A Power-efficient All-optical On-chip Interconnect Using Wavelength-based Oblivious Routing

Nevin Kırman José F. Martínez

Computer Systems Laboratory
Cornell University
Ithaca, NY 14850 USA
http://m3.csl.cornell.edu/

Abstract

We present an all-optical approach to constructing data networks on chip that combines the following key features: (1) Wavelength-based routing, where the route followed by a packet depends solely on the wavelength of its carrier signal, and not on information either contained in the packet or traveling along with it. (2) Oblivious routing, by which the wavelength (and thus the route) employed to connect a source-destination pair is invariant for that pair, and does not depend on ongoing transmissions by other nodes, thereby simplifying design and operation. And (3) passive optical wavelength routers, whose routing pattern is set at design time, which allows for area and power optimizations not generally available to solutions that use dynamic routing. Compared to prior proposals, our evaluation shows that our solution is significantly more power efficient at a similar level of performance.

Categories and Subject Descriptors C.1.2 [*Processor Architectures*]: Multiple Data Stream Architectures (Multiprocessors); B.4.3 [*Input/Output and Data Communications*]: Interconnections (Subsystems)

General Terms Design, Performance

Keywords On-chip Network, Optical Network, Wavelength-based Oblivious Routing, Nanophotonics

1. INTRODUCTION

Future large-scale chip multiprocessors (CMPs) will face will face the challenge of feeding data to on-chip cores at a sufficiently high rate, both from off- and on-chip sources. Electrical on-chip data networks are likely to be severely constrained by the limited on-chip power budget, as well as long multi-hop latencies. CMOS-compatible silicon photonics is a disruptive technology that can potentially enable higher-bandwidth, lower-latency, and lower-power interconnect solutions. Recently, significant advances in CMOS compatibility, size, integration, efficiency, and high-speed operation of basic nanophotonic devices have been achieved [4–6, 20, 34, 35, 39]. An integrated optical link has been recently demonstrated [7]. Thermal sensitivity issues are also under study [18]. Driven by such rapid advancements in nanophotonics, ITRS considers on-chip optical interconnects as an alternative to global electrical wires in future process technologies [14].

A high-bandwidth, low-latency on-chip optical network can significantly benefit applications as well as the operating system: It can reduce the overhead of data sharing between parallel threads, improving parallel efficiency and scalability. And by reducing the cost of global communication and providing more uniform access to cores and memory, it can also simplify memory management, thread scheduling, and resource sharing. As a result, nanophotonics has recently elicited great interest in the computer architecture field, in the context of large-scale CMPs. We briefly review some of the most recent contributions in the subject here:

Kırman et al. [15] employ broadcast-based data communication on a full optical crossbar. It is a high-bandwidth, low-latency organization which does not require global arbitration. However, the $O(N^2)$ detector/receiver requirement is likely to be an issue for high node counts (N), in terms of sheer component count and the complexity involved in processing all the messages a node can receive simultaneously. The authors also show that a fully optical implementation of their design is inviable due to excessive power consumption. In their final solution, the authors rein in this problem by resorting to a clustered electro-optical organization that reduces the number of nodes at the optical crossbar. The downside of a clustered electro-optical approach is that its potential may be limited by the latency and power of the electrical side. Their results show modest speedups for a number of SPLASH-2 applications with respect to a fully electrical solution.

Similarly, Pan et al. [25] employ optical crossbars in a hierarchical electro-optical topology. Intra-cluster communication is facilitated via an electrical packet-switched network, and inter-cluster communication is carried on multiple optical crossbars, each connecting the routers at the same position of every cluster. The organization retains all of the routers and a lot of the router-to-router wiring of a conventional electrical network, limiting the potential gains that photonics has to offer.

Shacham et al. [26] propose a circuit-switched on-chip photonic network with reconfigurable broadband optical switches. Transmitting a data packet requires setup and breakdown of an optical path, and these are carried out on an electrical packet-switched network, where each electrical router configures an optical switch. This makes it necessary to transmit data packets of hundreds of bytes on the optical network (well beyond the size of a typical cache block) to amortize the setup/breakdown cost. Flow control is based on a combination of dropping blocked packets and adaptive routing, though the paper does not fully flesh out how forward progress is guaranteed.

Cianchetti et al. [8] propose another switch-based on-chip photonic network. It uses source-based routing and reconfigurable optical switches to route data. Switch setup is performed by converting the optical control signals that travel along the data to electrical form, and setting up the switch accordingly. Optical data signals must remain steady throughout the control setup (i.e., transmit at the rate dictated by the control network), which may limit effective bandwidth. Contention at output ports is arbitrated, and "losing" packets are electrically buffered if sufficient buffering exists, or outright dropped otherwise. In the face of network-intensive work-

ASPLOS '10 March 13–17, 2010, Pittsburgh, Pennsylvania, USA.

loads, the network may necessitate large buffering at each switch to reduce packet drop rates and any associated performance loss. Even then, the paper does not flesh out how forward progress is guaranteed in the presence of dropped packets.

Unlike the works above, Vantrease et al. [29] propose a fully-optical solution. It is a high-bandwidth, low-latency optical crossbar that uses token-based optical arbitration to serialize data transmissions to each node. They report significant speedups for SPLASH-2 applications running on a large CMP configuration relative to an electrical packet-switched network. Every node has a separate port to all other nodes' data channels, requiring $O(N^2)$ modulators/transmitters, even though only $O(N)$ of them are active at a time. The token-based arbitration can limit effective throughput, especially in light traffic conditions. Also, the large number of components, especially for high node counts, makes the viability of this architecture highly dependent on its ability to rein in the power consumption and signal losses of optical components, which will be heavily dependent on the maturity and efficiency of the optical technology employed. Our evaluation revisits this approach and assesses the impact of the technology employed.

We believe that a careful design can deliver a fundamentally power-efficient all-optical solution that is reasonably robust to technology considerations. In this work, we argue for such an approach. Specifically, our proposed solution combines the following key features:

Wavelength-based routing. Within each optical router, the route followed by a packet depends solely on the wavelength of its carrier signal, and not on information either contained in the packet or traveling along with it. This allows us to adopt an all-optical solution for data transmission, where O-E/E-O conversion support at each router to figure out routes is unnecessary. Wavelength-based routing is a popular approach in optical LAN/WAN technology for this same reason [41].

Oblivious routing. The wavelength (and thus the route) employed to connect a source-destination pair is invariant for that pair, and does not depend on ongoing transmissions by other nodes, thereby simplifying design and operation.

Passive optical routers. Their routing pattern is set at design time, which allows for area and power optimizations not generally available to solutions that use dynamic routing. It also means no time lost in routing/arbitration decisions.

In our design, we construct an all-optical network layer, where each node has physical connectivity to all other nodes via static paths. Then, we replicate this network layer to increase bandwidth. To establish communication, we take a connection-based approach: a source node first establishes a logical connection with the destination node before transmitting data. A node may have concurrent connections to multiple nodes, on both the same and different network layers. Such a connection-based approach can benefit applications, by forming logical connections on the network layers that match the applications' communication pattern, thus minimizing global arbitration and streamlining data transfers. It also provides good isolation between exclusively communicating groups of nodes. We also propose techniques to hide and/or amortize connection setup overheads.

The flow of the paper is as follows: First, we construct a wavelength-routed, oblivious, all-optical network for CMPs, and describe its connection-based operation. Then, we evaluate the latency, cost, power consumption, and performance of the proposed network in the context of a 64-core, 256-thread shared-memory CMP design, and compare against other recent proposals for on-chip optical interconnects.

2. ARCHITECTURE

2.1 CMP Architecture

The CMP architecture of our study comprises 64 2-issue, in-order, 4-way multithreaded cores with their private L1 i- and d-caches. Each core is augmented with 4-way SIMD support, providing 16 GFLOP/s peak performance at 4 GHz core frequency, for an aggregate peak CMP performance of 1 TFLOP/s. Cores are organized in clusters of four, and cores within each cluster share a L2 cache. The system further employs eight memory controllers, each providing access to one of eight cache-block-interleaved L3 cache + memory banks. Each controller can deliver up to 256 GB/s.[1]

The shared-memory system maintains coherence across L2 caches and lower-level L3 cache and memory, using a MESI-based snoopy protocol, and a pipelined split-transaction opto-electrical command/snoop bus, along the lines of Kırman et al. [15], that runs at processor frequency. Actual transfer of cache blocks takes place in the data network, which is the main focus of our study. In the following sections, we describe the design and operation of an oblivious, wavelength-routed, all-optical data network that connects the sixteen L2-cache nodes and the eight memory-controller nodes. Section 4.1 provides more details on the CMP architecture.

2.2 Network Substrate

In wavelength-based routing, the route a packet takes at each point in the network depends exclusively on the wavelength of its carrier signal. This is advantageous because it allows us to offer end-to-end optical data transmission, without having to undertake O-E/E-O conversions and buffering in order to route a packet based on its content. Moreover, oblivious routing dictates that a given source-destination pair always communicates via a predetermined wavelength, which does not depend on the ongoing transmissions between other source-destination pairs. It enables us to provide connectivity using passive optical routers on the network, based on preset microring resonators that will automatically route each wavelength on the right path to the destination.

Ideally, one could conduct a multi-dimensional design space exploration (topology, routing, etc.) to devise a network that simultaneously optimizes for cost, complexity, and performance. For simplicity, however, in this paper we pick a reasonable regular physical topology, and then work out a viable routing scheme that effectively provides those three characteristics. After some preliminary trials, we opt for a 24-node, two-dimensional torus. A two-dimensional torus is attractive because, as we will see later, it yields relatively simple routers and waveguide layout.

