SPRINT: Scalable Photonic Switching Fabric for High-Performance Computing (HPC)

Brian Neel, Randy Morris, Dominic Ditomaso, and Avinash Kodi

Abstract—High-performance computing systems and datacenters will interconnect hundreds to thousands of heterogeneous general purpose or specialized cores in the future. As the number of network end-point sockets scales exponentially, the underlying communication fabric must deliver high bandwidth with low power and reduced switching complexity. While high-radix routers enable smaller diameter networks, the penalty is in increased switching complexity and router power. In this paper, we propose SPRINT (scalable photonic reconfigurable interconnect), which can scale to a large number of cores using photonic switching implemented with silicon micro-ring resonators (MRRs). MRRs are low power, high bandwidth photonic switching devices that can be arranged to function similarly to a high-radix router with reduced complexity and power. We will first show the design of a 64 core cluster using optical interconnects and electrical packet switching. To build scalable switching crossconnects, we investigate the design of 256-, 512- and 1024-socket versions of SPRINT connected to function as passive arrayed-waveguide gratings. Our proposed switching crossconnect with single and dual micro-rings minimizes the hop count to 4 for a 1024 core network while reducing the power dissipation, increasing the bandwidth and reducing the switching complexity.

Index Terms—Datacenters; HPC; Nanophotonics.

I. INTRODUCTION

Future high-performance computing (HPC) systems and datacenter networks (DCNs) will require scalable switching fabrics which are both energy-efficient and high-throughput networks with reasonable switching complexity. As the scale of the platform increases, the interconnection network can become a bottleneck decreasing the overall throughput. This bottleneck increases processor latency as it waits for memory responses from other processors decreasing the performance of the system as a whole. Further, the network may be difficult to scale or expand in future deployments as the system size increases. Therefore, HPC and DCNs require a balance of performance, scalability and cost.

Electrical signaling over large distances leads to many negative effects on performance such as increased processor latency, execution time and increased power consumption [1–3]. One solution is to use optical interconnects for longer communications as optics offers several advantages: (1) bit rates are independent of distance once the laser power is set to overcome the losses in the longest communication link, (2) high bandwidth due to multiplexing of wavelengths into a single waveguide, (3) large bandwidth density by stitching multiple waveguides close together and (4) low power since the majority of the power is consumed at the socket endpoints [1]. At the source and destination of optical communications, there is an electrical to optical (E/O) and optical to electrical (O/E) conversion to allow light to transmit the data. The reason optics is preferred for longer communications is that the laser power is tuned to overcome the losses in the longest link. Once this is done, the power is the same for all communications using the laser. For longer links, this power can be much less than what would be required for an electrical link. However, for shorter links there is much less power savings and it is not worth the extra cost for optical components. Optical communications can also increase bandwidth by making use of wavelength division multiplexing (WDM) and space division multiplexing (SDM) to send multiple wavelengths down the same or different physical links. A drawback to using optics is that the cost of optical components requires more upfront cost in comparison to an electrical network. This leads to the use of hybrid networks using short low cost electrical links for low level interconnections and low power optical links for longer high level connections.

The electrical Fat Tree topology with commodity switches is simple to scale, but does not provide adequate bandwidth at top level routers and has a large number of hops from source to destination, increasing network latency. The maximum number of hops (network diameter) can be reduced by increasing the radix of the switch, which in turn enables the connection of several cores from a single crossbar. Networks such as Flattened Butterfly [4] reduce the number of hops by increasing the radix of the switch but can run into scalability issues past 64 cores, as to continue scaling the network requires scaling the port counts of the routers and switches, which can become costly and increase the latency of a packet to move through the router. Another topology called Dragonfly [5] uses high-radix routers to reduce the hop count by implementing a virtual router using a group of lower radix routers to increase the radix of the virtual switch. Dragonfly uses a few long optical global channels to connect the high-radix virtual routers requiring load balancing on the global channels to make the best use of the limited global bandwidth available. The Black Widow [6] High-Radix Clos Network by Cray uses 64 × 64 routers to build a variation...
of a Folded Clos and Fat Tree network interconnected with side links to other trees. While Black Widow interconnects an impressive 32,000 processors, the routing through the network is complicated by routing variable length packets up and down the network tree from source to destination. Therefore, while scalable switching fabrics are available, the cost is in the switching complexity and increased power dissipation.

