FlexiShare: Channel Sharing for an Energy-Efficient Nanophotonic Crossbar

Yan Pan, John Kim†, Gokhan Memik

Northwestern University
2145 Sheridan Road, Evanston, IL
{panyan,g-memik}@northwestern.edu

†KAIST
Daejeon, Korea
jjk12@kaist.edu

Abstract

On-chip network is becoming critical to the scalability of future many-core architectures. Recently, nanophotonics has been proposed for on-chip networks because of its low latency and high bandwidth. However, nanophotonics has relatively high static power consumption, which can lead to inefficient architectures. In this work, we propose FlexiShare – a nanophotonic crossbar architecture that minimizes static power consumption by fully sharing a reduced number of channels across the network. To enable efficient global sharing, we decouple the allocation of the channels and the buffers, and introduce novel photonic token-stream mechanism for channel arbitration and credit distribution.

The flexibility of FlexiShare introduces additional router complexity and electrical power consumption. However, with the reduced number of optical channels, the overall power consumption is reduced without loss in performance. Our evaluation shows that the proposed token-stream arbitration applied to a conventional crossbar design improves network throughput by 5.5× under permutation traffic. In addition, FlexiShare achieves similar performance as a token-stream arbitrated conventional crossbar using only half the amount of channels under balanced, distributed traffic. With the extracted trace traffic from MineBench and SPLASH-2, FlexiShare can further reduce the amount of channels by up to 87.5%, while still providing better performance – resulting in up to 72% reduction in power consumption compared to the best alternative.

1. Introduction

With the prospect of many core processors [12, 6, 22] on the horizon, the energy efficiency of on-chip networks is becoming increasingly important. As on-chip network size continues to increase, high bandwidth and low latency are required to achieve high performance. Hence, architects have recently explored nanophotonics [23, 14, 13, 18] as an alternative to conventional electrical signaling.

However, nanophotonics pose many new challenges [11]. For example, unlike conventional electrical signaling, static power consumption constitutes a major portion of the total power [5]. The laser that carries the signal faces various losses along the path and the total loss can be significant; and the energy conversion efficiency of laser sources is around 30% [5], further aggravating the total power consumption. In addition, the ring resonators used in nanophotonics require thermal tuning, incurring significant heating power. Such static power consumption makes nanophotonic channels an expensive on-chip resource. Thus, future nanophotonic on-chip networks need to fully utilize their provisioned channels and avoid over-provisioning channel bandwidth.

Since nanophotonics is appropriate for global signaling in on-chip networks without requiring repeaters, they are suitable for implementing global crossbars [14, 23, 18]. However, in conventional nanophotonic crossbar designs, channels are dedicated to each node in the network – thus, as network size increases, the number of channels required increase proportionally. When global bandwidth is not heavily utilized, such bandwidth provisioning can be excessive. In Section 2, we will present Network-on-Chip (NoC) traces showing that some nodes are inactive for extended periods of time during the execution.

To address these problems, we propose the FlexiShare nanophotonic crossbar architecture, where global channels are detached from the routers and shared globally among all the routers in the network – implementing *global* concentration. To support efficient channel utilization with limited number of channels, we propose photonic token-stream arbitration and provide multiple tokens to increase network throughput. The token-stream concept is also extended to buffer management: token streams are used to manage the shared buffers and hence the allocation of the channels and buffers are decoupled.

FlexiShare introduces additional complexity to support the flexibility. However, through evaluation, we show how the number of channels and their associated static power consumption can be significantly reduced while maintaining similar performance as conventional designs – thus achieveing higher overall power efficiency. We compare FlexiShare against alternative architectures using both evenly distributed synthetic traffic patterns and traces from SPLASH-2 [25] and MineBench [17] benchmarks.

In summary, the contributions of this paper include:

- A flexible crossbar topology that allows channel provisioning according to average traffic load and create *global concentration* that exploits unbalanced bandwidth requirements.
- A distributed token stream arbitration which provides multiple tokens for a given channel and enables high channel utilization.
- Two-pass token-stream arbitration which provides lower bound on network fairness.
- A global buffer manangement scheme that decouples buffer resources from channels.

