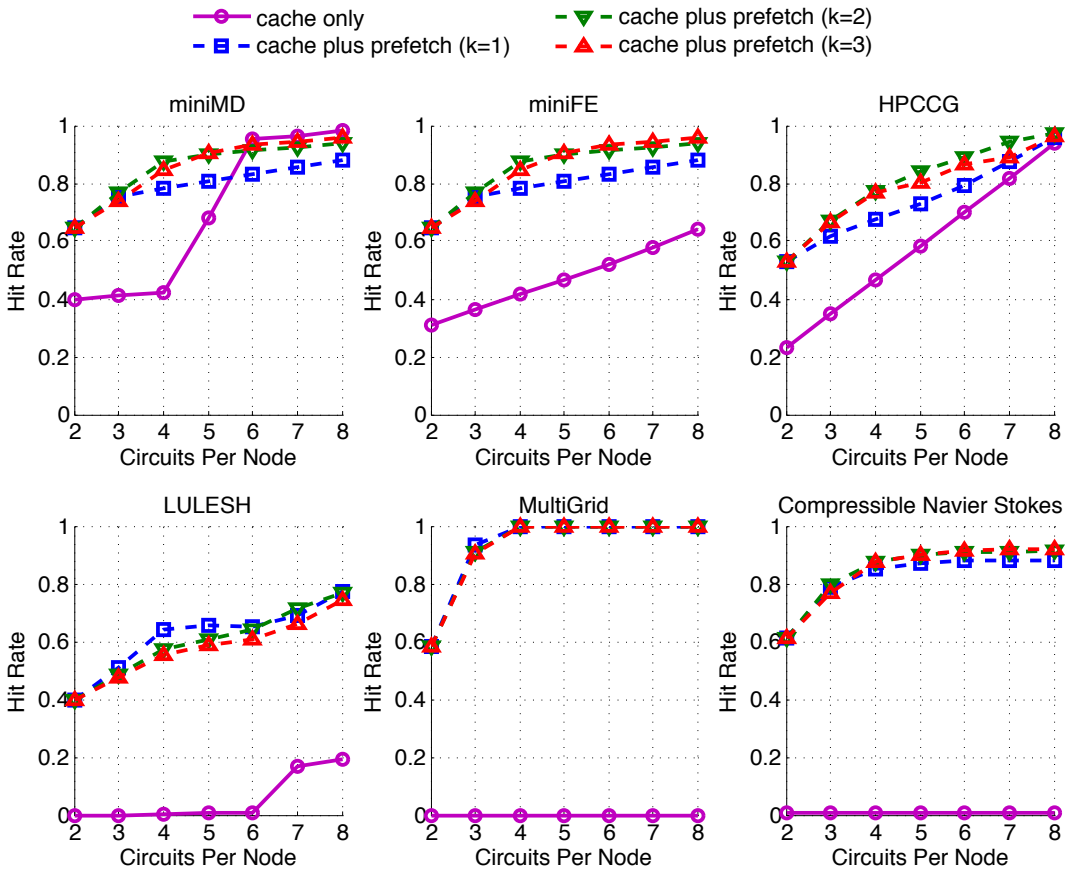


Figure 5.2: Circuit hit rates achieved by the caching-plus-prefetch scheme (*dashed*) with different tail lengths k, versus the caching-only scheme (*solid*), in a 256-rank simulation. Both schemes employ the *least recently used* replacement policy.

out of the k predicted ones.

Compared to the caching-only scheme, the caching-plus-prefetch scheme significantly increases the circuit hit rate in all of the applications. In applications such as HPCCG, miniFE and miniMD, the hit rate increase is as high as 40%. This number is even larger for LULESH (60%), Multigrid (90%) and CNS (80%). Regarding when the maximal enhancement is achieved by the prefetching scheme, a difference exists between the high-communication-degree (long-reuse-distance) and modest-communication-degree (modest-reuse-distance) applications. Here, we refer the communication degree to the average number of communication neighbors per

node, which is proportional to the neighbor-to-rank ratio listed in Table 5.1.

For modest-communication-degree (modest-reuse-distance) applications, such as HPCCG and miniMD, the maximal hit rate enhancement by the prefetching scheme occurs at small circuit provision numbers – for example, this provision number is two circuits per node for HPCCG and four for miniMD. As the number of provisioned circuits increases, the hit rate of the caching-only scheme grows fast because the "circuit cache size" becomes large enough to cover most of the modest reuse distances, narrowing the performance difference from the prefetching scheme.

For high-communication-degree (long-reuse-distance) applications, such as LULESH, Multigrid and CNS, provisioning more circuits improves the hit rate by a limited amount in the caching-only case. The reason for the limited improvement is that the number of circuits provisioned in the simulation, which is two to eight per node, is still smaller than the communication degree or the most probable reuse distances of the application. Take Multigrid for example, it has a communication degree of 37 (out of the 256 ranks) and a most probable reuse distance in the range of [32, 64), both of which are significantly larger than the maximum number of circuit provisioned. Therefore, the circuit cache size is not large enough to cover the communication degree or reuse distance, resulting in limited hit rate improvement in the caching-only case.

In comparison, the prefetching scheme does not require a large circuit cache size, a small communication degree, or a short reuse distance. The principle of the prefetching scheme merely relies on the existence of correlated communications. In this sense, the prefetch scheme is capable of creating circuit hits even when the number of circuit is small compared to the communication degree or reuse distance. See, for example, the hit rate of Multigrid and CNS in Fig. 5.2, where the prefetching scheme improves the hit rate from almost 0% to almost 100% compared to the caching-only scheme. Such capability is extremely important for a wide range of applications, 1) which by

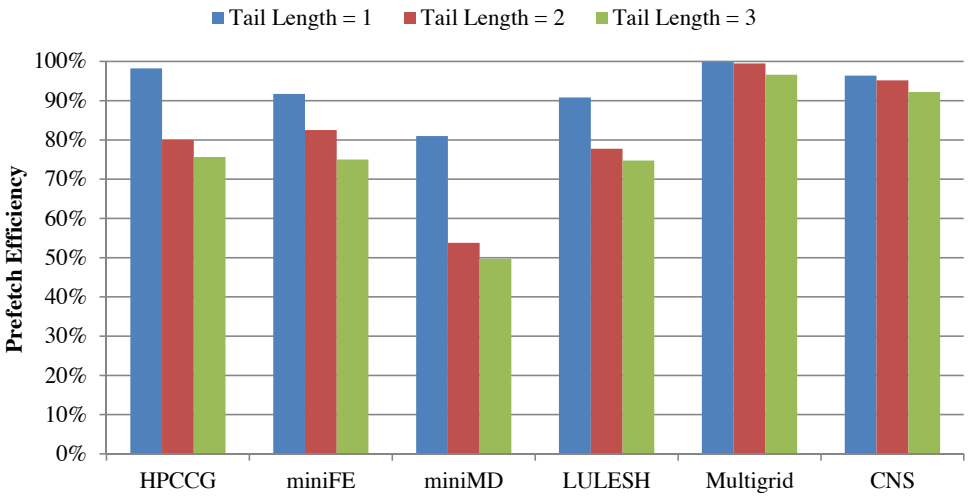


Figure 5.3: Prefetch efficiency against different tail lengths k. The prefetch efficiency is a percentage of prefetches that result in a hit, out of the total prefetches.

problem definition have a large communication degree, or 2) whose communication degree increases significantly as the parallelism degree scales. This also means that the prefetching scheme is more scalable than the caching-only scheme.