In this paper, we propose SPRINT (scalable photonic reconfigurable interconnect), a scalable optoelectronic switching network with high bandwidth (10 Gb/s per wavelength), low latency (<3 ns) and reduced complexity switches. We propose to exploit the emerging technology of nanophotonics to develop the high-radix functionality by re-arranging the micro-ring resonators (MRRs) as the switching interconnect [7–12]. By re-arranging the MRRs as switching blocks, we then implement the passive arrayed-waveguide grating (AWG) functionality for each of our proposed switches [13]. We propose a 4 × 4 non-blocking switching interconnect that allows four links to interact as if they were 16 physical links without the need for electrical packet switching within the optical crossbar. The optical crossbar implementation uses WDM to send multiple wavelengths per link, which can be exchanged between the links within an AWG switch designed using MRRs. Using these 4 × 4 switching interconnects as the building blocks, we propose to extend the design from 256 to 1024 cores. Further, we also evaluate two designs (single and dual MRRs) within the switching crossconnect which has area and power implications. To reduce the cost of the switching interconnect, we evaluate reduced switching configurations called SPRINT-8 (partial) and SPRINT-4 (minimal) using standard synthetic traffic patterns. Our simulation results clearly indicate that 1) an AWG optical crossbar built with low power MRRs can be used in a scalable network design for HPC systems, 2) SPRINT dissipates low power when compared to other network designs, 3) SPRINT provides as much as two times the throughput compared to mesh and cmesh and 4) the SPRINT-8 and SPRINT-4 configurations provide flexibility in network application with only a small loss in power and performance over a fully implemented SPRINT:

- We propose the design of a photonic interconnect for HPC systems that can scale from 64 to 1024 cores with minimal hop count; a 1024 core network sees only a hop count of 4.
- We utilize MRRs to function as a switching crossconnect that emulates an AWG with flexibility in switching individual channels.
- We evaluate the power and throughput of different configurations of SPRINT and compare them to electrical configurations such as mesh and cmesh on synthetic traffic for 256 to 1024 cores.

The paper is organized as follows: Section II discusses related work, Section III discusses the proposed architecture, Section IV discusses the photonic switching architecture, Section V explains the performance and Section VI concludes the paper.

II. RELATED WORK

With the recent growth in multi-core architectures and the size of datacenters growing to warehouse scale proportions, a few high-performance network designs have been proposed. C-Through [14] uses two networks in combination, an electrical packet switched network and an optical circuit switched network, to connect top of rack (TOR) switches within datacenters. While adding the optical network on top of the existing electrical network does have its advantages in terms of bandwidth, there are a few drawbacks. The circuit switched optical network requires milliseconds to reconfigure the circuit switched paths and tear them back down. It also requires modification of the TOR to enlarge the socket buffers allowing the two networks to be integrated together. Helios [15] uses a similar circuit switched network to connect pods of servers held in self-contained shipping containers. It uses multiple transceivers and receivers per pod to make multiple circuit switched connections to other pods. Helios also employs WDM to make “super links” for increased bandwidth between high traffic communication pods. However, like C-Through, Helios does not address the issue of high network loads from more than a couple of sources at a time and can cause latency issues with the time it takes to circuit switch the optical links. DOS [16] is a fully optical network using large N × N AWG switches which demonstrate good performance even with a low number of wavelengths per link. DOS implements an optical switch that requires arbitration at the output ports and an electrical buffer to store packets that do not receive an output port within the optical switch. The drawback is that the network can only scale with the size of the switch; this becomes expensive and complex to implement with a high port count as large 512 × 512 optical switches are considered. Proteus [17] is an optical network designed to change the optical topology of a DCN based on load. Using micro-electro-mechanical switches (MEMSs) Proteus can dynamically assign links from any TOR to any other TOR and then computes an optimal routing through the new network topology. Firefly [18] is an optoelectronic network on chip (NoC) that uses both an electrical and an optical network to improve performance between cores on a chip but can see large latencies with token sharing. A point-to-point macro-chip network has been proposed that was connected such that any row and column communicated with a single hop [19]; while the cluster communication in SPRINT is similar in design, the implementation using non-blocking switches implemented using MRRs is different. Moreover, we show that the design is scalable to very large numbers of cores (512 and 1024).