2.2.1 Wavelength assignment for oblivious routing

In oblivious routing, every source-destination pair must have an assigned wavelength through which to communicate. A trivial way to accomplish this is to employ as many wavelengths as the number of distinct source-destination pairs. This, however, not only is prohibitively expensive ($O(N^2)$ wavelengths, where N is the number of nodes), but also unnecessary. Indeed, Aggarwal et al. [1] prove that significant wavelength reuse is possible. Specifically, the number of wavelengths needed to support oblivious routing in a network with N nodes is $(\lceil \frac{N}{2} \rceil + 2)$ for $N = 4$ or $N \geq 6$, assuming that communication is one-to-one.[2]

[1] Preliminary estimations indicated, and later simulations confirmed, that this provisioning is adequate to support the bandwidth demand of the 64 cores.

[2] The authors also assume that each node is connected to a router through a pair of incoming and outgoing physical channels, in our case waveguides. However, the authors ignore the complexity of the routers and the connectivity between them. As we discuss later, our problem is more restrictive than this, since our network is physically constrained and communication pairs do share a physical medium often.

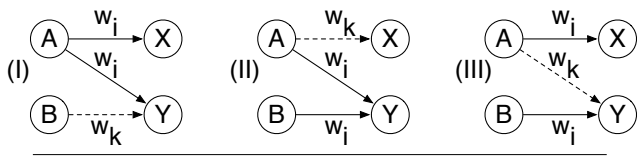

Figure 1. Three examples of wavelength reuse, where w_k must be different from w_i to ensure interference-free reception. For simplicity, the $B \rightarrow X$ wavelength assignment is not shown.

Tx \ Rx	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	7	5	4	3	2	1	6
1	1	1	1	1	7	0	5	4	3	2	6	1
2	2	2	2	7	1	0	5	4	3	6	2	2
3	3	3	7	2	1	0	5	4	6	3	3	3
4	4	7	3	2	1	0	5	6	4	4	4	4
5	7	4	3	2	1	0	6	5	5	5	5	5
6	5	4	3	2	1	6	0	0	0	0	0	7
7	5	4	3	2	6	1	1	1	1	1	7	0
8	5	4	3	6	2	2	2	2	2	7	1	0
9	5	4	6	3	3	3	3	3	7	2	1	0
10	5	6	4	4	4	4	4	7	3	2	1	0
11	6	5	5	5	5	5	7	4	3	2	1	0

I II III

Figure 2. Optimal wavelength assignment found using Aggarwal et al. [1] for oblivious routing in a 12-node wavelength-routed optical network. The (i, j) element in the matrix gives the wavelength that must be used when node i needs to transmit data to node j. 8 wavelengths (labeled 0 through 7) are required. Cases I, II, and III show examples of wavelength reuse.

Wavelength reuse requires a careful assignment. Figure 1 shows a few simple reuse scenarios. In all three examples, the communication pattern that we want to support is $A \rightarrow X$, $B \rightarrow Y$. In Case I, node A is configured to use wavelength w_i when transmitting to either node X or node Y. Thus, B must necessarily use $w_k \neq w_i$ when communicating with Y; otherwise, when A transmits to X and B transmits to Y concurrently, A's and B's signal would interfere at Y. In Case II, A and B are set up to use the same wavelength w_i whenever they communicate with Y. In that case, $A \rightarrow X$, $B \rightarrow Y$ can only be successful if A uses $w_k \neq w_i$ to communicate with X. Finally, Case III shows a case where A and B are set up to use wavelength w_i whenever transmitting to X and Y, respectively. In this case, A's wavelength to Y must be $w_k \neq w_i$, or else Y will receive information from A as a byproduct of A transmitting to X, which would interfere with B's.

In our work, we use the algorithm by Aggarwal et al. [1] to obtain the wavelength assignment for our 24-node system. Figure 2 is an example that shows a solution for a 12-node network using eight wavelengths (labeled 0 through 7). Element (i,j) in the matrix contains the wavelength that must be used when node i needs to communicate to node j. There is notable wavelength reuse: A source node may use the same wavelength to communicate to several nodes (Case I); multiple source nodes may use the same wavelength to communicate to the same node (Case II); and disjoint source-destination pairs may also use the same wavelength (Case III). Nevertheless, the wavelength assignment is such that one-to-one communication between distinct source-destination pairs can concurrently take place without conflict at any of the receivers. In our work, we set up communication pairs beforehand using a connection-based protocol.

2.2.2 Wavelength path layout

Once we derive a wavelength assignment for all source-destination pairs, we must determine the exact wavelength paths on the torus network, which comes down to determining the static routing configuration of the wavelength routers. A wavelength sourced from a

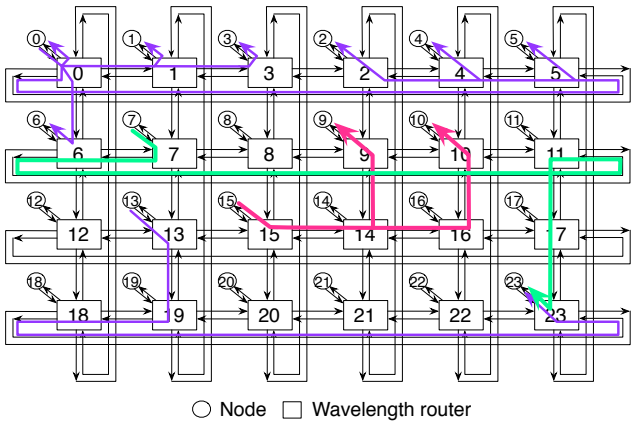

Figure 3. The 6×4 two-dimensional torus adopted in this study. The swapping of node labels 2-3 and 14-15 was done to help the genetic algorithm find a viable routing solution. Several routing paths from the actual solution are shown.

node should only reach the destinations designated by the assignment. The limited number of waveguide segments on the network makes it challenging to satisfy this routing constraint. In the worst case, it may not be possible to map the wavelength assignment. We must search the configuration space and find one which successfully routes the wavelengths from all nodes, necessarily without wavelength collisions within the physical medium between routes of disjoint source-destination pairs. Notice that, because we are using fully optical transmission, non-minimal routes are not necessarily a concern, and in fact they are attractive to the extent that they may enable a successful routing.

A manual search would be prohibitively time-consuming and error-prone. For this reason, we implement a genetic algorithm (GA) to find a viable configuration automatically. We solve the problem one wavelength at a time, observing that solutions for different wavelengths are independent of each other. Our GA begins with a set of randomly-generated configurations. In each initial configuration, for each source-destination pair that communicates through the wavelength, there is a random route originating from the source and represented with a list of router output ports. We place an upper limit on the route's hop count. The GA works its way toward a solution by applying a multi-objective fitness function (the lower the better), given by Equation 1. In the formulae, Boolean operators evaluate as in the C programming language.

$$\Phi = \Phi_1 + \Phi_2 \quad (1)$$

$$\Phi_1 = \sum_{i}^{Nodes} \left(\sum_{j}^{Nodes} C_{ij} \neq T_{ij} \right) + (Reconvergence_i \; ? \; 2 \; : \; 0)$$

$$\Phi_2 = \sum_{k}^{Routes} \mathrm{dist}(Dest_k, Target_k) + (Hops_k \neq MinHops_k) + (Cycle_k \; ? \; 2 \; : \; 0)$$

The first component Φ_1 provides a global view of the use of the wavelength under study. C and T stand for current and target connectivity matrix of the configuration, respectively: C_{ij} (T_{ij}) is set if and only if the wavelength being optimized optically connects (should connect) i to j, $i \rightarrow j$. A disagreement between C and T means that either connectivity or non-interference may be compromised, and a penalty is assessed in that case. The formula also assesses a penalty if two routes originating at i and going to different destinations reconverge (that is, they are disjoint at some point but

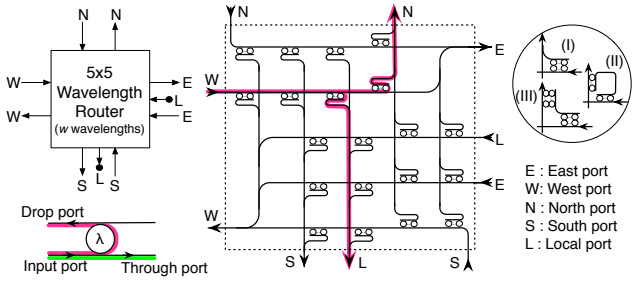

Figure 4. Passive wavelength-router implementation. A few alternative junction implementations are also shown (I, II, and III). On the lower left corner is a microring-resonator based filter.

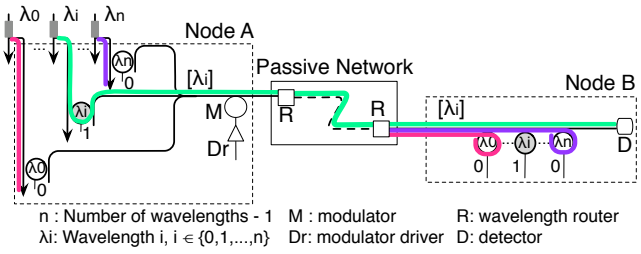

Figure 5. Simplified diagram of interfaces for transmitting and receiving data at end nodes. In the figure, node A is transmitting information to node B over wavelength λ_i.

later converge on the same router). This is wasteful because, even if it may be desirable for i to reuse the same wavelength to reach those destinations at different points in time, either route alone could be used to achieve such reuse.

The second component Φ_2 looks at individual routes (cells that are set in T). It rewards routes that are closer from successfully connecting an intended source-destination pair and have a hop count that is closer to the node distance.[3] It also penalizes routes that form cycles (i.e., they go through the same router twice), which are obviously useless. The GA terminates when it finds a solution for which $\Phi = 0$.

We initially ran our GA using XY labeling for the nodes of the underlying torus, but the GA was not able to find a viable solution for two of the wavelengths. Fortunately, we found that a few simple label swaps by hand (nodes 2 and 3, and nodes 14 and 15) were sufficient to make the GA produce complete, viable solutions. Among complete solutions from multiple GA runs that also minimize the number of hop counts, we picked the one which resulted in lower optical power and smaller propagation distances. (While we could have incorporated those features into the GA's fitness function, we were sufficiently satisfied with the solutions at hand that we did not pursue that for this paper.) Figure 3 shows the labeling of the nodes, as well as a few routes contained in the final configuration. For space reasons, we omit a full listing of all the routes.