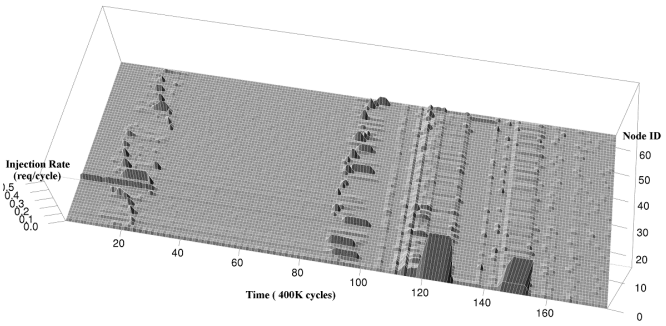

Figure 1. Network request rate for 64-core CMP running *radix* (SPLASH-2).

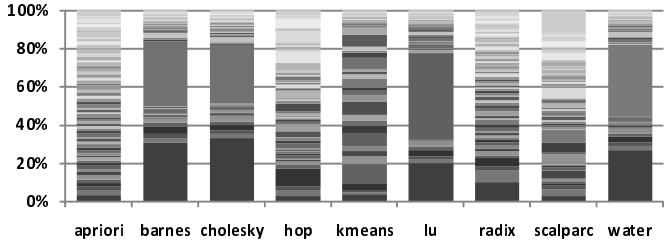

Figure 2. Load distribution across 64 nodes for the selected benchmarks. Different shades indicate different nodes.

The remainder of this paper is organized as follows. In Section 2, we present experiment results motivating the design of FlexiShare. Section 3 describes the details of the FlexiShare architecture and the token stream arbitration schemes for channels and buffers. We evaluate the performance and power efficiency of FlexiShare and compare it against alternative crossbar designs in Section 4. After reviewing the related work in Section 5, we conclude the paper in Section 6.

2. Motivation

In FlexiShare, we advocate reducing the number of of nanophotonic channels and share them globally among all the nodes. This is based on two observations: (a) unbalanced traffic load in NoC architectures and (b) dominant static power consumption in nanophotonic channels.

2.1. Unbalanced Traffic

Although different synthetic traffic patterns are often used in evaluation of on-chip networks, they typically assume equal traffic is injected from each node in the network, which is not necessarily representative of network traffic in CMPs. Figure 1 shows the average network request rate of each core when running *radix* (SPLASH-2 kernel) in a 64-core CMP under Simics/GEMS simulation environment assuming a point-to-point network topology. The horizontal axis is time, segmented in 400K-cycle frames. The traffic load is shown for all the 64 nodes in parallel. It can be seen that while some hot nodes (like node 0 and 1) have high loads, a large number of nodes have low bandwidth requirement, creating opportunity for bandwidth sharing among the nodes.

We also analyze the total network requests from each nodes in a 64-node CMP running 9 benchmarks from SPLASH-2 and MineBench, as shown in Figure 2. For some benchmarks, there is a small set of nodes that generate a large portion of the total traffic. Thus, if we could share a smaller number of channels across all the nodes, instead of dedicating a set of channels to each node, the channel utilization and the overall efficiency of the network can be improved.

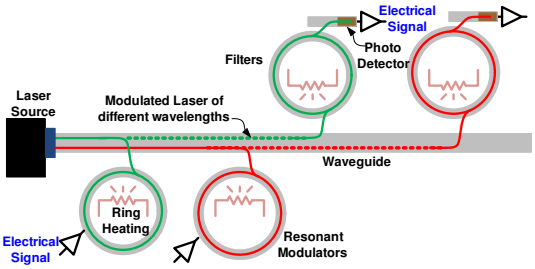

Figure 3. Nanophotonic Devices

2.2. Power Consumption of Nanophotonics

Silicon nanophotonics is an emerging technology that promises low latency and high bandwidth density. The basic components of nanophotonic signaling are shown in Figure 3. An off-chip laser source generates laser beam with multiple wavelengths, which is coupled to the on-chip waveguide. The resonant ring modulators modulates a specific wavelength with the electrical signal. At the receiver end, ring resonators filters out that specific wavelength and photodetectors convert the optical signal back into electrical signal. A common building block here is ring resonator, which needs to be thermally tuned to align its resonant wavelength.