Tradeoff of Tail Length

The tail length parameter determines the number of *followers* to be learned by the predictor and the number of circuits to be prefetched when a *predecessor* is recognized. Although a longer tail length, namely, more prefetches, can generally lead to a higher hit rate, as Fig. 5.2 shows, it can also increase the reconfiguration cost. We investigate such a tradeoff by evaluating the *prefetch efficiency* under different tail lengths. The *prefetch efficiency* is a percentage of prefetches resulting into a hit out of the total number of prefetches. Fig. 5.3 captures the prefetch efficiency when the number of circuits per node is three. While the efficiency is in general high for all applications when the tail length is 1, the result also shows that increasing the tail length may decrease the prefetch efficiency, by different extents and based on the applications. Such decrease depends on the length of typical *predecessor-follower* patterns in the

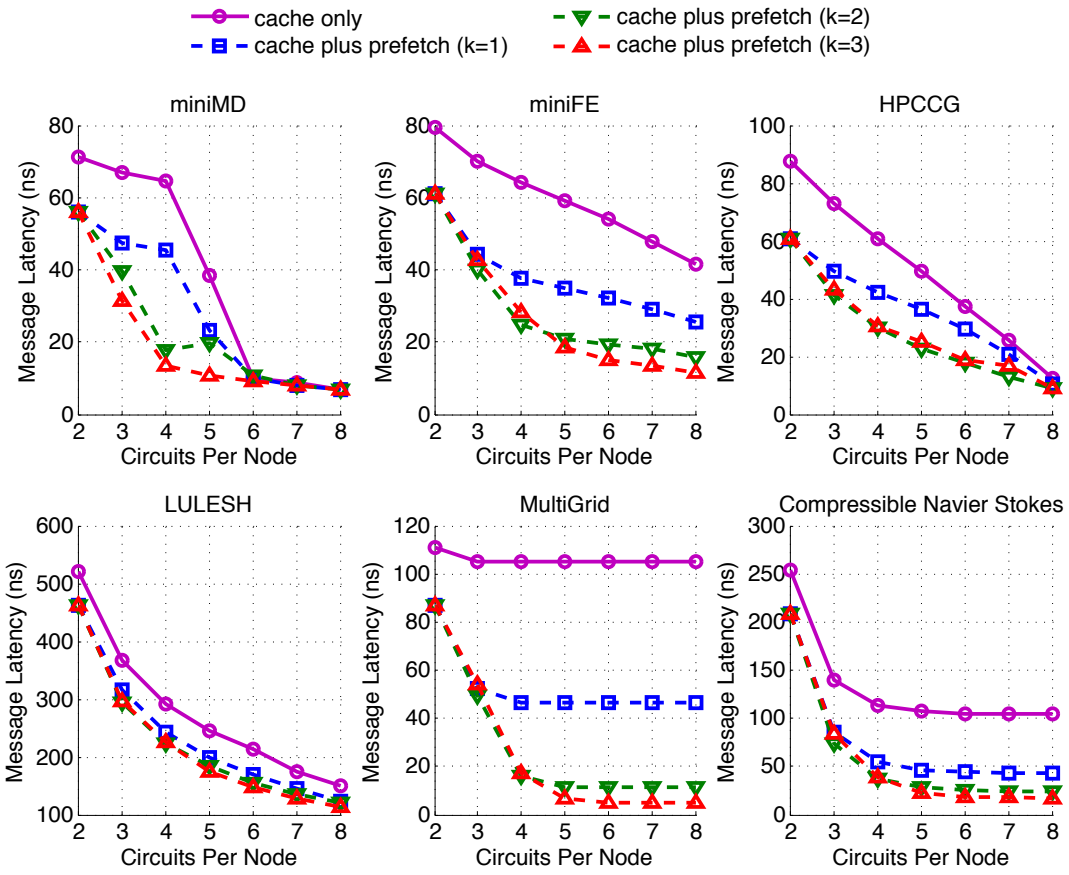


Figure 5.4: Message latency of caching-plus-prefetch scheme (*dashed*) with different tail lengths k, versus the caching-only scheme (*solid*), in a 256-rank simulation. Both schemes use the *least recently used* replacement policy. Circuit setup latency is 100 ns, circuit bandwidth is 100 Gb/s.

application, as well as the repeat frequency of the learned patterns defined by the selected tail length.

Latency

Message latencies of the two approaches are compared in Fig. 5.4. The simulation assumes a reconfiguration latency of 100 ns and a link bandwidth of 100 Gb/s. The reconfiguration latency assumed includes the time to release the old circuit (tens of ns), the time to switch the lightpath (tens of ns [132]), and the time for the hand-shake of the new circuit (tens of ns). As the results show, the prefetch approach

significantly reduces the average message latency thanks to a reduced missed rate. In applications such as miniFE and miniMD, a maximal latency reduction of 65% is achieved (at four circuits per node). In Multigrid, an even higher reduction of 90% is achieved (at five circuits per node). These results show that the proposed prefetching scheme can effectively hide the setup latency from application-oriented message communications.

We expect that the above latency-hiding capability would be especially useful to small-message communication in high-bandwdith optical networks. Consider a small message of 1KB, its transmission time is $1024 \times 8/100 = 81.92$ ns under the assumed link bandwidth (100Gb/s). Such a transmission time is even shorter than the circuit reconfiguration time, which makes hiding the reconfiguration latency even more important for the small-message case.

5.4 Summary and Discussion

In this chapter, we build on the circuit-cached architecture and develop a circuit prefetch approach, to actively avoid the circuit setup latency. We show a prefetch predictor that learns application-specific *predecessor-follower* patterns from the destination sequence, with a variable control on the length of such patterns. Simulation results based on a wide spectrum of scientific applications, which represent various communication degrees and circuit reuse distances, show that the prefetching scheme significantly improves the circuit hit rate over the caching-only scheme. The prefetching scheme is also shown to effectively hide the setup latency from application communication, which capability is important for small-message scenarios.

The current work is only an introductory exploration of reconfiguration in optical interconnection networks providing end-to-end communication circuits. Many issues deserve further study. A high circuit hit rate may not directly translate into a short

execution time for the application. More detailed system-level simulations at larger scale could further explore the performance impacts, particularly under more detailed circuit NIC and switch contention models. The current work does not follow the popular MPI-everywhere model with one rank per core. Future study could explore the tradeoffs involved in increasing the number of MPI ranks per node. However, the current work should be considered part of a co-design process, exploring how new technologies might tip the design point for balancing distributed- and shared-memory parallelism.

Chapter 6

Conclusion and Remarks

In this thesis, a lesson learned is that many innovations and exciting research exist in the intersection of two or multiple areas. One can design a photonic interconnection network purely based on random traffic generation. However, such designs may not bring specific benefits to the applications of interest. Architecture and application co-design is what we need for achieving exascale.