Once the routes are determined, each router is customized at design time to satisfy these. We discuss router design next.

2.2.3 Wavelength-router design

We construct passive wavelength routers as depicted in Figure 4. Routing a wavelength from an input to an output port is accomplished via careful placement of a passive microring resonant to the wavelength at the appropriate input-output waveguide junction. In a junction, there are as many microrings as the number of wavelengths that are routed from the input to the output port.

A microring-resonator-based filter [16, 27, 37, 38] is reviewed in the lower left corner of Figure 4. It is an optical component whose geometry (e.g. radius, separation with the neighbor waveguides) determines the resonance wavelength(s) and coupling ratio of the filter. Light from the input port passes by the microring and keeps on going if its wavelength does not match any of the resonance wavelengths of the microring (off-resonance). If it does match a resonance wavelength of the microring (on-resonance), the light couples into the microring and then into a different output waveguide. (Depending on the coupling ratio, a fraction of the light may continue traveling on the original waveguide.)

[3] We factor in hop count minimization with the expectation of reducing the optical power expended along the path (primarily by the routers).

Notice that the routers may be completely different from one another, as needed to implement all the routes found by the GA. The resulting router designs are rather compact, with 2.06 microrings per junction on average (8 maximum). This is encouraging in terms of potential area and power savings.

2.2.4 Transmitter/receiver interface

Recall that, in the wavelength-assignment formulation described, nodes may not be transmitting/receiving to/from more than one node at any point in time. Consequently, we restrict each node to have a single input and a single output port from/to the network. On a transmission, the source must select and modulate on the node's assigned wavelength for the intended destination. Likewise, the destination must select and detect the same wavelength.[4] To accomplish this, we implement wavelength filters at both ends using an array of active microrings, with a microring per wavelength (Figure 5). We assume that separate waveguides supply each wavelength to nodes; this allows us to optimize for power. At the source filter, when there is no transmission, the microrings are off-resonance by default (no power supplied). Therefore, wavelengths pass by the rings and are not injected into the network. On a transmission, only one of the rings is supplied power to shift on-resonance, allowing the corresponding wavelength to couple into the input waveguide. At the destination filter, on the other hand, the microrings are on-resonance by default (no power supplied). Therefore, when there is no transmission, wavelengths couple into the rings and are not extracted from the network. On a transmission, only one of the rings is supplied power to shift off-resonance, allowing the corresponding wavelength to pass to the detector at the destination. Simple decoders can be used to drive the microrings. With this organization, tuning can be very fast. Notice also that tuning need only change when a source/destination node must move to another wavelength, in order to participate in a data exchange with a different node. Finally, we use a modulator and detector that can work with whatever wavelength is offered by the preceding filter.

2.2.5 Multiple network layers

The single network layer discussed so far enables one transmission at optical modulation rate from each node at a time. A cost-effective way to augment the network's bandwidth is to embed multiple *virtual* networks in the same set of waveguides, using spare wavelengths which may be available depending on the maturity of the technology. One possibility is to employ the technique proposed by Small et al. [27], which essentially places several wavelengths in the resonance band of a microring resonator. In that case, it is possible to route multiple bits of a message in parallel with little extra hardware: At each node, multiple modulators/detectors must tap

[4] This will require a protocol to have source and destination nodes tune to their assigned wavelength prior to transmitting data. We explain one such protocol in the next section.

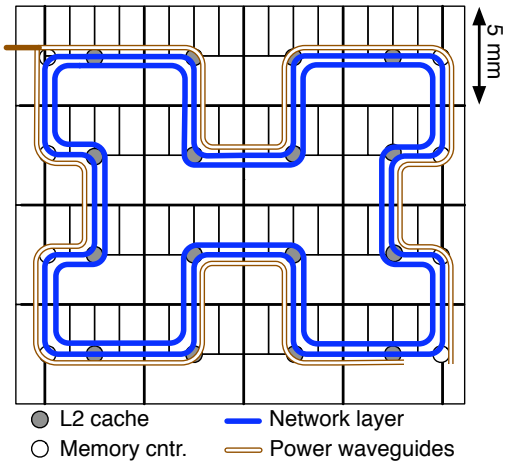

Figure 6. Concentric layout of multiple network layers.

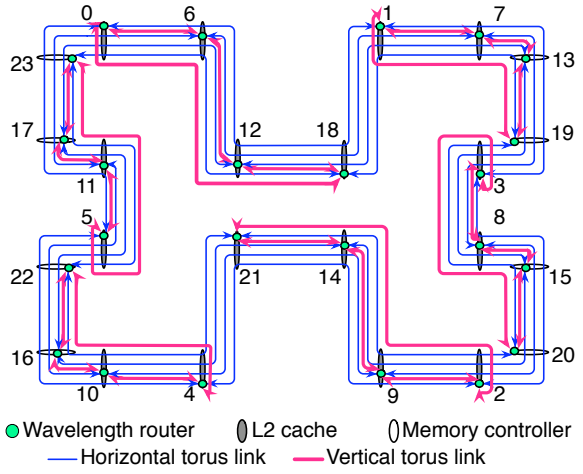

Figure 7. Circular layout of a torus network layer. The numbering is matched with the node/router numbering in Figure 3. For clarity, we draw the two unidirectional links between two routers as a bidirectional link.

separately on each of these wavelengths in order to inject/extract the bits of information; however at the routers and filters the only change comes from broadening the resonance band of their microrings, in order to correctly route such wavelength bundles.

Another way to achieve higher network bandwidth is simply to replicate the network. Notice that all such network layers must be laid out in a manner that minimizes waveguide crossings, which are a significant source of optical power losses. In our design, we lay out the network layers in a circular and concentric fashion (Figure 6), however each layer still conforms to a torus topology (Figure 7).

Multiple physical network layers can be used to transfer more bits of the same message, or alternatively, more messages. A node provides enough optical power from each wavelength to feed the maximum number of network layers that can transmit concurrently on that wavelength. This input light power is distributed to network layers based on demand by the active filters. In case multiple physical network layers transfer more bits of the same message, these layers share a source-side filter in a node. After the filter selects the appropriate wavelength based on the target destination, common to all of these layers, the light at the filter output is split among the individual layers for modulation.

Unless otherwise stated, in the network operation below, we assume layers are used to transmit different messages.

3. NETWORK OPERATION

In our design, a source-destination pair must tune to the assigned wavelength before the actual transmission takes place; and only one source node may transmit to a particular destination node at a time on each layer–which essentially requires arbitration for a receiver. In this section, we describe a distributed protocol that not only ensures that these constraints are satisfied for data transmissions, but also keeps connections alive for as long as possible, so that source nodes can transmit later on without incurring additional setup delays.

A source-destination pair may have at most one open connection at any given point in time (i.e., the multiple network layers cannot be used to open multiple connections simultaneously for the same pair.) During the lifetime of such a connection, the destination node's receiver remains tuned to the assigned wavelength; however, the source node's transmitter may time-multiplex over different connections, by selecting the right wavelength at each point in time. (Of course, the node can also establish different connections on different network layers.) A connection lasts until it is closed by the destination node–for example, when it needs to engage in communication with another node on that layer.

The connection protocol is supported by a few dedicated optical network layers via point-to-point transactions between nodes. The protocol uses time slotting to ensure that all potential source-destination pairs have a chance periodically to exchange protocol messages.[5]

In the following sections, we describe the connection protocol, network layer selection policy, the operation of the protocol network layers, and finally the hardware support at the network interfaces.

3.1 Connection Protocol

For simplicity, we first describe the connection protocol assuming a single network layer between nodes.

A source node issues a connection request to a destination node if it finds it is disconnected to that node on a data transfer attempt. In the simplest case, the destination directly acknowledges the connection request if its receiver is disconnected as well. If, on the other hand, the receiver is involved in a connection, the destination node first needs to break that connection, and wait for any scheduled transmissions by the previous owner to complete (signaled via a break acknowledgement message) before sending the connection acknowledgement to the new connection requester. Once the connection requester receives an acknowledgement to the connection request, it can start sending data at any time, without consideration of other nodes. The previous owner, on the other hand, would need to establish a new connection before any future data transmission to the same destination node.

The full protocol is slightly more involved, due to issues that arise from the non-atomic nature of the connection setup process. Below we briefly discuss these issues.

Connection request races: Multiple connection requests can compete for a receiver in a node. This destination node serves as the serialization point for all such requests: upon accepting the first request, it will not accept further requests until the first one

[5] Whereas the overhead of time slotting (as an alternative to connection-based communication) would be prohibitively high to manage data transmission itself, the small size of the connection protocol messages, and the relative infrequency of these transactions, makes it possible to use it here.

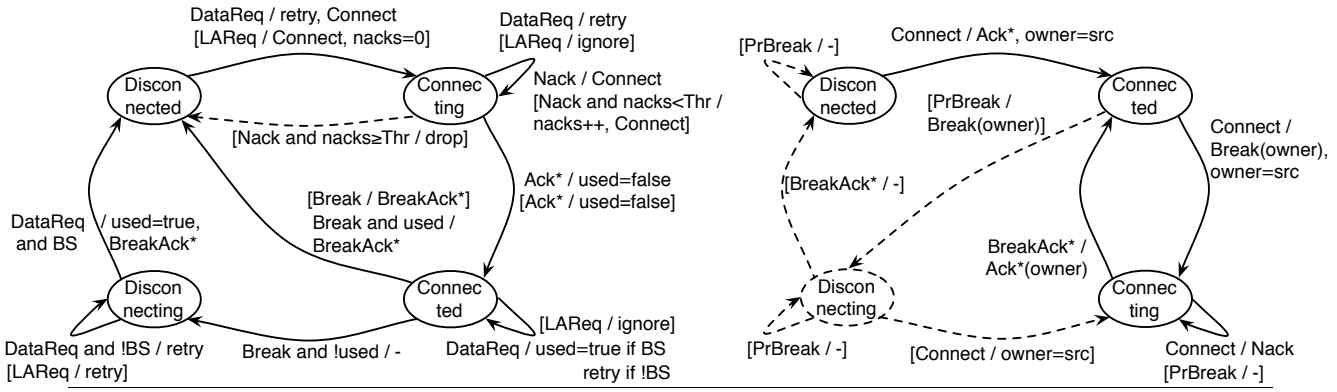

Figure 8. Protocol state diagrams for Tx-side (left) and Rx-side (right). A slash (/) separates a protocol message and the action taken in response. BS and LA stand for available transmitter buffer space and lookahead, respectively. An asterisk (*) next to a message indicates that the connection layer information is included in the message. The information in parentheses (()) specifies the destination of the message. Protocol extensions for proactive break and connection-lookahead support are encapsulated in brackets ([]) and use dashed transition lines. Notice that, in the Tx-side diagram, statements in brackets are for lookahead initiated connections / connection requests.

is resolved from the node's point of view. Requests that find the receiver busy connecting are nacked and retried later.[6]

Forward progress: In order to avoid receiver ownership to ping-pong between nodes without actually being used, a connection owner delays its break acknowledgement until the connection is used at least once. Because the connection is established in response to a data transmission attempt, it is guaranteed that the connection will be used at least once.