III. ARCHITECTURE

In this section, we evaluate the design and implementation of the proposed SPRINT architecture. One of the objectives is to minimize the hop count, which in turn will decrease the latency and thereby improve the performance. We will first discuss the implementation of a 64 core architecture and how this architecture can be extended to 256, 512 and 1024 cores.

A. Topology

In SPRINT, we combine four cores concentrated to form a core group [20]; this grouping reduces the cost of interconnecting optical transceivers at every core by reducing the number of necessary modulators and amount of O/E conversion circuitry.
While concentrating more cores is possible, it can lead to more core contention and increased latency to leave the core group itself; therefore we limit to four cores. We group core groups together to form a cluster and clusters into a system domain. SPRINT is scaled in two dimensions \((x, y)\) and the switches are interconnected within these two dimensions which also scale in two dimensions. We propose two levels of scaling, intra-cluster and inter-cluster, which are indicated with C and D (cluster and domain); therefore each core can be identified as \(S_p(x, y)\) within the cluster and \(D_x, D_y\) within the domain with the following constraints: \(0 \leq x \leq 3, 0 \leq y \leq 3\) and \(0 \leq D_x \leq D_y \leq 3\). With \(C_x = 4, C_y = 4, D_x = 4\) and \(D_y = 4\), we can potentially have 16 clusters with 256 core groups and 1024 cores with concentration. We use two levels of switching: local and global. Local switching (identified as \(S_f(x, y)\)) is used within the cluster and global switching (identified as \(S_g(x, y)\)) across the clusters.

Figure 1 shows a 64 core configuration of SPRINT with \(C_x = 4, C_y = 4, D_x = 0\), and \(D_y = 0\). The top inset shows the core concentration proposed in SPRINT. In this configuration, we interconnect a \(4 \times 4\) non-blocking switch to interconnect the four clusters in both the \(x\) and \(y\) dimensions using \(S_f(x, y)\), where \(1 \leq x \leq 4\) and \(1 \leq y \leq 4\). The switch numbers with \(x\) increasing are used to connect the core groups along the \(x\)-direction and those with \(y\) increasing are used to connect the core groups along the \(y\)-direction. The local switch inset shows the wavelength distribution for communication. The \(4 \times 4\) AWG can switch from the four ports in a non-blocking configuration. For example, consider the wavelengths \(\Lambda = \lambda_1^1, \lambda_1^2, \lambda_1^3, \lambda_1^4\); each of the output ports receives one wavelength, so that \(\lambda_1^1, \lambda_2^1, \lambda_3^1\) and \(\lambda_4^1\) arrive at each of the output ports 1, 2, 3, and 4, respectively. This permits optical communication across the row or column. Therefore, intra-cluster communication between any two core groups will be restricted to a maximum of two hops. To illustrate with an example, consider that \(S_p(0,0)\) (the other two coordinates are dropped for the single cluster) requires communication with \(S_p(3,2)\). We propose to use dimension-order YX routing, i.e., \(y\) is routed first and then the \(x\) dimension. The packet first moves into the router microarchitecture, then through \(S_f(1,0)\) to \(S_p(0,2)\), and then to \(S_p(3,2)\) via \(S_f(0,2)\); this results in a maximum of two hops at the intra-cluster level. Figure 1 also shows the router microarchitecture as an inset; with four cores connected to the crossbar and three connections in the \(x\) and \(y\) directions, the crossbar will be a \(10 \times 10\) router. With shared last level cache, we can reduce the core connections as all of the cores will be connected to the same input and output ports. This decreases the crossbar complexity to \(7 \times 7\). There is a tradeoff between the router radix and power consumption: as the radix increases, the power mostly increases, although the increase in radix helps to reduce the hop count and serialization latency.