The circuit-switched nature of photonics determines that it may not work as a standalone, generalized solution. The essential difference between a circuit-switched network and a packet-switched one – while many people believe is in the switching granularity – lies in whether there is pipelining. The DataVortex network [133]–[135] is a success because it is a pipelined optical design. Unfortunately, the number of fibers (working as buffers) in DataVortex is beyond the sustainable number for a large-scale network. We can, instead, design for a balance between optical and electrical switching and let them do what they are good at. Flexfly is an example. In this regard, the role of photonic switching in HPC can be to enable reconfigurable topologies, or to create dynamic locality, as we have shown in both macroscale and microscale (Chapters 2 and 3, respectively).

Of course, the balance point may shift based on technology and cost changes.

These changes, in future, may include: i) photonic transceivers (bandwidth) becomes cheaper and cheaper, ii) electronic switches cannot sustain the ever-increasing switching rate or iii) the bandwidth out of a single node to the even first switching level exceeds the physical limit of copper wires. These factors may gradually drive photonics into lower and lower hierarchies of a network. At a point where photonics is used node-to-node or where centralized control is no longer possible, the path setup delay should be carefully managed. The circuit-cached and circuit-prefetched schemes developed in this thesis are early explorations in this direction. We believe that the application-level characteristics, as explored in Chapters 4 and 5, can help us relax the design requirement and obtain better networks.

Chapter 7

Future Work Recommendations

7.1 On Reconfigurable Networks

On the topic of reconfigurable networks, the following research ideas are suggested:

Using Flexfly in a multi-job environment:

While Chapter 2 only investigates benefit of Flexfly for a single application spanning over the entire machine, the benefit of Flexfly for a multi-job workload is also obvious. Since network partitions for different jobs will have no communication between each other, the inter-partition links can be wired back to their own partition through photonic switching. Here, each network partitions can contain several Dragonfly groups, hence those inter-partition links, being global links, can be used for intensive group pairs inside a partition.

Integrating topology reconfiguration into job schedulers:

The job scheduler is in charge of node allocation and job placement in supercomputers. Hence the scheduler may have knowledge of, or predict with high accuracy, the future traffic pattern in the supercomputer network. The scheduler can then reconfigure the

topology, using the bandwidth steering functionality, to "carve" out a partition for the job to be launched.

Co-optimizing topology reconfiguration and task mapping:

Topology-aware task mapping (TATM) are prevalent in optimizing communication locality and avoiding network congestions. For the Dragonfly topology alone, there proposed numerous mapping approaches in literature [29], [136], including linear, blocked node, blocked supernode, random, hybrid, etc. Designed for the same goal, TATM and Flexfly represent a top-down and a bottom-up approach, respectively, and there is an obvious inter-dependency between task mapping and topology reconfiguration. A joint optimization of the two may lead to even better performance.

Two-level reconfigurability enabled by spatial and wavelength switching:

Photonic switching works in both spatial and wavelength domains. By coupling reconfigurable wavelength demultiplexers to an optical spatial switch (so called "broadband switches" such as MZI), two-level switching is possible. Although this kind of switch is a blocking type, it maps well with two-level networks such as Dragonfly. There is hence a chance to achieve both inter- and intra- group reconfigurability at the same time.

7.2 On Setup Delay Avoidance

On the topic of setup delay avoidance, the following research ideas are suggested:

Investigating "MPI-everywhere" model:

The MPI-everywhere model assumes one rank per core, while the present thesis assumes one rank per node. Future study could explore the tradeoffs involved in in-

creasing the number of MPI ranks per node, exploring how new technologies like photonic circuits might tip the design point for balancing distributed- and shared-memory parallelism.

Leveraging knowledge of asynchronous task runtimes:

Asynchronous task runtimes like DHARMA [137] may look ahead for future communications due to task movements. Such knowledge, available ahead of time, is perfect for guiding circuit prefetches. This synergy also brings a rationale for integrating circuit communication runtime with task runtime.

Bibliography

[1] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ACM, 2009, p. 41.

[2] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, *et al.*, "The percs high-performance interconnect," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, IEEE, 2010, pp. 75–82.

[3] M. Besta and T. Hoefler, "Slim fly: a cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Press, 2014, pp. 348–359.

[4] S. Rumley, D. Nikolova, R. Hendry, Q. Li, D. Calhoun, and K. Bergman, "Silicon photonics for exascale systems," *Journal of Lightwave Technology*, vol. 33, no. 3, pp. 547–562, 2015.

[5] K. Wen, D. Calhoun, S. Rumley, X. Zhu, Y. Liu, L. W. Luo, R. Ding, T. B. Jones, M. Hochberg, M. Lipson, *et al.*, "Reuse distance based circuit replacement in silicon photonic interconnection networks for hpc," in *High-Performance Interconnects (HOTI), 2014 IEEE 22nd Annual Symposium on*, IEEE, 2014, pp. 49–56.

[6] P. Kogge and J. Shalf, "Exascale computing trends: adjusting to the new normal for computer architecture," *Computing in Science & Engineering*, vol. 15, no. 6, pp. 16–26, 2013.

[7] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[8] P. Colella, "Defining software requirements for scientific computing," 2004.

[9] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, *et al.*, "The landscape of parallel computing research: a view from berkeley," Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. Rep., 2006.

[10] S. Kamil, J. Shalf, L. Oliker, and D. Skinner, "Understanding ultra-scale application communication requirements," in *IEEE International. 2005 Proceedings of the IEEE Workload Characterization Symposium, 2005.*, IEEE, 2005, pp. 178–187.

[11] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quema, and M. Roth, "Traffic management: a holistic approach to memory placement on numa systems," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '13, Houston, Texas, USA: ACM, 2013, pp. 381–394, ISBN: 978-1-4503-1870-9. DOI: 10.1145/2451116.2451157. [Online]. Available: http://doi.acm.org/10.1145/2451116.2451157.

[12] C. Gunn, "Cmos photonics for high-speed interconnects," *Micro, IEEE*, vol. 26, no. 2, pp. 58–66, 2006, ISSN: 0272-1732. DOI: 10.1109/MM.2006.32.

[13] G. T. Reed, G. Mashanovich, F. Gardes, and D. Thomson, "Silicon optical modulators," *Nature photonics*, vol. 4, no. 8, pp. 518–526, 2010.

[14] A. Shacham, B. Lee, and K. Bergman, "A scalable, self-routed, terabit capacity, photonic interconnection network," in *High Performance Interconnects, 2005. Proceedings. 13th Symposium on*, 2005, pp. 147–150. DOI: 10.1109/CONECT.2005.6.

[15] A. Shacham and K. Bergman, "Building ultralow-latency interconnection networks using photonic integration," *IEEE Micro*, vol. 27, no. 4, pp. 6–20, 2007, ISSN: 0272-1732. DOI: http://doi.ieeecomputersociety.org/10.1109/MM.2007.64.