Scheduled transmissions: At the time a connection owner receives a break request, it may have scheduled data packets on this connection in the transmitter's buffer. The break acknowledgement is piggybacked on the last of these packets. The connection is closed from the point of view of the owner. If there are no scheduled transmissions, and if the connection has been used at least once, the break request is directly acknowledged on the protocol network layers; if the connection has not been used, the break acknowledgement will be piggybacked on the first data packet to be scheduled on this connection. As a result, a break acknowledgement for a connection always reaches the destination after all packets scheduled on this connection.

Reply-request races: A connection acknowledgement and a subsequent break request for the same connection, could potentially overlap in the network. The protocol network layer's interface guarantees that the reply is delivered before the request (Section 3.3). This simplifies the protocol.

Similarly, a break acknowledgement and a subsequent connection request for the same source-destination pair can overlap in time. Although ordered delivery of the reply and the subsequent request to the same node is guaranteed on the protocol network layers, recall that a break acknowledgement can be delivered over a data network layer, possibly after the connection request. Fortunately, this does not constitute a problem, because the connection request will find the receiver busy connecting and it will be nacked.

Figure 8 summarizes all protocol actions in two state diagrams– one for the transmitter side and one for the receiver side. The diagram also shows protocol extensions to support a few performance optimizations that we describe later.

3.2 Network Layer Selection

The main challenge with multiple network layers is to decide on which layer to establish a new connection.

On a connection request to a node, the node applies a selection policy to choose a network layer on which to establish the connection. This may result in evicting an existing connection to the receiver from another transmitter. This is conceptually similar to victim selection in a cache replacement policy. The selection policy that we implement in this work is LRU; we tried others (round-robin, random, etc.) and found their performance to be at most as good as LRU's. Once a layer is selected, the connection protocol is executed for this layer. The layer information needs to be communicated in the relevant protocol transactions, which we already include in the state diagram in Figure 8.

A data transmission necessarily takes place on the layer with an established connection to the destination node. A node keeps track of connection status to each destination separately. The status information include the layer id.

Notice that, in the case of unordered delivery of break acknowledgement and a subsequent connection request from the same source to the same destination, a different layer may be selected for the new connection, even as the previous connection is still in the process of being disconnected on its corresponding layer. Since the older connection has already been severed from the source's point of view, this corner case is harmless.

3.3 Protocol Network Layers

A node transmits protocol messages on a few dedicated network layers, as described in Section 2.2. A deterministic and periodic schedule dictates to which destinations a node can transmit connection protocol messages at each time slot and on each network layer. Thus, a node can send message to a particular node every N/M time slots on a specific layer, where N is the number of nodes, and M is the number of protocol network layers. Node schedules are shifted from one another to ensure that only one node attempts to transmit to a particular node at each point in time. A time slot should be long enough for a protocol message to reach its destination node, in order to guarantee in-order delivery of messages across time slots (which is not guaranteed by the network topology). It should also accommodate the tuning delays.

A node processes incoming protocol messages in a non-blocking and pipelined fashion.[7] It places outgoing protocol messages in an output buffer per layer with an entry for each possible destination. Messages wait here for their time slots. Note that, there may be

[6] There are different techniques to avoid starvation for nacked requests [9].

[7] Multiple cycles in a time slot allow for more relaxed port throughput requirements in order to process the incoming messages in a slot.

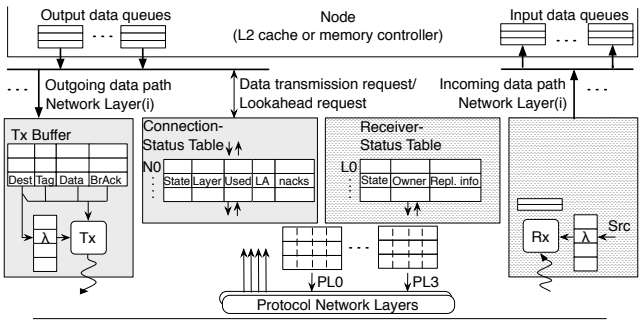

Figure 9. Node's interface to data (L) and protocol (PL) network layers. LA is short for lookahead flag.

multiple protocol messages waiting for a time slot. However, because of the way the protocol and its network layers work, there can be no two messages of the same type (connection request, connection ack/nack, break request, and break ack) for the same destination node. Therefore, an entry has separate fields for the four protocol-message types. Protocol-message arbitration ensures that reply-request races (Section 3.1) for the same destination are properly ordered. Because very little information needs to be stored in each entry, the overall storage overhead is small.

3.4 Hardware Interface

Figure 9 depicts a node's interface to the optical network. A connection status table tracks the outgoing connections to each node, while a receiver status table holds connection information for the receiver on each network layer.

A data transmission attempt first checks the connection status for the destination. If a connection exists, the data is placed into the transmitter buffer. (If the transmitter buffer is full, the data transmission is retried later.) On the other hand, if the connection is not ready or does not exist, the transmission attempt is delayed until the connection is established.

Newly generated protocol messages are scheduled for transmission on a protocol network layer. They wait for a proper time slot in the layer's outgoing buffer (Section 3.3).

Protocol messages received from the protocol network layers are processed using either connection or receiver status table depending on whether the message is directed to an outgoing or incoming connection.[8]

On an actual data transmission from the FIFO transmitter buffer on a network layer[9], the transmitter first tunes to the wavelength for the destination, according to a wavelength mapping table, and then transmits the data. Note that the node may have multiple connections on this layer that use the same wavelength (similarly to Case I in Figure 2). In this case, a data transmission will reach all receivers for these connections. A receiver, therefore, must check the intended destination of the data packet before delivering it to the node. We implement nonblocking message delivery, by providing enough buffering to accomplish bandwidth "impedance matching." Additionally, note that the command/snoop phase for a memory request preceding any data transmission guarantees there is available buffering at the receiving node. For other implementations where buffer space at destination is not guaranteed, or the delivery rate does not match the receive rate, credit-based flow-control support can be easily added by leveraging the protocol network layers to communicate the credits.

[8] Note that, for correct operation, a break-request processing must see the simultaneous connection use from the node.

[9] If a set of network layers is used to transmit bits of the same message in parallel, these layers can share the same transmitter buffer.

3.5 Optimizations

Here we discuss a number of possible optimizations, and any protocol or hardware changes when required.

3.5.1 Lookahead connection requests

In the basic protocol, a node requests a connection only on an actual data transmission attempt. It is possible for a node to act earlier for establishing a connection in anticipation of a future data transmission. The hope is to hide connection establishment latency. There are several circumstances when we apply this feature:

– A memory controller issues a lookahead request when it sends a read request to the L3 cache and memory, so it can relay the data promptly to the requesting node once it returns.

– A cache with E or M state in a snoopy MESI-based coherence protocol issues a lookahead request concurrently to sending its snoop response, in preparation for the data transmission that will follow shortly.

– On a cache line eviction, a lookahead request is issued in parallel to sending the write-back request through the command/snoop bus (this assumes that the cache knows the memory bank's location in the network).

Figure 8 shows the protocol changes on the transmitter's side needed to support this feature; the protocol in the receiver's side is unchanged. Note that, because it is not guaranteed that a connection established through lookahead is going to be ever used, the protocol directly breaks such a connection on a break request if no transmissions are pending. Also, we drop a lookahead request after being negative acknowledged for a certain number of times.

3.5.2 Connection-aware cache coherence

In the context of the coherence protocol, upon a read/read with write intent request by a node for a cache line in Shared state at one or more remote nodes, a subset of the sharers may already have an established connection with the data requester node (necessarily on different network layers). We propose that one of these sharers leverages the existing connection and provides the data. Such sharer nodes should include the fact that they have an open connection in their snoop response, so that the coherence controller may consider them as preferred suppliers.

3.5.3 Proactive connection break requests

In the basic protocol, a node breaks a connection only in response to connection requests by other nodes (Section 3.1). We extend the protocol so that a node can proactively initiate the break of one of its incoming connections. The hope is to hide the latency of the break handshake on a subsequent connection establishment. On a connection request that replaces a proactively broken connection, the node need not resend a break request; it simply waits for the break acknowledgement from the previous receiver owner. Figure 8 shows the protocol changes on the receiver's side needed to support this feature; the transmitter's side does not change.

In our implementation, after a node processes a connection request, possibly selecting an existing connection to evict, it opportunistically applies the LRU policy to select an additional layer whose used connection, if any, will be broken proactively. Depending on the particular network configuration, some nodes may not trigger proactive break at all. We describe this case in our experimental setup.