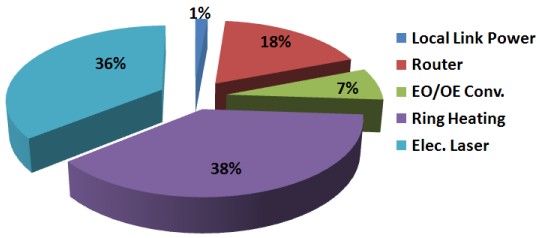

Figure 4. Energy breakdown in a conventional radix-32 nanophotonic crossbar

For conventional electrical network designs, the buffers and switches dominate the total power consumption [24] and with the abundant on-chip wire resources available, there is little motivation to provide sharing of the channels. In addition, much of the power consumption is dynamic power, which is proportional to the activity of the channels. However, for nanophotonic on-chip networks, the power consumption scenario is different. The laser power and the ring tuning power, which are activity-independent, dominate the total power consumption. For example, prior work Corona [23] estimated a static power of 26W for the optical crossbar, regardless of the traffic in the application. The nanophotonic Clos network [13] adopts point-to-point nanophotonic links and based on their power model, a 33W laser power budget is assumed, while the

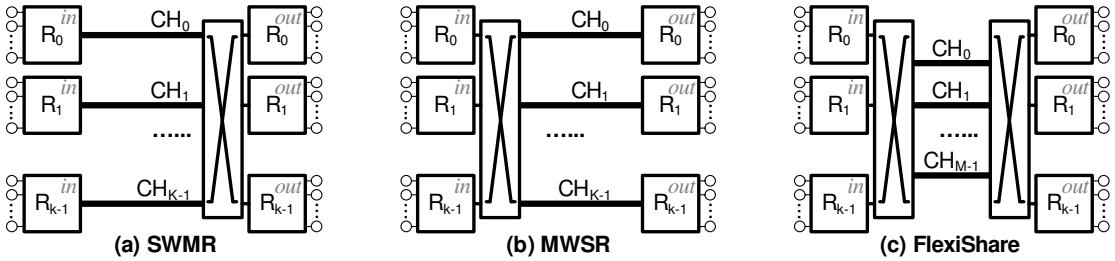

Figure 5. Radix-k nanophotonic crossbar implementations

routers and electrical local links together consume less than 8W for a high bandwidth design. Based on the power model in Section 4.7, we analyzed the energy consumption breakdown in a conventional radix-32 single-write-multiple read crossbar [18], as shown in Figure 4. It shows how static power dominates the overall power consumption. With such high static power cost in using nanophotonic links, it is important to improve the utilization of the channels and try to reduce the number of channels without losing performance. Through flexible channel sharing, we show how this is achieved in FlexiShare in the following sections.

3. FlexiShare Architecture

In this section, we describe the FlexiShare nanophotonic crossbar architecture which includes a novel token-stream based arbitration scheme, a distributed credit-based flow control scheme, and the router microarchitecture description.

3.1. Overview : Shared Channel Design

High-level architecture diagrams of two conventional nanophotonic crossbar implementations are shown in Figure 5, in comparison with FlexiShare. Concentration is assumed as each router is connected to multiple nodes. To illustrate the logical organization of the nanophotonic data channels (CH_i), we separate each router (R_i) into an input router (R_i^{in}) or the sending router and an output router (R_i^{out}) or the receiving router. However, they would be physically implemented as a single router.

In the single-write-multiple-read (SWMR) design, each router has a dedicated sending channel, and all the routers read on the other routers' channels to implement the logical crossbar at the receivers' end. This only requires local arbitration at the receiver side and has been proposed by Kirman et al. [14] and in Firefly [18]. Multiple-write-single-read (MWSR) crossbars, on the other hand, provision each router with a dedicated receiving channel, and all the routers transmit data on the receivers' channels, forming a crossbar at the senders' end. This design requires global arbitration [23, 18] to resolve channel contention. For both designs of conventional nanophotonic crossbar, the number of channels required in the design is proportional to the radix (k) of the crossbar to provide full connectivity. If the network traffic is unbalanced (e.g., only a few nodes are actively exchanging data) or if the network bandwidth usage is low, not all of these channels will be fully utilized.