Figure 2 shows the scalable architecture from 256 to 1024 cores with the same framework as discussed earlier. Figure 2(a) shows the proposed architecture with 256 cores, which consists of four clusters connected with similar non-blocking switches. The intra-cluster communication remains as before; for inter-cluster switching we use global \(4 \times 4\) switches which connect similar core groups in different clusters. For example, core groups \(S_p(0,0,0,0), S_p(0,0,1,0), S_p(0,2,0,0)\) and \(S_g(0,0,3,0)\) are connected together with switch \(S_g(x(1,0))\); similarly, \(S_p(1,0,D_x,0)\) where \(0 \leq D_x \leq 3\) are connected with switch \(S_g(x(2,0))\), and so on. This provides each cluster with...
a direct communication channel to the global crossbar and enhances communication. With scalable bandwidth, any core can be reached within four hops: two for intra-cluster and two for inter-cluster communication. Figure 2(b) shows the 512 configuration where the network is scaled by duplicating the 256 core configuration and adding switch connections in the y dimension. Here, core groups \( S_p(0,0,0,0) \), \( S_p(0,1,0,0) \), \( S_p(0,0,0,1) \) and \( S_p(0,1,1,1) \) are connected with switch \( S_{gy}(0,1) \). We minimize the connections by connecting two of the core groups within each cluster. Figure 2(c) shows the 1024 core configuration. Here, we duplicate the 512 configuration, and connect in the y dimension similarly to the x dimension (as in 256 core organization). Core groups \( S_p(0,0,0,D_y) \) where \( 0 \leq D_y \leq 3 \) are connected together with switch \( S_{gy}(0,1) \). The hop count for a 1024 core SPRINT network is four: two for intra-cluster and two for inter-cluster. For example, suppose the source core is \( S_p(0,1,1,0) \) and the destination is \( S_p(3,3,2,1) \). With dimension-order routing (DOR), we first route in Y and then X. First, we focus on the routing within the cluster: from \( S_p(0,1,1,0) \) to \( S_p(3,3,1,0) \) and then in x towards \( S_p(3,3,1,0) \). With the intra-cluster routing completed, we move on to inter-cluster routing, where the packet will travel from \( S_p(3,3,1,0) \) to \( S_p(3,3,1,1) \) in the y dimension and then to \( S_p(3,3,2,1) \) in the x dimension. While we propose YX routing, XY routing can also be implemented and both are deadlock free.

### B. Configurations

The SPRINT network requires at least 16 global switches to connect in each dimension. While high switch counts offer scalable bandwidth and lower the latency, one problem is with the high number of switches. We will now consider some alternate designs that can be tuned to fit the needs of the platform SPRINT is being applied to. Within the group switches with global links there would be a 20 × 20 electrical crossbar (4 cores × 4 × number of dimensions). One way to reduce the size of the crossbar is to reduce the number of dimensions each core group switch will connect. We explore the power and performance of connecting only four switches for SPRINT-4 or eight switches for SPRINT-8 in both the x and y dimensions between clusters to reduce the size of the crossbar required within some of the group switches. These networks could be used for other reasons such as saving upfront costs in building the network or in the case that less bandwidth is needed between clusters.

Another option available is to increase the number of wavelengths used per link from 4 to 16, 32, or even 64. The network itself is really unaffected by changing the number of wavelengths as we can still use a 4 × 4 optical crossbar but it considers a group of wavelengths to be exchanged rather than just a single wavelength. For example, if we instead used 64 wavelengths...
wavelengths, $\lambda_{0-63}$, we could consider them as four groups, $\lambda_{(0-15)}$, $\lambda_{(16-31)}$, $\lambda_{(32-47)}$, and $\lambda_{(48-63)}$. The four groups could then be exchanged in the $4 \times 4$ optical crossbar in the same manner as the four single wavelengths, allowing the same function with 16 times the bandwidth. This design may require more MRs, where each wavelength to be switched uses a separate MRR within the AWG optical crossbars. This design also requires more circuitry and lasers in the group switches, but does not affect the network topology and may better suit the needs of the platform requirements. It should also be noted that we are not proposing extra wavelengths for increased connectivity between clusters but extra bandwidth on the links so that a larger chip size can be used to reduce the time in sending a single packet. Future designs using 3D stacking of resonators or combinations of smaller optical crossbars may lead to scaling the optical crossbars to larger sizes for increased connectivity between core groups.

### IV. Optical Switching Crossbar

This section provides a detailed explanation of the operation of an AWG [21]. In addition, we propose two different implementations of the AWG and give an analysis of both designs.