4. EVALUATION

This section analyzes the power and performance of the proposed network, in the context of a 64-core 256-threaded CMP targeted

Processor Core	
Frequency	4 GHz
Issue	2, in order
Int ALUs/Branch units	2
Ld/St units	1
Mul/Div units	1
FP ALU/MUL units	4-way SIMD
FP Div units	1
Write-buffer entries	16
Store forward delay	2 cycles
Branch min. cycles	5
Branch predictor, (Hybrid of GAg + SAg)	13-bit GHR, 2,048 10-bit BHRs 8,192-entry chooser
BTB/RAS entries	2,048/32
IL1/ DL1 size, associativity	32 KB, 4-way
IL1/ DL1 access latency	2 cycles
IL1/ DL1 block size	64 B
DL1 writeback policy	Write-through
DL1 MSHR entries	16

Memory Subsystem		
	L2 cache	L3 cache
Caches	16	1
Cache size	2 MB	64 MB
Cache banks	8	8
Cache associativity	16 way	16 way
Cache access latencies	9 cycles	45 cycles
Cache writeback policy	Write-back	Write-back
Cache block size	64 B	64 B
MSHR entries	64	128
Coherence protocol		MESI
Address-network snoops per cycle		8
Address-network snoop-request latency		8
Address-network snoop-response latency		6
L3/Memory controllers		8
L3/Memory controllers' bandwidth		8×256 GB/s
Memory latency		100 cycles

Table 1. Summary of the modeled system. In the table, GHR, PHT, BTB, MSHR, and RAS stand for global history register, pattern history table, branch target buffer, miss status holding register, and return address stack, respectively. Cycle counts are in processor cycles. Bus latencies are contention-less latencies.

for 32 nm technology node, compared against alternative designs proposed in the literature.

4.1 Experimental Setup

This section provides more details on the evaluated CMP architecture, whose overall organization we highlighted in Section 2.1. We conduct our evaluation using a cycle-accurate execution-driven simulator based on SESC [23]. Latencies and occupancies of all structures are modeled in detail. Table 1 summarizes core and memory-system parameters.[10] We use CACTI 5.3 [28] to obtain cache latencies. We assume a 450 mm^2 die area, which is in line with server-oriented CPUs.

Following common practice for SPLASH-2 applications, we use a reduced L2 cache size of 256 KB to compensate for the applications' small working sets [32]. Still, we use the latency of a full-size 2 MB cache.

The banked L3 cache is on a separate 3D layer; a 3D interconnect provides 256 GB/s bandwidth from each bank. 2 TB/s of off-chip memory bandwidth (256 GB/s per memory bank) is provided through optical channels. (Optical access to memory arrays reduces the memory latency as well [3, 29].)

We evaluate several configurations of the proposed network. We also compare them against two optical networks modeled after previously proposed architectures [15, 29].

The aggregate network bandwidth in all configurations is set to 6 TB/s (each node can receive 64 bytes every cycle [29]), to serve as an equalizing parameter to make meaningful power and performance comparisons. For all configurations, we assume support for up to 64 wavelengths [29], 10.45 ps/mm light propagation delay [15], and 32 Gbit/s optical data rate [6, 19, 20, 39]. The same opto-electrical command/snoop bus is used in all configurations [15], which is excluded from the power figures in order to isolate the contribution of the data network, which is the subject of our study.

Oblivious, wavelength-routed network (*Oblivious*): This is our proposal. We evaluate three different configurations based on how they use the multiple network layers. All configurations require 16 data and 4 protocol network layers, and a total of 56 wavelengths.

[10] In our simulation infrastructure, 4-way SIMD processing is emulated by issuing up to 4 consecutive independent floating-point add/sub/mult instructions with ready operands in one cycle. Any intervening instruction in the code not of one of these types terminates the SIMD bundle.

Oblivious Data Networks			
Data network layers	16x1	8x2	4x4
Virtual layers per network layer	4	4	4
Network bandwidth (TB/s)	6	6	6
First-word transmit cycles[a]	4	2	1
Network latency[b] / Delivery	1-3 cycles / 1 cycle		
Conn. break replacement policy	LRU		
Transmitter buffer entries	4		
Protocol network layers	4		
Time-slot duration	4 cycles		
Arbitration cycle (in a time slot)	4^{th} cycle		

[a] Includes E-O delays for first-word bits

[b] Includes 4 FO4 + light propagation + O-E delays

Table 2. Evaluated configurations of the proposed network.

In *Oblivious-16*, each message is transmitted over a single network layer, whereas in *Oblivious-8* (*Oblivious-4*), each message is transmitted over two (four) layers. Table 2 summarizes the main parameters. The average (max) path length is 31.5 mm (67.5 mm). In all configurations, we match the receive bandwidth in a node through four 128-bit delivery ports, each serving a subset of the data network layers. Unless otherwise stated, all optimizations described in Section 3.5 are employed.

We employ a LRU replacement policy for connection breaks. Notice, however, that in *Oblivious-16* a memory node can simultaneously accommodate connections from all sixteen cache nodes on non-conflicting layers. This effectively eliminates the need for proactive breaks at memory nodes. Also, a cache node can simultaneously accommodate connections from all eight memory controllers on non-conflicting layers (though those still may conflict with connections from other cache nodes). Consequently, in *Oblivious-16*, we use a static node-to-layer mapping in memory and cache nodes, to promote an even distribution of cache-to-memory and memory-to-cache connections across network layers at both sender and receiver sides. This results in good load balancing and minimal number of connection setups.

Optical crossbar with broadcasting (*Xbar-Bcast*): This optical network is modeled after the data network of Kırman et al. [15]. Its optical fabric essentially implements a full crossbar on a set of waveguides that loop around all nodes. Each source node has

SPLASH-2	Problem size	SPLASH-2	Problem size
Barnes	64k particles	Radix	1,024 radix,
Cholesky	tk29.O		4M integers
FFT	256k points	Raytrace	balls4
LU	1,024×1,024 matrix	Water-NSq	4,096 molecules
Ocean	514×514 ocean	Water-Sp	4,096 molecules

Table 3. Applications simulated and problem sizes.

exclusive set of wavelengths on which it broadcasts data packets. All other nodes tap into the data, but only the true destination processes it. As a result, the network operation does not require global arbitration. However, this comes at the expense of O(N) number of receivers per node. To mitigate the resulting cost, the authors suggest a hierarchical opto-electrical organization where the optical fabric serves several (electrical) switches at the top level, and each switch serves multiple nodes at the lower level. We perform a design space exploration to determine the organization that provides the best power-performance trade-off for our target bandwidth of 6 TB/s. The resulting configuration has 6 switches on the bus, each capable of transmitting 4 messages using 2 wavelengths per message in a waveguide, and a flit size of 64 bytes. A total of 48 wavelengths are used.

Optical crossbar with arbitration (*Xbar-Arb*): This network is modeled after the data network of Vantrease et al. [29]. It implements a crossbar as well, however this time each node has an exclusive set of waveguides to receive data that loop around all other nodes. The network operation requires arbitration for transmitting to a node which is accomplished through token-based all-optical arbitration. The network has O(N) transmitters per node. The target bandwidth is reached with a flit size of 64 bytes, using 64 wavelengths in one data waveguide. We estimate a 5-cycle latency for a token to circulate around the nodes for our layout and optical parameters. We also assume nodes request one token at a time.

We would like to point out that, because our target system is implemented in an earlier technology node than the one assumed in [29], our results do not necessarily represent the behavior of the system at the scale proposed in that work.

4.2 Applications

We resort to the SPLASH-2 applications [32] compiled into MIPS binaries with -O3 optimization level, and use the data input sets provided in Table 3. For the 256-threaded executions, we tried to scale the default data sizes (suggested for up to 64 threads) to account for the four time increase in thread count. We fast-forward the initialization regions (at which point we start modeling timing and collecting statistics) and run them to completion. Our simulation infrastructure currently does not support 256-threaded executions of Volrend and Radiosity. FMM is also excluded due to its long execution time–in any case, it is not sensitive to network performance [15, 29].

4.3 Power Evaluation

We estimate the maximum on-chip electrical and optical power consumption of *Xbar-Bcast*, *Xbar-Arb*, and *Oblivious* networks. Based on the optical power requirements, we also estimate the power of the off-chip laser required in each configuration. We first describe our methodology and then discuss the results, provided in Table 7.

4.3.1 On-chip electrical power estimation

We break down the on-chip electrical power consumption into four categories. In all cases, we assume maximum electrical activity ($\alpha = 1$).

Switches/(De)multiplexers: *Xbar-Bcast* employs electrical routers, whereas the all-optical configurations only use (de)multiplexers at the network interfaces. Table 4 lists the count, type, and size of these components. We account for the data buffers at network interfaces along these structures. We use Orion 1.0 [31] to estimate their maximum power consumption.

	Electrical Switches/(De)multiplexers
Xbar-Bcast	6 4x5 routers, 512b, 4-entry input, 1-entry output buffers 6 21x4 routers, 512b, 4-entry input, 1-entry output buffers
Xbar-Arb	24 1x23 demux, 512b, 1-entry output buffers 24 1x1 mux, 512b, 2-entry input buffer
Oblivious-16	24 1x16 demux, 512b, 4-entry output buffers 96 4x1 mux, 128b, 2-entry input buffers
Oblivious-8	24 1x8 demux, 512b, 4-entry output buffers 96 2x1 mux, 128b, 2-entry input buffers
Oblivious-4	24 1x4 demux, 512b, 4-entry output buffers 96 1x1 mux, 128b, 2-entry input buffers

Table 4. Electrical switches/(de)multiplexers in the evaluated networks.

Wiring: *Xbar-Bcast* consumes additional wiring power at the links that connect nodes and routers. We estimate wiring power assuming 280 nm global-wire pitch [22] and ITRS device-performance and interconnect projections [14] for power-performance optimized repeatered global wires using the methodology in Ho et al. [13]. Leakage power per repeater is assumed to be 1 μW [15].