In FlexiShare, we propose to detach the channel resources from the routers and provision the channel bandwidth independently, as shown in Figure 5(c). In this design, on the sender side, the architecture is similar to MWSR, where each sender can transmit its packets on any available channel, forming a logical crossbar. On the receiver side, it is similar to SWMR, where all the receivers are equipped to read from all of the channels, forming a second logical crossbar.

To support the two back-to-back crossbars, FlexiShare requires additional router complexity and approximately twice the amount of optical hardware (ring resonators) as SWMR or MWSR *if* the number of channels in the crossbars is held constant. However, the added flexibility allows FlexiShare to be provisioned with any number of channels (M), independent of the crossbar radix (k). Such flexibility improves utilization of the nanophotonic data channels and hence results in higher power efficiency, especially under unbalanced traffic load.

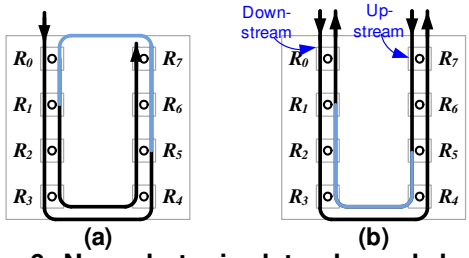

Figure 6. Nanophotonic data channel designs: (a) two-round and (b) singl-round data channels

3.2. Nanophotonic Data Channel Design

There are two ways to implement a nanophotonic data channel [13]. The waveguide layout of both are shown in Figure 6, with the path of data from R_5 to R_1 highlighted. In Figure 6(a), a *two-round data channel* employs a single set of wavelengths to traverse each node twice – allowing senders to modulate in the first round and the receivers to detect and demodulate in the second round. An alternative is the *single-round data channel*, as shown in Figure 6(b), which uses two sets of wavelengths in opposite directions. Here, we call the direction of increasign router number as "downstream", and the opposite direction as "upstream". Even though single-round data channel requires extra wavelengths (and waveguides), it reduces the waveguide length and minimizes optical loss, and hence is adopted in FlexiShare. In the following sections, we call the data channel in one direction as a *sub-channel* and our discussion only focuses

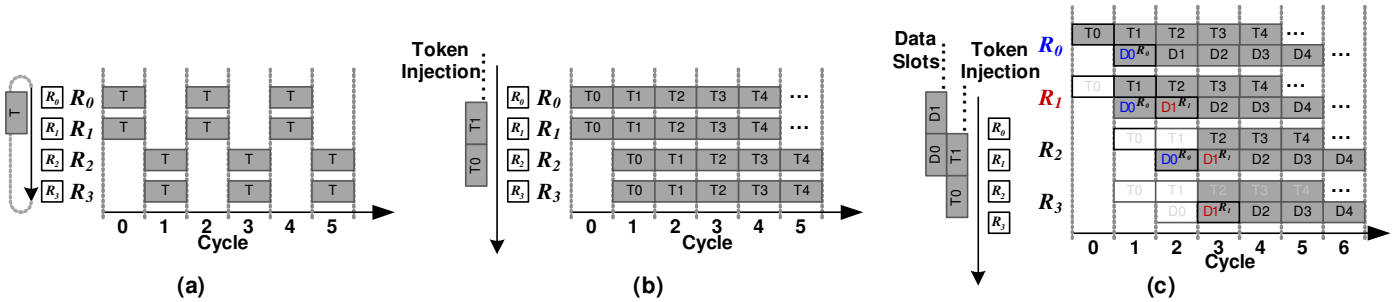

Figure 7. Token based arbitration schemes on a 4-node network. (a) Token-ring scheme; (b) Token-stream scheme; (c) Token-stream scheme applied to data channel arbitration, with requests from R_0 and R_1 in Cycle 0, R_2 in Cycle 1 and R_1 in Cycle 2.

on the downstream sub-channel if not otherwise indicated.