#### A. AWG Functionality

Figure 3 shows an example of a $4 \times 4$ 64-wavelength AWG. Here, the wavelengths are indicated as $\lambda_{(a-b)}$, where $a-b$ indicates the wavelength range and $c$ indicates the input port. For the $4 \times 4$ 64-wavelength AWG, consider input port 0. All input wavelengths are indicated as $\lambda_{(0-15)}$, $\lambda_{(16-31)}$, $\lambda_{(32-47)}$, and $\lambda_{(48-63)}$. After traversing the series of ring resonators, the wavelengths $\lambda_{(0-15)}$ arrive at output port 0, $\lambda_{(16-31)}$ at output port 1, $\lambda_{(32-47)}$ at output port 2 and $\lambda_{(48-63)}$ at output port 3. This enables a $1 \times N$ switching functionality per waveguide. Now consider input port 1. All input wavelengths are indicated as $\lambda_{(1-15)}$, $\lambda_{(16-31)}$, $\lambda_{(32-47)}$, and $\lambda_{(48-63)}$. After traversing the series of ring resonators, the wavelengths $\lambda_{(1-15)}$ arrive at output port 1, $\lambda_{(16-31)}$ at output port 2, $\lambda_{(32-47)}$ at output port 3 and $\lambda_{(48-63)}$ at output port 0. In a similar manner, the input wavelengths for input port 2 are $\lambda_{(2-15)}$, $\lambda_{(16-31)}$, $\lambda_{(32-47)}$, and $\lambda_{(48-63)}$. After traversing the series of MRRs, $\lambda_{(2-15)}$ arrives at output 2, $\lambda_{(16-31)}$ arrive at output 3, $\lambda_{(32-47)}$ arrives at output 0 and $\lambda_{(48-63)}$ arrives at output 1. Lastly, input 3 wavelengths are given as $\lambda_{(3-15)}$, $\lambda_{(16-31)}$, $\lambda_{(32-47)}$, and $\lambda_{(48-63)}$. After traversing the series of MRRs, $\lambda_{(3-15)}$ arrives at output 3, $\lambda_{(16-31)}$ arrives at output 0, $\lambda_{(32-47)}$ arrives at output 1 and $\lambda_{(48-63)}$ arrives at output 2. This creates a $4 \times 4$ switching functionality device. It should be mentioned that the above process can be directly applied to create any size switching device.

#### B. Single Ring AWG

This subsection discusses the construction of an AWG using single MRRs. Figure 4(a) shows the single MRR implementation of a 64-wavelength $4 \times 4$ AWG. It should be noted that this AWG implementation is an extended version of the optical crossbar proposed by [22]. As can be seen, each waveguide is routed in a manner that allows it to come in close proximity to the other three waveguides. At these closest proximity points, a select range of wavelengths is switched between waveguides. This switching of light between two different waveguides allows light coming from one waveguide to be switched to another waveguide. In terms of functionality, this allows a single tile the ability to communicate with multiple other tiles using only one input waveguide. In Fig. 4(a), $\lambda_{(32-47)}$ is switched at the intersection of waveguide 0 and waveguide 1 and also at the intersection of waveguide 2 and waveguide 3, $\lambda_{(0-15)}$ is switched at the intersection of waveguide 0 and waveguide 3 and also at the intersection of waveguide 1 and waveguide 2, $\lambda_{(16-31)}$ is switched at the intersection of waveguide 0 and waveguide 2 and, lastly, $\lambda_{(48-63)}$ is switched at the intersection of waveguide 1 and waveguide 3. The above-mentioned switching of selected wavelengths at unique waveguide intersections results in a $4 \times 4$ 64-wavelength AWG.

For a further understanding of the single ring AWG, we will show how light from waveguide 0 is switched and arrives on the four output waveguides. As light travels down waveguide 0, it first encounters the intersection with waveguide 1. At this point $\lambda_{(0)}$ is placed on waveguide 1, allowing input 0 to communicate with output 2 using $\lambda_{(0)}$. Then light traveling down waveguide 0 encounters the intersection with waveguide 3. At this point, $\lambda_{(1)}$ is placed on waveguide 3, allowing input 0 to communicate with output 0 using $\lambda_{(0)}$. As the light continues traveling down, it encounters the intersection with waveguide 2. At this point $\lambda_{(2)}$ is placed on waveguide 2, allowing input 0 to communicate with output 1 using $\lambda_{(2)}$. Lastly, $\lambda_{(3)}$ arrives at output 3 as it was the only light not switched. This allows input 0 to communicate with output 0 using $\lambda_{(0)}$. This concept is expanded for other inputs, which creates a $4 \times 4$ 64-wavelength AWG.