Transmitters/Receivers: Following the methodology in [12, 24] and assuming a conservative 100 fF capacitance for driver plus modulator, as well as a 2.4 fF photodetector capacitance (as reported in [6]), we estimate 40.5 μW/Gb/s and 147 μW/Gb/s power at 32 nm technology node for a single transmitter and receiver, respectively. This corresponds to 1.3 mW transmitter power and 4.7 mW receiver power at 32 Gb/s optical data rate. Component counts are provided in Table 5. Power estimations consider busy components only. Notice that *Xbar-Bcast* and *Xbar-Arb* have a large number of receivers or transmitters. *Oblivious* networks, on the other hand, have just enough components to satisfy the target bandwidth, with a few extra ones in the protocol network layers.

	Txs (Busy)	Rxs (Busy)	Microrings	
			Switching (Busy)	Passive
Xbar-Bcast	(1,536)	(7,680)	(1,536) mod.	7,728
Xbar-Arb	35,328 (1,536)	(1,536)	35,328 (1,536) mod. 1,152 (600) filter	1,536
Oblivious-16	(1,920)	(1,920)	(1,920) mod. 11,504 (960) filter	18,879
Oblivious-8	(1,920)	(1,920)	(1,920) mod. 8,816 (768) filter	18,879
Oblivious-4	(1,920)	(1,920)	(1,920) mod. 7,472 (672) filter	18,879

Table 5. Component counts in the evaluated networks. Counts not in parentheses are total component counts, while counts in parentheses show the maximum number of simultaneously active (busy) components. When both counts coincide, only one figure in parenthesis is provided. In the table, mod. is short for modulators.

Microrings: Active microrings also consume power. Using the methodology in [17], we estimate dynamic modulation energy to be 82 fJ/bit, assuming $V_{on} = 2$ V, $V_{pp} = 4$ V, $I_{on} = 50$ μA [20, 36], and modulator capacitance of 10 fF [17]. In steady active state, a microring consumes 100 μW power (in line with [27]). Accordingly, we estimate busy ring-resonator-based modulators' and active microring filters' power consumption using the component

Modulator insertion loss / pass loss (dB)	0.1 / 0.01	[10, 38]
Detector insertion loss (dB)	0.1	[3]
Active ring drop / through / pass losses (dB)	1 / 0.1 / 0.01	[4, 10, 38]
Passive ring drop / through / pass losses (dB)	0.5 / 0.01 / 0.01	[10, 34]
Waveguide propagation loss (dB/mm)	0.1	[5]
90° Waveguide bend loss, 2μm radius (dB)	0.02	[30]
90° Waveguide bend loss, $>6.5\mu$m radius (dB)	0.005	[33]
90° Waveguide intersection loss (dB)	0.12	[35]
Waveguide split excess loss / merge loss (dB)	0.04	[21]
Layer-to-layer coupling loss (dB)	1	[11]
Fiber-to-waveguide loss (dB)	0.5	[16]
Laser efficiency (%)	30	[2]
Detector sensitivity (μW)	10	[6, 24, 40]

Table 6. Loss values used for unit components/events, compiled from recent literature.

counts provided in Table 5.[11] Passive microrings do not consume electrical power.

4.3.2 On-chip optical power estimation

Emitted light power from the light source must be large enough to ensure that sufficient optical power reaches detectors at the end of the light paths. Along the way, a light beam encounters various structures such as splitters, merges, bends, crosses, couplers, off- and on-resonance passive or active microrings, modulators, detectors, etc. (e.g. see the light beam in Figure 4). In practice, all such interactions, including the propagation in waveguides, incur losses in the optical power.

We perform a detailed power analysis for each evaluated optical system. We compile and use state-of-the-art or projected component efficiencies from recent literature on the most common high-index-contrast silicon-on-insulator (SOI) photonic technology. We list corresponding unit losses, in dB, in Table 6.

In our light-path model, emitted light is first coupled on chip and then demultiplexed into two sets of power waveguides that are each routed to half of the nodes (Figure 6) on a separate optical layer, avoiding crossings between power and data waveguides.

For each node, we cap the number of data messages that can be transmitted simultaneously on all layers for a particular wavelength. We cap it at the number of possible destinations on that wavelength for that node. Thus, we need only provide enough optical power to transmit as much data using that wavelength (plus any power needed to transmit connection protocol messages using that wavelength).

We provide the network model with the lengths and bend, merge, cross counts for all waveguide segments throughout. We carefully estimate the count and type (splitting, fully coupling, or passing) of the microrings in every junction at routers by processing the full routing pattern of the network (Section 2).

A protocol network layer uses only a subset of the available wavelengths and light paths based on the time-slot schedule (Section 3.3). We estimate the optical power of a protocol network layer excluding the components for the unused wavelengths and light paths.

Starting from the end detectors and walking the light path in reverse direction to that of light propagation, we find the system loss for a particular wavelength up until the off-chip light source. We estimate the corresponding optical power using this loss value and the optical power required at a detector (Table 6). Then, we sum the optical powers for all wavelengths.

We apply similar methodology to the other networks.

[11] A protocol network layer uses only a subset of the wavelengths and light paths, based on the time-slot schedule (Section 3.3). We exclude the components for the unused light paths.

4.3.3 Laser power estimation

Laser sources, which provide light to the on-chip optical network, consume electrical power to generate the required optical power. Among light-source alternatives, we assume off-chip laser(s). We assume 30% efficiency for a laser [2] when converting electrical power to optical.

4.3.4 Discussion

	On-chip Electrical Power (W)				On-chip Optical	Laser
	Switch	Wiring	Txs/Rxs	μRings	Power (W)	Power (W)
Xbar-Bcast	39.24	60.40	38.12	4.01	0.91	3.04
Xbar-Arb	14.37	-	9.22	4.07	90.44	301.45
Oblivious-16	14.26	-	11.52	5.11	6.13	20.45
Oblivious-8	8.05	-	11.52	5.09	7.81	26.03
Oblivious-4	5.01	-	11.52	5.08	8.71	29.04

Table 7. On-chip power consumption breakdown for all evaluated networks. We assume maximum activity factor ($\alpha = 1$) for the electrical components. The off-chip laser source's required power is also shown.

Table 7 shows that the proposed network is the only one among the evaluated configurations that can support very high bandwidth with reasonable electrical and optical power consumption. While *Xbar-Bcast*'s electrical components consume a lot of power, *Xbar-Arb* is very power-hungry on the optical front.

Xbar-Bcast's power is larger than the one reported in [15]. The reason is the very different bandwidth support of the two configurations–our model has higher network operation frequency and optical data rate, wider flit width, and larger number of wavelengths.

Through a sensitivity study, we identified that *Xbar-Arb*'s power consumption is most sensitive to waveguide propagation and off-resonance active microring and modulator losses, followed by on-resonance active microring and modulator losses. The reason is twofold: First, in both data and arbitration parts, the critical paths circulate around nodes twice, once for power distribution and once on actual data or arbitration waveguide. Second, there are many active microrings on the data and arbitration waveguides. If we use two power distribution branches for the data waveguides as in *Oblivious* networks (Figure 6) instead of looping as described in [29], the optical power for *Xbar-Arb* drops to 46.50 W from 90.44 W, and the laser power drops to 155 W. Even then, the optical power remains high.

The losses that we use are aimed at the most commonly used SOI-based strip waveguides, which allow for very sharp bends and therefore compact designs. The lowest propagation losses for these waveguides are in the 0.1-0.2 dB/mm range [5]. Alternatively, ridge waveguides have much smaller propagation losses around 0.02 dB/mm. However, they have large pitches, and they also suffer from significantly higher bending losses [5]. (Bending losses can be curbed by using a large $200-600$ μm bending radius, but this may make implementation significantly more challenging, especially when trying to construct compact microrings.) Nevertheless, if we optimistically assume their more aggressive propagation loss across the board, and ignore all the negative aspects of such a technology, the optical power of *Xbar-Arb* could potentially drop to 3.93 W with the original power distribution, and to 3.27 W with the two power distribution branches that we propose for *Xbar-Arb*. (In that optimistic scenario, the proposed network's optical power also drops significantly, to 1.77 W, 2.25 W, and 2.5 W for *Oblivious-16*, *-8*, and *-4*, respectively.)

We conclude that our proposal *Oblivious* is by far the most power-efficient configuration of all the ones evaluated. It is able to deliver 6 TB/s of aggregate on-chip network bandwidth while exhibiting moderate on-chip power consumption and requiring a

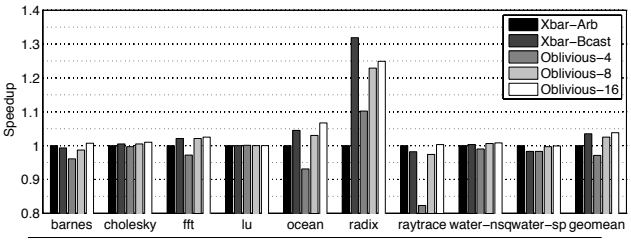

Figure 10. Performance of the optical networks relative to Xbar-Arb. In all cases, aggregate network bandwidth is 6 TB/s.

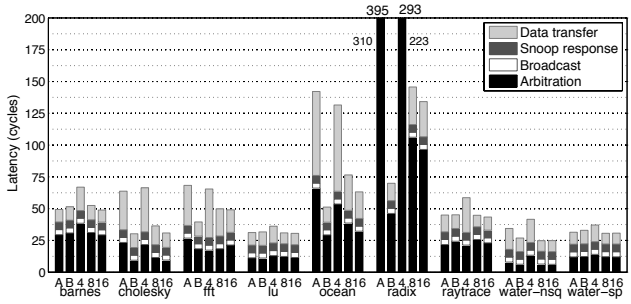

Figure 11. Average latency breakdown of a memory operation in the address network and each of the evaluated data networks. Labels A, B, 4, 8, and 16 correspond to Xbar-Arb, Xbar-Bcast, Oblivious-4,-8, and -16 configurations.

relatively thrifty laser source. We also observe that *XBar-Arb* is the configuration whose power consumption is most sensitive to assumptions about optical devices. In the next section, we look at how this bandwidth translates to performance in each configuration.