[16] A. Shacham, B. A. Small, O. Liboiron-Ladouceur, and K. Bergman, "A fully implemented 12 ? 12 data vortex optical packet switching interconnection network," *J. Lightwave Technol.*, vol. 23, no. 10, p. 3066, Oct. 2005.

[17] O. Liboiron-Ladouceur, A. Shacham, B. A. Small, B. G. Lee, H. Wang, C. P. Lai, A. Biberman, and K. Bergman, "The data vortex optical packet switched interconnection network," *J. Lightwave Technol.*, vol. 26, no. 13, pp. 1777–1789, Jul. 2008.

[18] X. Ye, Y. Yin, S. J. B. Yoo, P. Mejia, R. Proietti, and V. Akella, "Dos: a scalable optical switch for datacenters," in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '10, La Jolla, California: ACM, 2010, 24:1–24:12, ISBN: 978-1-4503-0379-8. DOI: 10.1145/1872007.1872037.

[19] B. Lee, A. Biberman, P. Dong, M. Lipson, and K. Bergman, "All-optical comb switch for multiwavelength message routing in silicon photonic networks," *Photonics Technology Letters, IEEE*, vol. 20, no. 10, pp. 767–769, 2008, ISSN: 1041-1135. DOI: 10.1109/LPT.2008.921100.

[20] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *ACM SIGARCH Computer Architecture News*, IEEE Computer Society, vol. 36, 2008, pp. 77–88.

[21] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Cost-efficient dragonfly topology for large-scale systems," in *Optical Fiber Communication Conference*, Optical Society of America, 2009, OTuI2.

[22] 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*, IEEE Computer Society, 2007, pp. 172–182.

[23] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Analyzing network health and congestion in dragonfly-based supercomputers," in *Proceedings of the International Parallel and Distributed Processing Symposium*, ser. IDPDS '16, Chicago, Illinois: IEEE Press, 2016.

[24] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *ACM SIGARCH Computer Architecture News*, ACM, vol. 37, 2009, pp. 220–231.

[25] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kale, "Maximizing throughput on a dragonfly network," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Press, 2014, pp. 336–347.

[26] G. Michelogiannakis, N. Jiang, D. Becker, and W. J. Dally, "Channel reservation protocol for over-subscribed channels and destinations," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ACM, 2013, p. 52.

[27] N. Jiang, D. U. Becker, G. Michelogiannakis, and W. J. Dally, "Network congestion avoidance through speculative reservation," in *High Performance Com-*

puter Architecture (HPCA), 2012 IEEE 18th International Symposium on, IEEE, 2012, pp. 1–12.

[28] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero, "Efficient routing mechanisms for dragonfly networks," in *Parallel Processing (ICPP), 2013 42nd International Conference on*, IEEE, 2013, pp. 582–592.

[29] A. Bhatele, N. Jain, W. D. Gropp, and L. V. Kale, "Avoiding hot-spots on two-level direct networks," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, 2011, p. 76.

[30] M. García, E. Vallejo, R. Beivide, M. Valero, and G. Rodríguez, "Ofar-cm: efficient dragonfly networks with simple congestion management," in *High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on*, Aug. 2013, pp. 55–62. DOI: 10.1109/HOTI.2013.16.

[31] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, and T. Hoefler, "Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, ACM, 2014, pp. 129–140.

[32] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: performance degradation due to nearby jobs," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ACM, 2013, p. 41.

[33] B. Alverson, E. Froese, L. Kaplan, and D. Roweth. (). Cray xc series network, [Online]. Available: http://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf.

[34] K. Madduri, K. Z. Ibrahim, S. Williams, E.-J. Im, S. Ethier, J. Shalf, and L. Oliker, "Gyrokinetic toroidal simulations on leading multi-and manycore hpc systems," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, 2011, p. 23.

[35] K. Z. Ibrahim, K. Madduri, S. Williams, B. Wang, S. Ethier, and L. Oliker, "Analysis and optimization of gyrokinetic toroidal simulations on homogenous and heterogenous platforms," *International Journal of High Performance Computing Applications*, vol. 27, no. 4, pp. 454–473, 2013.

[36] I. Ivanov, J. Gong, D. Akhmetova, I. B. Peng, S. Markidis, E. Laure, R. Machado, M. Rahn, V. Bartsch, A. Hart, and P. Fischer, "Evaluation of parallel communication models in nekbone, a nek5000 mini-application," in *Cluster*

Computing (CLUSTER), 2015 IEEE International Conference on, Sep. 2015, pp. 760–767. DOI: 10.1109/CLUSTER.2015.131.

[37] I. Karlin, A. Bhatele, J. Keasler, B. L. Chamberlain, J. Cohen, Z. DeVito, R. Haque, D. Laney, E. Luke, F. Wang, D. Richards, M. Schulz, and C. Still, "Exploring traditional and emerging parallel programming models using a proxy application," in *27th IEEE International Parallel & Distributed Processing Symposium (IEEE IPDPS 2013)*, Boston, USA, May 2013.

[38] M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich, "Improving performance via mini-applications," *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, vol. 3, 2009.

[39] J. Bell, A. Almgren, V. Beckner, M. Day, M. Lijewski, A. Nonaka, and W. Zhang, "Boxlib user's guide," Technical Report, CCSE, Lawrence Berkeley National Laboratory. Available at: https://ccse. lbl. gov/BoxLib/BoxLibUsersGuide. pdf, Tech. Rep., 2012.

[40] Y. Xia, M. Schlansker, T. S. E. Ng, and J. Tourrilhes, "Enabling topological flexibility for data centers using omniswitch," in *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, Santa Clara, CA: USENIX Association, Jul. 2015. [Online]. Available: https://www.usenix.org/conference/hotcloud15/workshop-program/presentation/xia.

[41] C. Minkenberg, G. Rodriguez, B. Prisacari, L. Schares, P. Heidelberger, D. Chen, and C. Stunkel, "Performance benefits of optical circuit switches for large-scale dragonfly networks," in *Optical Fiber Communication Conference*, Optical Society of America, 2016, W3J–3.

[42] M. Prais and C. C. Ribeiro, "Reactive grasp: an application to a matrix decomposition problem in tdma traffic assignment," *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 164–176, 2000.

[43] A. Singh, "Load-balanced routing in interconnection networks," PhD thesis, Stanford University, 2005.

[44] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls, and B. Jacob, "The structural simulation toolkit," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 4, pp. 37–42, Mar. 2011, ISSN: 0163-5999. DOI: 10.1145/1964218.1964225.

[45] J. J. Wilke, K. Sargsyan, J. P. Kenny, B. Debusschere, H. N. Najm, and G. Hendry, "Validation and uncertainty assessment of extreme-scale hpc simulation through bayesian inference," in *Proceedings of the 19th International Con-*

ference on Parallel Processing, ser. Euro-Par'13, Aachen, Germany: Springer-Verlag, 2013, pp. 41–52, ISBN: 978-3-642-40046-9. DOI: 10.1007/978-3-642-40047-6_7.

[46] (). Characterization of the doe mini-apps, [Online]. Available: http://portal.nersc.gov/project/CAL/doe-miniapps.htm.

[47] (). Opsis, [Online]. Available: http://opsisfoundry.org/.