#### C. Double Ring AWG

This subsection discusses the construction of an AWG using double MRRs. A double MRR consists of two MRRs that are placed in between two waveguides to retain the same direction of light from the input to the output port. Figure 4(c) shows the operating principle of a double micro-ring resonator allowing light of the same wavelength to be switched between the two waveguides. $\lambda_{(0)}$ is switched to the bottom waveguide and $\lambda_{(2)}$ is switched to the top waveguide. Figure 4(b) shows the...
double MRR implementation of a 64-wavelength 4 × 4 AWG. It should be mentioned that each MRR in the figure represents 16 MRRs for clarity and these would be placed adjacent to each other allowing 15 additional wavelengths to be switched between waveguides. In the double ring AWG, wavelengths are switched at locations where two waveguides are running parallel to each other. At this point, light is switched from one waveguide to the other. This switching enables an input port to be connected to all the output ports. The operating principle and functionality of the double ring AWG is identical to the single ring AWG, where the input and output wavelengths are the same. The difference between the two designs is how each one switches wavelengths. In Fig. 4(b), λ(0–15) is switched between waveguide 0 and waveguide 1, waveguide 2 and waveguide 3 and also between waveguide 0 and waveguide 3. λ(16–31) is switched between waveguide 1 and waveguide 2, λ(32–47) is switched between waveguide 0 and waveguide 3 and also between waveguide 1 and waveguide 2 and also between waveguide 0 and waveguide 3. This creates a functionality identical to the single ring AWG implementation.

D. Comparison

In this subsection, we compare the two AWG implementations in terms of area, optical loss and number of MRRs.

1) Single Ring AWG Analysis: The single ring AWG consists of three and four sets of MRRs in the vertical and horizontal directions, respectively. The estimated area overhead for each set is 60 μm × 90 μm. As the four-input 64-wavelength AWG crossbar consists of 16 MRRs at each set, the horizontal and vertical lengths will increase by 16 fold. This causes the four-input 64-wavelength AWG area to be 1440 μm × 960 μm. In addition, each waveguide has three waveguide crossings and traverses a maximum distance of about 1.7 mm. Moreover, the AWG is constructed with 96 MRRs.

2) Double Ring AWG Analysis: The double ring AWG consists of a total of eight sets of double ring resonators. Each set is comprised of 32 MRRs or a total of 256 MRRs are used to construct the double ring AWG. The AWG contains 96 double MRRs, where each double MRR has dimensions of 15 μm × 20 μm. This includes a 5 μm spacing between each double MRR, which is used to prevent cross coupling between adjacent MRRs. This results in the AWG having a height of 80 μm (three double MRR sets and four waveguides) and a width of 720 μm (three double micro-ring sets). The double ring AWG has two waveguide crossings with a maximum distance of about 1 mm.

Table I shows the calculated values for each AWG design. In calculating the optical loss, we used a waveguide loss of −1.3 dB/em, a micro-ring traversal loss of −1 dB, a waveguide cross-over loss of −0.05 dB and a bending loss of −1 dB. The single ring AWG has less optical power loss than the double ring AWG, mainly due to the fact that the double rings incur a traversal loss of −2 dB. The major contribution to the optical loss for the single ring AWG is bending losses as multiple bends are required to accommodate the 16-wavelength switching. The double ring AWG occupies less area overhead, which may come as a surprise, as it uses 160 more MRRs. The reason is that multiple ring resonators can be stacked on top of each other due to the different wavelengths being switched. Moreover, the single ring AWG requires more bending than the double ring AWG, which increases the area overhead. In comparison, the single ring AWG design should be used if optical loss needs to be minimized and the double ring AWG design should be used if area overhead needs to be minimized.

V. Performance Evaluation

We evaluate SPRINT and compare different configurations of SPRINT (SPRINT-8 and SPRINT-4) with electrical topologies such as mesh and cmesh (concentrated mesh) in terms of power and performance.