4.4 Performance Evaluation

Figure 10 compares the performance of *Oblivious* to those of *Xbar-Arb* and *Xbar-Bcast*. Speedups are relative to *Xbar-Arb*. Recall that all configurations have the same 6 TB/s aggregate bandwidth. The results show that, on average, all networks are capable of exploiting their aggregate bandwidth to a similar extent. This means that *Oblivious* configurations offer the best power-performance tradeoff among the ones studied, as they yield significant power savings (Section 4.3). Among the *Oblivious* configurations, *Oblivious-4*'s performance takes a modest hit sometimes. As we discuss later, this is due mainly to the fact that nodes in such a configuration cannot keep sufficiently many open connections, resulting in larger connection overheads.

Figure 11 shows the average number of cycles a memory operation spends in each data network. For reference, we also provide the average latencies for phases on the command/snoop bus. From left to right, the bars for each application correspond to the *Xbar-Arb*, *Xbar-Bcast*, and *Oblivious-4,-8,-16*. We observe very low data transfer latencies on all networks, except in the cases of *Ocean* and *Radix*. This means that all configurations are adequately equipped in terms of bandwidth, resulting in small levels of congestion. In the cases of Radix and Ocean, the increased latencies are a consequence of bursty requests and high contention for the same addresses across threads of the applications, which taxes mainly the address network.

4.5 Performance Analysis

We conduct additional experiments to gain more insight into the operation of the proposed design.

The plots in Figure 12 break down the true data transmission requests (i.e. excluding lookaheads) by a node based on initially

	Oblivious-8			Oblivious-16		
Appl.	**Setup**	**Lifetime (K)**	**Uses**	**Setup**	**Lifetime (K)**	**Uses**
Barnes	38/39	3.2/1.7	3/2	35/50	51/45	25/37
Cholesky	52/45	3.5/2.9	3/3	36/53	112/131	47/73
FFT	50/50	0.8/1.1	2/6	36/48	128/198	130/742
LU	39/38	145/16	6/3	35/43	16127/8449	398/616
Ocean	61/51	1.5/0.9	3/7	39/60	74/41	61/196
Radix	78/48	0.3/1.6	3/10	35/47	384/1450	844/7455
Raytrace	42/39	4.0/0.5	22/2	35/42	77/14	247/18
Water-NSq	47/40	4.0/4.7	7/12	34/42	581/222	420/315
Water-Sp	39/42	2.7/5.4	4/9	35/50	339/655	166/771

Table 8. Average connection statistics, provided separately for L2 caches and memory controllers (L2 cache/Mem Cntr). Connection setup cycles and lifetimes are in processor cycles.

encountered connection state. We show separate plots for memory controllers and L2 caches because of their different characteristics. The three bars for each application show the results for *Oblivious-4,-8,-16* from left to right. A request is classified as *Hit* if the connection exists and there is free space in the transmitter buffer, *FullBuffHit* if the connection exists but there is no buffer space, *Miss* if the connection is unowned, after which connection establishment is initiated, and lastly *HalfMiss* if the connection is currently being established.

Figure 13 has a similar setup as Figure 12. The two plots show the breakdown of connection-lookahead requests by a node. *Ignored* encounters valid connection or one currently being established; *OnTime* is successful in setting up a connection before the first use; *Late* establishes a connection but not on time for the first use; *Useless* establishes a connection that is broken before being used, or is processed later than the true data transmission attempt; and *Dropped* is dropped due to two unsuccessful attempts (Section 3.1).

Recall that *Oblivious-16* can accommodate all memory-to-cache connections on non-conflicting layers simultaneously (Section 4.1). These connections may still conflict with cache-to-cache connections, but those represent a minority (less than 35%) at receiving nodes, except in *Raytrace*, *Barnes*, and *Cholesky*. As a result, memory nodes have very high connection hit rates (Figure 12, left) and almost all lookaheads are ignored (Figure 13, left). *Oblivious-8* cannot accommodate as many connections, however the number of simultaneous connections and the effectiveness of lookaheads (Figure 13, left) help the configuration overcome this handicap successfully. Indeed, L3 cache and memory latencies are large enough to hide connection setup delays via lookaheads. The contributions of these two components, however, differ across applications. For example, applications with small amount of cache-to-cache transfers, such as *FFT*, *Ocean*, *Radix*, and *Water-**, have high hit rates because a cache node can keep the connections from memory nodes on its eight layers open for a longer time. Finally, *Oblivious-4* can support relatively few connections at any one time, resulting in more conflicts and in turn reduced connection hit rates at memory nodes. Although lookaheads remain effective, in some applications the impact of the shortage of simultaneous connections is severe enough to impact performance with respect to the other two configurations.

Turning to the L2 cache-side results, we observe slightly different behavior. *Oblivious-16* simultaneously accommodates all cache-to-memory connections on non-conflicting layers without other conflicts (Section 4.1). Notice that there are no memory-to-memory data transfers. As a result, *Oblivious-16* can accommodate most of the required connections from a L2 cache at the same time, resulting in nearly perfect hit rates (Figure 12, right) and ignored lookaheads (Figure 13, right). Recall that cache-to-cache connections are typically a minority, and although they can conflict

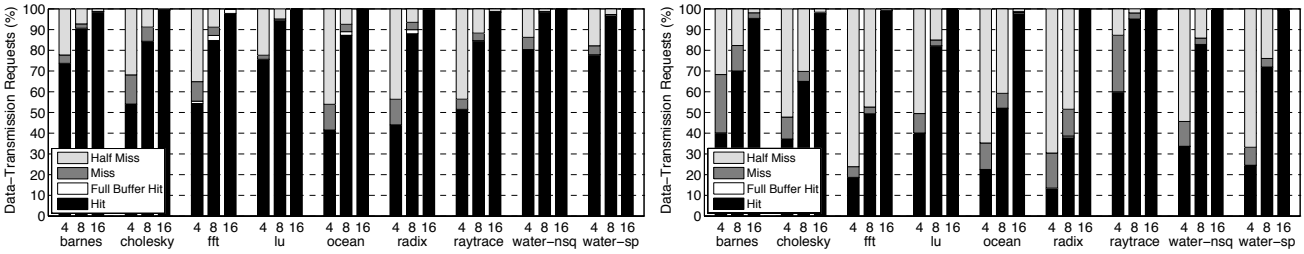

Figure 12. Average breakdown of data-transmission requests by a memory controller (left) and by a L2 cache (right). The three bars for each application show the results for Oblivious-4,-8,-16, respectively.

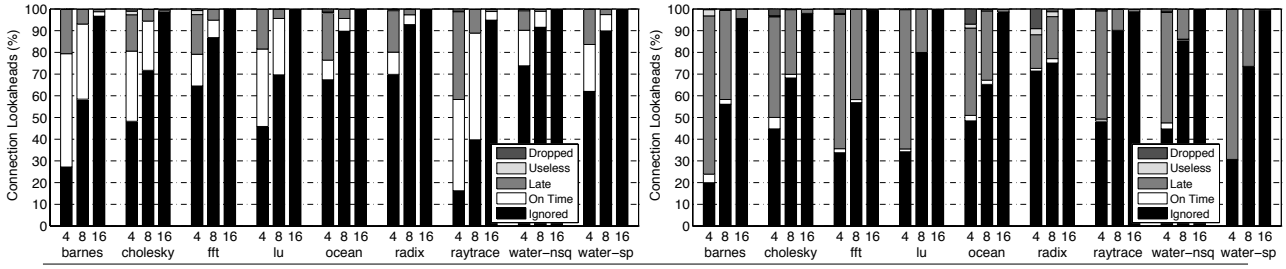

Figure 13. Average breakdown of connection-lookahead requests by a memory controller (left) and by a L2 cache (right). The three bars for each application show the results for Oblivious-4,-8,-16, respectively.

Appl.	Oblivious-4	Oblivious-8	Oblivious-16
Barnes	55.9	81.3	95.9
Cholesky	67.9	86.4	97.6
FFT	60.5	81.0	99.2
LU	72.9	91.9	99.7
Ocean	65.2	85.1	97.3
Radix	50.9	77.4	98.6
Raytrace	68.3	96.9	99.6
Water-NSq	55.0	90.1	99.4
Water-Sp	45.4	68.6	97.6

Table 9. Fraction (%) of all data supplies by sharer caches with existing connection in Oblivious-4, -8, and -16.

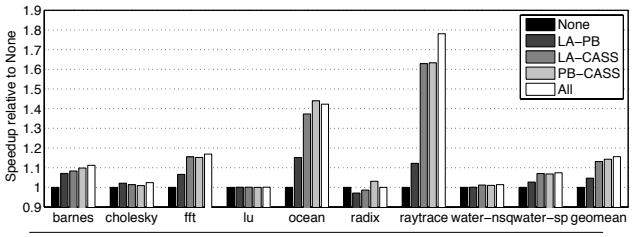

Figure 14. Study on effectiveness of connection-aware sharer-supplied data (CASS), proactive break (BP), and connection lookahead (LA) optimizations in Oblivious-8.

with memory-to-cache connections at receiver caches, they can be generally accommodated on one of the layers. Our optimizations further help increase connection utilization of cache-to-cache connections (these will be analyzed next). For cache nodes, there is a dramatic reduction in hits for *Oblivious-8* and *-4* due to increased amount of connection conflicts. Most of the missing requests are half misses because the preceding lookahead requests are late (Figure 13, right). Snoop response and L2 cache read latencies are not long enough to hide the connection setup delay on lookaheads.

Next, we provide connection-related statistics in Table 8 for *Oblivious-8* and *-16*. We show average connection setup latency, connection lifetime (in thousands of cycles), and number of times that a connection is used. The two numbers in each entry correspond to connections established by L2 cache and memory controller, respectively.

In *Oblivious-8*, for example, it takes $\sim$47 cycles on average to establish a connection. A protocol message alone takes $\sim$18 cycles on the time-slotted protocol network. These correspond to 2.6 messages per connection establishment on average, thanks to the effectiveness of proactive breaks. Without proactive breaks, typically a connection setup requires 4 messages (Section 3). Connection lifetimes and connection uses are very high in *Oblivious-16*, as expected, due to the large connection capacity it can accommodate. For *Oblivious-8*, connections are broken and set up more frequently, resulting in reduced lifetimes and uses.