[48] G. M. Masson and B. W. Jordan, "Generalized multi-stage connection networks," *Networks*, vol. 2, no. 3, pp. 191–209, 1972, ISSN: 1097-0037. DOI: 10.1002/net.3230020303. [Online]. Available: http://dx.doi.org/10.1002/net.3230020303.

[49] C. P. Chen, X. Zhu, Y. Liu, K. Wen, M. S. Chik, T. Baehr-Jones, M. Hochberg, and K. Bergman, "Programmable dynamically-controlled silicon photonic switch fabric," *Journal of Lightwave Technology*, vol. PP, no. 99, pp. 1–1, 2015, ISSN: 0733-8724. DOI: 10.1109/JLT.2015.2505314.

[50] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008, ISSN: 0146-4833. DOI: 10.1145/1355734.1355746. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746.

[51] *Hpcbench, High Performance Networks Benchmarcking*. [Online]. Available: http://hpcbench.sourceforge.net/.

[52] *Iperf*. [Online]. Available: https://iperf.fr/.

[53] *OpenvSwitch*. [Online]. Available: http://openvswitch.org/.

[54] *The OpenDayLight Platform*. [Online]. Available: https://www.opendaylight.org/.

[55] *Crossproject:integration group:performance tests*. [Online]. Available: https://wiki.opendaylight.org/view/CrossProject:Integration.

[56] *Cluster SSH- Cluster Admin Via SSH*. [Online]. Available: https://github.com/duncs/clusterssh.

[57] *Cray xc40*. [Online]. Available: http://www.cray.com/sites/default/files/resources/CrayXC40Brochure.pdf.

[58] R. Ramaswami, K. Sivarajan, and G. Sasaki, *Optical networks: a practical perspective*. Morgan Kaufmann, 2009.

[59] M. Bahadori, S. Rumley, D. Nikolova, and K. Bergman, "Comprehensive design space exploration of silicon photonic interconnects," *IEEE Journal of Lightwave Technology*, 2016.

[60] N. Dupuis, B. G. Lee, A. V. Rylyakov, D. M. Kuchta, C. W. Baks, J. S. Orcutt, D. M. Gill, W. M. Green, and C. L. Schow, "Design and fabrication of low-insertion-loss and low-crosstalk broadband 2x2 mach-zehnder silicon photonic switches," *Journal of Lightwave Technology*, vol. 33, no. 17, pp. 3597–3606, 2015.

[61] N. Dupuis, B. G. Lee, A. V. Rylyakov, D. M. Kuchta, C. W. Baks, J. S. Orcutt, D. M. Gill, W. M. J. Green, and C. L. Schow, "Modeling and characterization of a nonblocking 4x4 mach–zehnder silicon photonic switch fabric," *J. Lightwave Technol.*, vol. 33, no. 20, pp. 4329–4337, Oct. 2015. [Online]. Available: `http://jlt.osa.org/abstract.cfm?URI=jlt-33-20-4329`.

[62] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," in *ACM SIGARCH Computer Architecture News*, ACM, vol. 37, 2009, pp. 429–440.

[63] C. Batten, A. Joshi, V. Stojanovć, and K. Asanović, "Designing chip-level nanophotonic interconnection networks," in *Integrated Optical Interconnect Architectures for Embedded Systems*, Springer, 2013, pp. 81–135.

[64] 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 *ACM SIGARCH Computer Architecture News*, IEEE Computer Society, vol. 36, 2008, pp. 153–164.

[65] J. Ahn, M. Fiorentino, R. G. Beausoleil, N. Binkert, A. Davis, D. Fattal, N. P. Jouppi, M. McLaren, C. M. Santori, R. S. Schreiber, *et al.*, "Devices and architectures for photonic chip-scale integration," *Applied Physics A*, vol. 95, no. 4, pp. 989–997, 2009.

[66] R. Beausoleil, J. Ahn, N. Binkert, A. Davis, D. Fattal, M. Fiorentino, N. P. Jouppi, M. McLaren, C. Santori, R. S. Schreiber, *et al.*, "A nanophotonic interconnect for high-performance many-core computation," in *Integrated Photonics and Nanophotonics Research and Applications*, Optical Society of America, 2008, ITuD2.

[67] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, *et al.*, "On the feasibility of optical circuit switching for high performance computing systems," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, 2005, p. 16.

[68] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10, New Delhi, India: ACM, 2010, pp. 339–350, ISBN: 978-1-4503-0201-2. DOI: 10.1145/1851182.1851223. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851223.

[69] N. Farrington, A. Forencich, P.-C. Sun, S. Fainman, J. Ford, A. Vahdat, G. Porter, and G. C. Papen, "A 10 us hybrid optical-circuit/electrical-packet network for datacenters," in *Optical Fiber Communication Conference*, Optical Society of America, 2013, OW3H–3.

[70] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "C-through: part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.

[71] J. Shalf, S. Kamil, L. Oliker, and D. Skinner, "Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, 2005, p. 17.

[72] P. Dong, X. Liu, S. Chandrasekhar, L. L. Buhl, R. Aroca, and Y.-K. Chen, "Monolithic silicon photonic integrated circuits for compact 100+gb/s coherent optical receivers and transmitters," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 20, no. 4, 2014.

[73] M. Yang, W. M. J. Green, S. Assefa, J. V. Campenhout, B. G. Lee, C. V. Janes, F. E. Doany, C. L. Schow, J. A. Kash, and Y. A. Vlasov, "Non-blocking 4x4 electro-optic silicon switch for on-chip photonic networks," *Optics Express*, vol. 19, 2011.

[74] L. Chen and Y.-K. Chen, "Compact, low-loss and low-power 8x8 broadband silicon optical switch," *Optics Express*, vol. 20, no. 17, pp. 18 977–18 985, 2012.

[75] B. G. Lee, A. V. Rylyakov, W. M. J. Green, S. Assefa, C. Baks, R. Rimolo-Donadio, D. M. Kuchta, M. H. Khater, T. Barwicz, C. Reinholm, S. M. Kiewra E. Shank, C. L. Schow, and Y. A. Vlasov, "Monolithic silicon integration of scaled photonic switch fabrics, cmos logic, and device driver circuits," *Journal of Lightwave Technology*, vol. 32, 2014.

[76] D. A. B. Miller, "Rationale and challenges for optical interconnects to electronic chips," *Proceedings of the IEEE*, vol. 88, no. 6, pp. 728–749, Jun. 2000, ISSN: 0018-9219. DOI: 10.1109/5.867687.

[77] S. Rumley, R. Polster, S. D. Hammond, A. F. Rodrigues, and K. Bergman, "End-to-end modeling and optimization of power consumption in hpc interconnects," in *Proc. of the 45th International Conference on Parallel Processing Workshops*, Philadelphia, PA, Aug. 2016.

[78] J. Lira, C. Molina, R. N. Rakvic, and A. González, "Replacement techniques for dynamic nuca cache designs on cmps," *The Journal of Supercomputing*, vol. 64, no. 2, pp. 548–579, 2013.