A. Power Evaluation

We use an Orion [23] power and area simulator to determine the power of the electrical components within the network. A flit is considered as the smallest value of a data packet that can be routed individually through the network. Assuming 32 nm technology with a 128 bit flit and a network frequency of 1 GHz, we estimate the buffer power to be 8.09 mW for a single 128 bit register. The crossbar is the largest consumer of power.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>COMPARISON BETWEEN THE SINGLE AND DOUBLE MICRO-RING AWG DESIGNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
</tr>
<tr>
<td>Micro-ring resonators</td>
<td>96</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>1.38</td>
</tr>
<tr>
<td>Optical loss (−dB)</td>
<td>4.32</td>
</tr>
</tbody>
</table>
within the network as we consider a $16 \times 16$ crossbar for the four dimensionally linked switches and a $13 \times 13$ crossbar for three dimensionally linked switches. A $16 \times 16$ crossbar uses 169 mW of power per 128 bit input while a $13 \times 13$ crossbar uses 111 mW. We simulate under large crossbar conditions as the worst case scenario, as the crossbar could be reduced in size as mentioned above. The conversion from E/O and back from an optical signal to an electrical signal requires only 158 pJ/b or approximately 20.2 nJ/flit making the conversion process power very low in comparison to the other components. The optical devices will have a combined latency of 0.3 ns providing low-latency communications [24]. We use the single MRR optical crossbar architecture, as it is slightly more power efficient than the double ring design, with a $-26$ dB in optical losses. The losses per component are shown in Table II and the $-26$ dB link loss leads to a laser power required of 7.3 mW. The MRRs require 26 µW/ring in ring heating to keep them in resonance with their respective wavelength. Considering four wavelengths 12 MRRs would be required leading to 312 µW in ring heating power per optical crossbar. It should be noted that the laser power and ring heating are considered as static power as the laser is always on and modulated in the E/O circuitry and the MRRs are tuned to a specific wavelength. The static optical power will be amortized over the total number of packets passing through the optical crossbar. When comparing to mesh, a mesh requires only a $5 \times 5$ crossbar using 28.9 mW per flit per hop. A cmesh requires an $8 \times 8$ crossbar dissipating 56.9 mW per flit. For the electrical links there is a power cost of 0.2 pJ/b/mm with longer global links between group switches, medium length links for local switch links and short links for core to switch connections.

### B. Simulation Methodology

In this subsection, we first describe our simulation methodology before we present our results on synthetic traffic. We simulated different versions of the SPRINT network on several synthetic traffic traces including Uniform Random and permutation patterns, such as Butterfly, Complement and Tornado [25]. Each traffic pattern simulates a communication pattern found in multiprocessor shared memory programs such as the SPLASH-2 benchmark. The cycle accurate simulator OPTISIM [26] was used to evaluate the performance of PROPEL and the above-mentioned networks. OPTISIM simulates both optical and electrical components in terms of bandwidth, latency and power at an abstract level needed for performance evaluation. The optical components simulated in OPTISIM are buffers, crossbars and photodetectors. The electrical components simulated in OPTISIM are modulators, transmitters, waveguides, fibers, nanophotonic crossbars and photodetectors. The electrical

### C. Simulation Results

1) **Power:** We show the results for three power comparisons for SPRINT: 1) we compare SPRINT to mesh and cmesh for uniform traffic for 256, 512 and 1024 cores; 2) we compare 256 core networks in different traffic traces; 3) we show a comparison of the different SPRINT networks for different core counts and inter-cluster AWG crossbar counts to clearly show the benefits of each configuration.

Figure 5(a) shows the power dissipation for the 256 core version of SPRINT. We compare SPRINT-16 (fully connected switches) and SPRINT-8 (partially connected). As 256 cores is the baseline case for SPRINT we compare synthetic traffic traces to show the power dissipation for different communication patterns and compare it to the mesh network. As seen, the graph mesh power is nearly constant across all traffic patterns as it has no global links between switches. SPRINT-16 is the lowest for every traffic pattern and compared to mesh uses an average of 40% of the power. When SPRINT-8 is compared to SPRINT it uses approximately 3–5% more power with half the number of switches required for SPRINT. Therefore, even without fully implementing SPRINT we can still save around half the power that mesh uses.

Figure 5(b) shows the power dissipation when comparing different configurations of SPRINT (256, 512 and 1024) to mesh and cmesh architectures. In terms of power, all SPRINT configurations outperform the mesh and cmesh networks due to the large number of hops from source to destination for the other two networks. For mesh this is on average 11, 15 and 22 hops for 256, 512, and 1024 cores, respectively. When compared to cmesh, SPRINT only slightly outperforms for 256 cores but with more cores begins to outperform cmesh by as much as a factor of 2. It would be expected that, if the network were expanded to further dimensions, SPRINT would continue to further outperform the other networks. Cmesh performs nearly as well as SPRINT for the smaller 256 core network as cmesh is similar in average hop count to SPRINT for the 256 case. At 1024 cores, however, the hop count is nearly double that of SPRINT and allows SPRINT to make up for its slightly larger crossbar power with a lower hop count and lower power global links.