Next we extract the fraction (%) of all data supplies by sharer caches with an already existing connection in Table 9. The results clearly show that the connection-aware sharer-supplied data optimization significantly increases connection utilization and improves connection hit rates. Even in *Oblivious-4*, a sharer with an existing connection can be frequently found.

Finally, we run *Oblivious-8* without any optimization, and all combinations where only two of the optimizations are included. Performance improvements in Figure 14 are relative to the configuration with no optimization. As a reference, we also provide the original results with all optimizations. The results show that combining all three optimizations is needed to extract maximum performance from the architecture, and that within the combinations attempted, those with CASS impact performance the most.

5. CONCLUSIONS

In this paper, we have proposed an all-optical approach to constructing data networks on chip that combines wavelength-based oblivious routing, passive optical routers, and connection-based operation. Our evaluation shows that a careful design based on these features yields a solution that is competitive with prior proposals from the performance standpoint, while consuming significantly

less power. The resulting mechanism can provide very high on-chip bandwidth at modest on-chip power consumption and off-chip laser power, and is reasonably robust to optical technology assumptions.

Acknowledgments

We thank Meyrem Kırman and the anonymous reviewers for useful feedback. This work was supported in part by NSF CAREER Award CCF-0545995, NSF Award CNS-0720773, and gifts from IBM, Intel, and Microsoft. Nevin Kırman was supported in part by an IBM Scholarship.

References

[1] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber, and M. Sudan. Efficient routing in optical networks. *J. of ACM*, 43(6):973–1001, November 1996.

[2] J. Ahn, M. Fiorentino, R. G. Beausoleil, N. Binkert, A. Davis, D. Fattal, N. P. Jouppi, M. McLaren, C. M. Santori, R. S. Schreiber, S. M. Spillane, D. Vantrease, and Q. Xu. Devices and architectures for photonic chip-scale integration. *J. of Applied Physics A: Materials Science & Processing*, 95(4):989–997, June 2009.

[3] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth, M. Popovic, H. Li, H. Smith, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic and K. Asanovic. Building manycore processor-to-DRAM networks with monolithic silicon photonics. In *Hot Interconnects*, pages 21–30, Stanford, CA, August 2008.

[4] A. Biberman, B. G. Lee, K. Bergman, P. Dong, and M. Lipson. Demonstration of all-optical multi-wavelength message routing for silicon photonic networks. In *Optical Fiber Communication Conf.*, pages 1–3, February 2008.

[5] J. Cardenas, C.B. Poitras, J.T. Robinson, K. Preston, L. Chen, and M. Lipson. Low loss etchless silicon photonic waveguides. *Optics Express*, 17(6):4752–4757, March 2009.

[6] L. Chen and M. Lipson. Ultra-low capacitance and high speed germanium photodetectors on silicon. *Optics Express*, 17(10):7901–7906, May 2009.

[7] L. Chen, K. Preston, S. Manipatruni, and M. Lipson. Integrated GHz silicon photonic interconnect with micrometer-scale modulators and detectors. *Optics Express*, 17(17):15248–15256, August 2009.

[8] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi. Phastlane: A rapid transit optical routing network. In *Intl. Symp. on Computer Architecture*, pages 441–450, Austin, TX, June 2009.

[9] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, San Francisco, CA, first edition, 1999.

[10] D. Ding and D. Z. Pan. OIL: A nano-photonics optical interconnect library for a new photonic networks-on-chip architecture. In *Intl. Wkshp. on System-Level Interconnect Prediction*, pages 11–18, San Francisco, CA, July 2009.

[11] R. K. Dokania and A. B. Apsel. Analysis of challenges for on-chip optical interconnects. In *Great Lakes Symp. on VLSI*, pages 275–280, Boston, MA, May 2009.

[12] A. Emami-Neyestanak, S. Palermo, H.-C. Lee, and M. Horowitz. CMOS transceiver with baud rate clock recovery for optical interconnects. In *Symp. on VLSI Circuits Digest of Technical Papers*, pages 410–413, Piscataway, NJ, June 2004.

[13] R. Ho, W. Mai, and M. A. Horowitz. The future of wires. *Proc. of the IEEE*, 89(4):490–504, April 2001.

[14] The ITRS Technology Working Groups, http://www.itrs.net. *Intl. Technology Roadmap for Semiconductors (ITRS) 2007 Edition*.

[15] N. Kırman, M. Kırman, R. K. Dokania, J. F. Martínez, A. B. Apsel, M. A. Watkins, and D. H. Albonesi. Leveraging optical technology in future bus-based chip multiprocessors. In *Intl. Symp. on Microarchitecture*, Orlando, FL, December 2006.

[16] M. Lipson. Guiding, modulating, and emitting light on silicon-challenges and opportunities. *J. of Lightwave Technology*, 23(12):4222–4238, December 2005.

[17] J. Liu, M. Beals, A. Pomerene, S. Bernardis, R. Sun, J. Cheng, L.C. Kimerling, and J. Michel. Waveguide-integrated, ultralow-energy GeSi electro-absorption modulators. *Nature Photonics*, 2:433–437, July 2008.

[18] S. Manipatruni, R. K. Dokania, B. Schmidt, N. Droz, C. Poitras, A. Apsel, and M. Lipson. Wide temperature range operation of micron-scale silicon electro-optic modulators. *Optics Letters*, 33(19):2185–2187, September 2008.

[19] S. Manipatruni, Q. Xu, and M. Lipson. PINIP based high-speed high-extinction ratio micron-size silicon electro-optic modulator. *Optics Express*, 15(20):13035–13042, October 2007.

[20] S. Manipatruni, Q. Xu, B. Schmidt, J. Shakya, and M. Lipson. High speed carrier injection 18 Gb/s silicon micro-ring electro-optic modulator. In *Proc. of the IEEE Lasers and Electro-Optics Society*, pages 537–538, Lake Buena Vista, FL, October 2007.

[21] C. Manolatou, S. G. Johnson, S. Fan, P. R. Villeneuve, H. A. Haus, and J.D. Joannopoulos. High-density integrated optics. *J. of Lightwave Technology*, 17(9):1682–1692, September 1999.

[22] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani et al. A 45nm logic technology with high-k+metal gate transistors, strained silicon, 9 Cu interconnect layers, 193nm dry patterning and 100% Pb-free packaging. In *Intl. Electron Devices Meeting*, pages 247–250, Washington, DC, December 2007.

[23] `http://sesc.sourceforge.net`, 2005.

[24] S. Palermo, A. Emami-Neyestanak, and M. Horowitz. A 90nm CMOS 16 Gb/s transceiver for optical interconnects. *IEEE J. of Solid-State Circuits*, 43(5):1235–1246, May 2008.

[25] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary. Firefly: Illuminating future network-on-chip with nanophotonics. In *Intl. Symp. on Computer Architecture*, pages 429–440, Austin, TX, June 2009.

[26] A. Shacham, K. Bergman, and L.P. Carloni. Photonic networks-on-chip for future generations of chip multiprocessors. *IEEE Trans. on Computers*, 57(9):1246–1260, September 2008.

[27] B.A. Small, B.G. Lee, K. Bergman, Q. Xu, and M. Lipson. Multiple-wavelength integrated photonic networks based on microring resonator devices. *J. of Optical Networking*, 6(2):112–120, February 2007.

[28] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. CACTI 5.3, HP Laboratories Palo Alto, `http://quid.hpl.hp.com:9081/cacti/`, 2009.

[29] 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 *Intl. Symp. on Computer Architecture*, pages 153–164, Beijing, China, June 2008.

[30] Y. A. Vlasov and S. J. McNab. Losses in single-mode silicon-on-insulator strip waveguides and bends. *Optics Express*, 12(8):1622–1631, April 2004.

[31] H.-S Wang, L.-S. Peh X. Zhu, and S. Malik. Orion: A power-performance simulator for interconnect networks. In *Intl. Symp. on Microarchitecture*, pages 294–305, Istanbul, Turkey, November 2002.

[32] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Intl. Symp. on Computer Architecture*, pages 24–36, Santa Margherita Ligure, Italy, June 1995.

[33] F. Xia, L. Sekaric, and Y. Vlasov. Ultracompact optical buffers on a silicon chip. *Nature Photonics*, 1:65–71, January 2007.

[34] S. Xiao, M. H. Khan, H. Shen, and M. Qi. Multiple-channel silicon micro-resonator based filters for WDM applications. *Optics Express*, 15(12):7489–7498, June 2007.

[35] F. Xu and A. W. Poon. Silicon cross-connect filters using microring resonator coupled multimode-interference-based waveguide crossings. *Optics Express*, 16(12):8649–8657, June 2008.

[36] Q. Xu, S. Manipatruni, B. Schmidt, J. Shakya, and M. Lipson. 12.5 Gbit/s carrier-injection-based silicon micro-ring silicon modulators. *Optics Express*, 15(2):430, January 2007.

[37] Q. Xu, B. Schmidt, J. Shakya, and M. Lipson. Cascaded silicon micro-ring modulators for WDM optical interconnection. *Optics Express*, 14(20):9430–9435, October 2006.

[38] Q. Xu, B. Schmidt, S. Pradhan, and M. Lipson. Micrometer-scale silicon electro-optic modulator. *Nature*, 435(19), May 2005.

[39] T. Yin, R. Cohen, M.M. Morse, G. Sarid, Y. Chetrit, D. Rubin, and M. J. Paniccia. 31 GHz Ge n-i-p waveguide photodetectors on silicon-on-insulator substrate. *Optics Express*, 15(21):13965–13971, October 2007.

[40] T. Yin, R. Cohen, M.M. Morse, G. Sarid, Y. Chetrit, D. Rubin, and M.J. Paniccia. 40 Gb/s Ge-on-SOI waveguide photodetectors by selective Ge growth. In *Optical Fiber Communication Conf.*, pages 1–3, February 2008.

[41] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *SPIE Optical Networks Magazine*, 1(1), January 2000.