[79] M. Chaudhuri, "Pagenuca: selected policies for page-grain locality management in large shared chip-multiprocessor caches," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, IEEE, 2009, pp. 227–238.

[80] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches," in *Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium on*, IEEE, 2004, pp. 319–330.

[81] Q. Lu, C. Alias, U. Bondhugula, T. Henretty, S. Krishnamoorthy, J. Ramanujam, A. Rountev, P. Sadayappan, Y. Chen, H. Lin, *et al.*, "Data layout transformation for enhancing data locality on nuca chip multiprocessors," in *Parallel Architectures and Compilation Techniques, 2009. PACT'09. 18th International Conference on*, IEEE, 2009, pp. 348–357.

[82] M. Kandemir, Y. Zhang, J. Liu, and T. Yemliha, "Neighborhood-aware data locality optimization for noc-based multicores," in *Code Generation and Optimization (CGO), 2011 9th Annual IEEE/ACM International Symposium on*, IEEE, 2011, pp. 191–200.

[83] D. Brunina, X. Zhu, K. Padmaraju, L. Chen, M. Lipson, and K. Bergman, "10-gb/s wdm optically-connected memory system using silicon microring modulators," in *European Conference and Exhibition on Optical Communication*, Optical Society of America, 2012, Mo.2.A.5. DOI: 10.1364/ECEOC.2012.Mo.2.A.5. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=ECEOC-2012-Mo.2.A.5.

[84] T. Shiraishi, Q. Li, Y. Liu, X. Zhu, K. Padmaraju, R. Ding, M. Hochberg, and K. Bergman, "A reconfigurable and redundant optically-connected memory system using a silicon photonic switch," in *Optical Fiber Communication Conference*, Optical Society of America, 2014, Th2A.10. DOI: 10.1364/OFC.2014.

Th2A.10. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2014-Th2A.10.

[85] C. Sun, M. T. Wade, Y. Lee, J. S. Orcutt, L. Alloatti, M. S. Georgas, A. S. Waterman, J. M. Shainline, R. R. Avizienis, S. Lin, *et al.*, "Single-chip microprocessor that communicates directly using light," *Nature*, vol. 528, no. 7583, pp. 534–538, 2015.

[86] Y. Arakawa, T. Nakamura, Y. Urino, and T. Fujita, "Silicon photonics for next generation system integration platform," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 72–77, 2013.

[87] C. Kopp, S. Bernabe, B. B. Bakir, J.-M. Fedeli, R. Orobtchouk, F. Schrank, H. Porte, L. Zimmermann, and T. Tekin, "Silicon photonic circuits: on-cmos integration, fiber optical coupling, and packaging," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 17, no. 3, pp. 498–509, 2011.

[88] A. Masood, W. Bogaerts, J. Van Olmen, J. Van Aelst, D. Van Thourhout, and D. S. Tezcan, "Photonics-cmos 3d integration: copper through-silicon-via approach," in *Proc. of the 2009 Annual Symposium of the IEEE Photonics Benelux Chapter*, 2009, pp. 165–168.

[89] C. Sun, M. Georgas, J. Orcutt, B. Moss, Y.-H. Chen, J. Shainline, M. Wade, K. Mehta, K. Nammari, E. Timurdogan, *et al.*, "A monolithically-integrated chip-to-chip optical link in bulk cmos," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 828–844, 2015.

[90] J. Raring, E. Skogen, L. Johansson, M. Sysak, S. DenBaars, and L. Coldren, "Widely tunable negative-chirp sg-dbr laser/ea-modulated transmitter," *Journal of lightwave Technology*, vol. 23, no. 1, p. 80, 2005.

[91] C. Browning, K. Shi, A. D. Ellis, and L. P. Barry, "Optical burst-switched ssb-ofdm using a fast switching sg-dbr laser," *Journal of Optical Communications and Networking*, vol. 5, no. 9, pp. 994–1000, 2013.

[92] B. R. Koch, E. J. Norberg, B. Kim, J. Hutchinson, J.-H. Shin, G. Fish, and A. Fang, "Integrated silicon photonic laser sources for telecom and datacom," in *National Fiber Optic Engineers Conference*, Optical Society of America, 2013, PDP5C–8.

[93] S. Keyvaninia, G. Roelkens, D. Van Thourhout, C. Jany, M. Lamponi, A. Le Liepvre, F. Lelarge, D. Make, G.-H. Duan, D. Bordel, *et al.*, "Demonstration of a heterogeneously integrated iii-v/soi single wavelength tunable laser," *Optics express*, vol. 21, no. 3, pp. 3784–3792, 2013.

[94] T. Creazzo, E. Marchena, S. B. Krasulick, K. Paul, D. Van Orden, J. Y. Spann, C. C. Blivin, L. He, H. Cai, J. M. Dallesasse, *et al.*, "Integrated tunable cmos laser," *Optics express*, vol. 21, no. 23, pp. 28 048–28 053, 2013.

[95] K. Wen, H. Guan, D. M. Calhoun, D. Donofrio, J. Shalf, and K. Bergman, "Silicon photonic memory interconnect for many-core architectures," in *Proc. IEEE High Perform. Extreme Comput. Conf*, 2016.

[96] 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*, ACM, 2008, pp. 72–81.

[97] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: theory and practice," *IEEE transactions on knowledge and data engineering*, vol. 15, no. 3, pp. 515–528, 2003.

[98] D. Unat, T. Nguyen, W. Zhang, M. N. Farooqi, B. Bastem, G. Michelogiannakis, A. Almgren, and J. Shalf, "Tida: high-level programming abstractions for data locality management," in *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings*, M. J. Kunkel, P. Balaji, and J. Dongarra, Eds. Cham: Springer International Publishing, 2016, pp. 116–135, ISBN: 978-3-319-41321-1. DOI: 10.1007/978-3-319-41321-1_7. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-41321-1_7.

[99] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick, "Stencil computation optimization and autotuning on state-of-the-art multicore architectures," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, IEEE Press, 2008, p. 4.

[100] K. Datta, S. Williams, V. Volkov, J. Carter, L. Oliker, J. Shalf, and K. Yelick, "Auto-tuning the 27-point stencil for multicore," in *In Proc. iWAPT2009: The Fourth International Workshop on Automatic Performance Tuning*, 2009.

[101] S. Kamil, C. Chan, L. Oliker, J. Shalf, and S. Williams, "An auto-tuning framework for parallel multicore stencil computations," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, IEEE, 2010, pp. 1–12.

[102] S. Kamil, P. Husbands, L. Oliker, J. Shalf, and K. Yelick, "Impact of modern memory subsystems on cache optimizations for stencil computations," in *Proceedings of the 2005 workshop on Memory system performance*, ACM, 2005, pp. 36–43.

[103] F. Fatollahi-Fard, D. Donofrio, G. Michelogiannakis, and J. Shalf, "Opensoc fabric: on-chip network generator: using chisel to generate a parameterizable on-chip interconnect fabric," in *Proceedings of the 2014 International Workshop on Network on Chip Architectures*, ACM, 2014, pp. 45–50.