Up to this point we have compared SPRINT to mesh and cmesh for different core counts for uniform traffic and to mesh for multiple traffic patterns. Figure 5(c) shows the power when comparing the different versions of SPRINT (SPRINT-4, SPRINT-8 and SPRINT-16) configurations on different synthetic traffic traces. As expected, SPRINT-16 uses the lowest power in nearly every case because it has a maximum hop count of four when the domain is greater than 1. The other two configurations add on average one extra hop to reduce the crossbar switch count. Interestingly SPRINT-4 performs a little better than SPRINT-8 for some traffic traces.
such as Complement. This is due to the reduced power by a factor of two with the same hop count. As seen in the next section, this does affect throughput as there is more contention at the global switches. From an overall comparison of the three networks the major conclusion is that in terms of power, SPRINT-8 and SPRINT-4 are good alternatives to standard electrical networks, expending nearly half the power, and when SPRINT-16 is fully implemented it can provide the best results.

2) Saturation Throughput: Figure 6 shows the saturation throughput of the proposed SPRINT and competitive networks. SPRINT shows exceptional advantages in terms of network performance at high core counts as compared to mesh and cmesh. In terms of throughput SPRINT provides more than twice the performance of mesh and four times the performance of cmesh for 256 cores as seen in Fig. 6(a). As expected, mesh has more throughput than cmesh due to the concentration of cores, which is the tradeoff of power for performance. All three networks decrease in throughput as the core count increases as expected, because there is more contention within the switch crossbars due to the increased number of packets injected into the network, as seen in Fig. 6(b). However, SPRINT is less affected by an increase in core count because SPRINT’s diameter increases by only one with an increase in dimension whereas mesh’s diameter can increase by more than 10 with an increase in dimension. When comparing the different network configurations of SPRINT, using 16 switches per cluster has the expected effect of increasing throughput over using four or eight switches per cluster, as seen in Fig. 6(c). SPRINT has nearly a two times speedup over SPRINT-8 for uniform traffic. Overall, as indicated by the results, the use of low power optical crossbars for communication between groups of cores in HPC systems is an efficient design showing not only an improvement in power but also a dramatic improvement in performance over comparable networks.
VI. CONCLUSION

In this paper, we proposed SPRINT, a scalable photonic network for high-performance computing that tackles many of the problems in building and designing networks for large core counts. We demonstrated the scalability of the network by showing simple increases in dimensions using small $4 \times 4$ AWG optical crossbars designed using MRRs. Our analysis demonstrated two different crossbar designs and explained the pros and cons of each design. Next, we compared data from our SPRINT networks for 256, 512 and 1024 cores to mesh and cmesh networks with synthetic traffic patterns and showed the performance of each. SPRINT in all configurations outperformed the other networks with more than double the throughput and half the power in most cases. Finally, we showed that even with a minimally connected SPRINT configuration we can get a significant improvement in power and throughput while leaving flexibility in the network design.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation grants CCF-0915418, CCF-1054339 (CAREER) and ECCS-1129010.

REFERENCES


Brian Neel received his B.S. in Electrical Engineering and Computer Science from Ohio University, Athens in 2011. He is currently pursuing his M.S. degree at Ohio University. His research interests include datacenter networks, optical interconnects and computer architecture.

Randy Morris received his B.S., M.S., and Ph.D. in Electrical Engineering and Computer Science from Ohio University, Athens, in 2007, 2009, and 2012, respectively. He is currently at Intel Labs. His research interests include optical interconnects, network-on-chips (NoCs) and computer architecture.

Dominic DiTomaso received his B.S. and M.S. degrees in Electrical Engineering and Computer Science from Ohio University, Athens, in 2010 and 2012, respectively. He is currently pursuing his Ph.D. degree at Ohio University. His research interests include wireless interconnects, network-on-chips (NoCs) and computer architecture.

Avinash Karanth Kodi received his Ph.D. and M.S. degrees in Electrical and Computer Engineering from the University of Arizona, Tucson, in 2006 and 2003, respectively. He is currently an Assistant Professor of Electrical Engineering and Computer Science at Ohio University, Athens. He is the recipient of the National Science Foundation (NSF) CAREER award in 2011. His research interests include computer architecture, optical interconnects, chip multiprocessors (CMPs) and network-on-chips (NoCs). Dr. Kodi is a member of IEEE.