[104] J. Jayaraj, A. F. Rodrigues, S. D. Hammond, and G. R. Voskuilen, "The potential and perils of multi-level memory," in *Proceedings of the 2015 International Symposium on Memory Systems*, ACM, 2015, pp. 191–196.

[105] M. Glick, "Optical interconnects in next generation data centers: an end to end view," in *Optical Interconnects for Future Data Center Networks*, Springer, 2013, pp. 31–46.

[106] K. Padmaraju, L.-W. Luo, X. Zhu, M. Glick, R. Dutt, M. Lipson, and K. Bergman, "Wavelength locking of a wdm silicon microring demultiplexer using dithering signals," in *Optical Fiber Communication Conference*, Optical Society of America, 2014, Tu2E–4.

[107] K. Padmaraju and K. Bergman, "Resolving the thermal challenges for silicon microring resonator devices," *Nanophotonics*, vol. 3, no. 4-5, pp. 269–281, 2014.

[108] J. Liu, J. Wu, and D. K. Panda, "High performance rdma-based mpi implementation over infiniband," *International Journal of Parallel Programming*, vol. 32, no. 3, pp. 167–198, 2004.

[109] P. Balaji, A. Chan, W. Gropp, R. Thakur, and E. Lusk, "The importance of non-data-communication overheads in mpi," *International Journal of High Performance Computing Applications*, vol. 24, no. 1, pp. 5–15, 2010.

[110] W. Jiang, J. Liu, H.-W. Jin, D. K. Panda, W. Gropp, and R. Thakur, "High performance mpi-2 one-sided communication over infiniband," in *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*, IEEE, 2004, pp. 531–538.

[111] T. Hoefler, J. Dinan, D. Buntinas, P. Balaji, B. W. Barrett, R. Brightwell, W. Gropp, V. Kale, and R. Thakur, "Leveraging mpi's one-sided communication interface for shared-memory programming," in *European MPI Users' Group Meeting*, Springer, 2012, pp. 132–141.

[112] S. W. Poole, O. Hernandez, J. A. Kuehn, G. M. Shipman, A. Curtis, and K. Feind, "Openshmem-toward a unified rma model," in *Encyclopedia of Parallel Computing*, Springer, 2011, pp. 1379–1391.

[113] K. Yelick, D. Bonachea, W.-Y. Chen, P. Colella, K. Datta, J. Duell, S. L. Graham, P. Hargrove, P. Hilfinger, P. Husbands, *et al.*, "Productivity and

performance using partitioned global address space languages," in *Proceedings of the 2007 international workshop on Parallel symbolic computation*, ACM, 2007, pp. 24–32.

[114] X. Zhu, K. Padmaraju, L.-W. Luo, S. Yang, M. Glick, R. Dutt, M. Lipson, and K. Bergman, "Fast wavelength locking of a microring resonator," *IEEE Photonics Technology Letters*, vol. 26, no. 23, pp. 2365–2368, 2014.

[115] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," in *Proc. ACM/IEEE SC 2005 Conf. Supercomputing*, Nov. 2005, p. 16. DOI: 10.1109/SC.2005.48.

[116] A. Shacham, B. G. Lee, A. Biberman, K. Bergman, and L. P. Carloni, "Photonic noc for dma communications in chip multiprocessors," in *15th Annual IEEE Symposium on High-Performance Interconnects (HOTI 2007)*, IEEE, 2007, pp. 29–38.

[117] G. Hendry, E. Robinson, V. Gleyzer, J. Chan, L. Carloni, N. Bliss, and K. Bergman, "Circuit-switched memory access in photonic interconnection networks for high-performance embedded computing," in *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2010, pp. 1–12.

[118] M. R. Madarbux, A. Van Laer, and P. M. Watts, "Low latency scheduling algorithm for shared memory communications over optical networks," in *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*, IEEE, 2013, pp. 83–86.

[119] G. Hendry, "Decreasing network power with on-off links informed by scientific applications," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, IEEE, 2013, pp. 868–875.

[120] G. Keramidas, P. Petoumenos, and S. Kaxiras, "Cache replacement based on reuse-distance prediction," in *Computer Design, 2007. ICCD 2007. 25th International Conference on*, IEEE, 2007, pp. 245–250.

[121] T. Shiraishi, Q. Li, Y. Liu, X. Zhu, K. Padmaraju, R. Ding, M. Hochberg, and K. Bergman, "A reconfigurable and redundant optically-connected memory system using a silicon photonic switch," in *Optical Fiber Communication Conference*, Optical Society of America, 2014, Th2A–10.

[122] R. W. Numrich and M. A. Heroux, "A performance model with a fixed point for a molecular dynamics kernel," *Computer Science-Research and Development*, vol. 23, no. 3-4, pp. 195–201, 2009.

[123] I. Karlin, A. Bhatele, B. L. Chamberlain, J. Cohen, Z. Devito, M. Gokhale, R. Haque, R. Hornung, J. Keasler, D. Laney, *et al.*, "Lulesh programming model and performance ports overview," *Lawrence Livermore National Laboratory (LLNL), Livermore, CA, Tech. Rep*, 2012.

[124] I. Karlin, A. Bhatele, J. Keasler, B. L. Chamberlain, J. Cohen, Z. Devito, R. Haque, D. Laney, E. Luke, F. Wang, *et al.*, "Exploring traditional and emerging parallel programming models using a proxy application," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, IEEE, 2013, pp. 919–932.

[125] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in *Proc. Symp. th Annual Int Computer Architecture*, May 1990, pp. 364–373. DOI: 10.1109/ISCA.1990.134547.

[126] T. C. Mowry, M. S. Lam, and A. Gupta, "Design and evaluation of a compiler algorithm for prefetching," in *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS V, Boston, Massachusetts, USA: ACM, 1992, pp. 62–73, ISBN: 0-89791-534-8. DOI: 10.1145/143365.143488. [Online]. Available: http://doi.acm.org/10.1145/143365.143488.

[127] P. Cao, E. W. Felten, A. R. Karlin, and K. Li, "A study of integrated prefetching and caching strategies," *SIGMETRICS Perform. Eval. Rev.*, vol. 23, no. 1, pp. 188–197, May 1995, ISSN: 0163-5999. DOI: 10.1145/223586.223608. [Online]. Available: http://doi.acm.org/10.1145/223586.223608.

[128] *EXACT Mini Apps*, http://portal.nersc.gov/project/CAL/exact.htm, [Online].

[129] J. J. Wilke and J. P. Kenny, "Using discrete event simulation for programming model exploration at extreme-scale: macroscale components for the structural simulation toolkit (sst)," *Sandia National Laboratories, Tech. Rep. SAND2015-1027*, 2015.

[130] M. Petracca, B. G. Lee, K. Bergman, and L. P. Carloni, "Design exploration of optical interconnection networks for chip multiprocessors," in *2008 16th IEEE Symposium on High Performance Interconnects*, IEEE, 2008, pp. 31–40.

[131] N. Sherwood-Droz, H. Wang, L. Chen, B. G. Lee, A. Biberman, K. Bergman, and M. Lipson, "Optical 4× 4 hitless silicon router for optical networks-on-chip (noc)," *Optics express*, vol. 16, no. 20, pp. 15 915–15 922, 2008, ISSN: 1094-4087.

[132] D. Calhoun, K. Wen, X. Zhu, S. Rumley, L. Luo, Y. Liu, R. Ding, T. Baehr-Jones, M. Hochberg, and M. Lipson, "Dynamic reconfiguration of silicon photonic circuit switched interconnection networks," in *Proc. IEEE High Perform. Extreme Comput. Conf*, Citeseer, 2014.

[133] Q. Yang, K. Bergman, G. D. Hughes, and F. G. Johnson, "Wdm packet routing for high-capacity data networks," *Journal of Lightwave Technology*, vol. 19, no. 10, pp. 1420–1426, 2001.

[134] A. Shacham, B. A. Small, O. Liboiron-Ladouceur, and K. Bergman, "A fully implemented 12 12 data vortex optical packet switching interconnection network," *Journal of Lightwave Technology*, vol. 23, no. 10, p. 3066, 2005.

[135] O. Liboiron-Ladouceur, A. Shacham, B. A. Small, B. G. Lee, H. Wang, C. P. Lai, A. Biberman, and K. Bergman, "The data vortex optical packet switched interconnection network," *Journal of Lightwave Technology*, vol. 26, no. 13, pp. 1777–1789, 2008.

[136] X. Yang, J. Jenkins, M. Mubarak, R. B. Ross, and Z. Lan, "Watch out for the bully! job interference study on dragonfly network," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ser. SC*, vol. 16, 2016.

[137] J. C. Bennett, J. J. Wilke, N. L. Slattengren, K. Teranishi, K. Franko, G. D. Sjaardema, P. Lin, and H. Kolla, "Dharma: distributed asynchronous adaptive resilient management of applications.," Sandia National Laboratories (SNL-CA), Livermore, CA (United States); Sandia National Laboratories, Albuquerque, NM, Tech. Rep., 2015.

Appendix

Relevant Author Publications

1. K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, J. Wilke, K. Bergman, "Flexfly: Enabling a Reconfigurable Dragonfly Through Silicon Photonics," (Best Student Paper Award) The International Conference for High Performance Computing, Networking, Storage and Analysis (SC) (Nov 2016).
2. K. Wen, S. Rumley, P. Samadi, C. P. Chen, K. Bergman, "Silicon Photonics in Post Moore's Law Era: Technological and Architectural Implications," Post-Moore's Era Supercomputing (PMES) Workshop in conjuction with SC 2016.
3. K. Wen, H. Guan, D. M. Calhoun, D. Donofrio, J. Shalf, K. Bergman, "Silicon Photonic Memory Interconnect for Many-Core Architectures," IEEE High Performance Extreme Computing Conference (HPEC) (Sep 2016).
4. K. Wen, S. Rumley, J. Wilke, K. Bergman, "Latency-avoiding Dynamic Optical Circuit Prefetching Using Application-specific Predictors," First International Workshop on Communication Architectures at Extreme Scale (ExaComm) (Jul 2015).
5. K. Wen, S. Rumley, P. Mantovani, L. Carloni, K. Bergman, "Characterization of Accelerated 2D FFT with Off-Chip Optical Channels and Kernel Adaptation for Efficient Utilization," SEAK 2015: DAC Workshop on Suite of Embedded Applications and Kernels (Jun 2015).

6. K. Wen, D. M. Calhoun, S. Rumley, X. Zhu, Y. Liu, L. Luo, R. Ding, T. Baehr-Jones, M. Hochberg, M. Lipson, K. Bergman, "Reuse Distance Based Circuit Replacement in Silicon Photonic Interconnection Networks for HPC," IEEE Symposium on High Performance Interconnects (Hot Interconnects) (Aug 2014).
7. K. Wen, J. Wilke, S. Rumley, K. Bergman, "Modeling Performance and Energy Consumption of Silicon Photonic Interconnection Networks via Analytical Cache Models," Workshop on Modeling & Simulation of Systems and Applications (Aug 2014).
8. K. Wen, S. Rumley, K. Bergman, "Designing Silicon Photonic Interconnection Networks for Deadline-Driven Applications (invited)," Opto Electronics and Communications Conference (OECC) (Jul 2014).
9. K. Wen, S. Rumley, K. Bergman, "Reducing Energy per Delivered Bit in Silicon Photonic Interconnection Networks," IEEE Optical Interconnects Conference (May 2014).
10. P. Samadi, K. Wen, J. Xu, K. Bergman, "Software-defined optical network for metro-scale geographically distributed data centers," Optics Express 24 (11) (May 2016).
11. P. Samadi, K. Wen, J. Xu, Y. Shen, K. Bergman, "Reconfigurable Optical Dragonfly Architecture for High Performance Computing," Optical Fiber Communication (OFC) Th2A.60 (Mar 2016).
12. S. Rumley, M. Bahadori, K. Wen, D. Nikolova, K. Bergman, "PhoenixSim: Crosslayer Design and Modeling of Silicon Photonic Interconnects," 1st International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems (AISTECS) (Jan 2016).
13. D. M. Calhoun, Q. Li, D. Nikolova, C. P. Chen, K. Wen, S. Rumley, K. Bergman, "Hardware-Software Integrated Silicon Photonics for Computing Sys-

tems [Book Chapter]," Silicon Photonics III of the series Topics in Applied Physics 122 pp 157-189 (Jan 9, 2016).

14. C. P. Chen, X. Zhu, Y. Liu, K. Wen, M. S. Chik, T. Baehr-Jones, M. Hochberg, K. Bergman, "Programmable Dynamically-Controlled Silicon Photonic Switch Fabric [invited]," IEEE/OSA Journal of Lightwave Technology 34 (12) 2952-2958 (Dec 2015).
15. P. Samadi, J. Xu, K. Wen, H. Guan, Z. Li, K. Bergman, "Experimental Demonstration of Converged Inter/Intra Data Center Network Architecture," 17th International Conference on Transparent Optical Networks ICTON 2015 (Jul 2015).
16. P. Samadi, H. Guan, K. Wen, K. Bergman, "A Software-Defined Optical Gateway for Converged Inter/Intra Data Center Networks," IEEE Optical Interconnects Conference (Apr 2015).
17. D. M. Calhoun, K. Wen, X. Zhu, S. Rumley, L. Luo, Y. Liu, R. Ding, T. Baehr-Jones, M. Hochberg, M. Lipson, K. Bergman, "Dynamic Reconfiguration of Silicon Photonic Circuit Switched Interconnection Networks," IEEE High Performance Extreme Computing Conference (HPEC) (Aug 2014).
18. S. Rumley, D. Nikolova, R. Hendry, K. Wen, K. Bergman, "Modeling Silicon Photonics in Distributed Computing Systems: From the Device to the Rack [Invited]," Advanced Photonics for Communications JM4B.2 (Jul 2014).
19. K. Bergman, S. Rumley, N. Ophir, D. Nikolova, R. Hendry, Q. Li, K. Padmaraju, K. Wen, X. Zhu, "Silicon photonics for exascale systems (Tutorial)," Optical Fiber Communications Conference and Exhibition (OFC) (Mar 2014).