3.3. Token Stream Channel Arbitration

One challenge faced by FlexiShare crossbar is to make sure the data of different senders are not overlapped on a shared channel – i.e., to resolve contention for the shared channel resources properly while maintaining high channel utilization. Such arbitration is needed in the first logical crossbar which is identical to the global arbitration needed in an MWSR crossbar. Prior work [18, 23] on MWSR crossbars have used token-based global arbitration, which leverages the low latency of nanophotonics.[1] A single photonic token, which represents the right to transmit data on a channel, circulates around the network and passes all nodes. An example is shown on the left of Figure 7(a), where a token circulates around 4 routers in the clockwise direction. If a node wants to transmit data on a data channel, it has to first grab the corresponding token by coupling the energy of the token off the waveguide.

However, with such token-ring arbitration, the round-trip latency of the token can limit the throughput of the network and result in poor channel utilization. For the previous example of Figure 7(a), if we assume a token round-trip latency of 2 cycles, the resulting timing diagram of the token circulation is shown on the right in Figure 7 (a). Each node can only grab the token every other cycle – thus, resulting in a maximum of 50% throughput on traffic patterns such as permutation traffic. In general, if the token round-trip latency is r cycles, the network throughput can be limited to $1/r$ on adversarial traffic patterns.

3.3.1 Single-Pass Token-Stream Arbitration

In FlexiShare, we overcome this throughput bottleneck by introducing a novel *token-stream* arbitration scheme, as shown in Figure 7(b). Similar to token-ring arbitration, photonic tokens are used to arbitrate the channels. However, *multiple* tokens are continuously injected to create a "stream" of tokens that traverses all the nodes, in parallel to its corresponding data channel. The low latency of nanophotonics allows a token to traverse multiple nodes in a single cycle (two nodes in the example in Figure 7(b)), allowing, e.g., either R_0 or R_1 to grab $T0$ in cycle 0. Each cycle, a new token is injected at the origin of the stream, and a token, if not grabbed by any router, will be eliminated at the end of the stream.

As opposed to the token-ring arbitration, a token in the token stream represents the right to modulate on the corresponding data channel in the *next data slot*. For example, in Figure 7(c), the timing of the tokens $(T0, T1, \cdots)$ are shown together with their respective data slots $(D0, D1, \cdots)$. The superscript of a data slot indicates the router that has modulated on that slot. Tokens that are already grabbed by an upstream router are greyed out and filled with white color, to illustrate that other routers cannot grab these tokens. In this example, R_0 and R_1 both request a token in cycle 0, but only R_0 will be able to couple the energy of token $T0$ off the waveguide as it is upstream to R_1. R_0 will occupy data slot $D0$ in the next cycle (cycle 1), and the modulated data in $D0$ will arrive at its destination R_2 in cycle 2 to be detected and demodulated. R_1 does not get a valid $T0$ and will retry in cycle 1 for $T1$, which will subsequently allow it to occupy $D1$ and transmit to R_3. As there is a one-to-one correspondence between the tokens and the data slots, data channel conflicts are resolved (e.g., R_1 and R_2 will never overwrite R_0's data in slot $D0$), and a daisy-chain-like priority scheme is implemented.

This token stream arbitration scheme leverages the passive writing nature of nanophotonics, where the modulation is done by imprinting message onto a laser beam rather than driving a whole channel. Thus, the key for arbitration is not to avoid two writers on a same channel at any moment, but rather to avoid the overwriting on the same *slot* by two senders. By arbitrating on a fine-grain data slot basis, the channel utilization can be significantly improved, compared to the token-ring scheme.

However, the token-stream arbitration also has its limitations. For example, it is not possible to hold on to a channel and send multiple flits in sequence, as compared to token-ring arbitration where a node can delay the re-injection of the token to occupy the channel for more than 1 cycle. However, given the high bandwidth density of nanophotonics, the channels in an on-chip nanophotonic crossbar are often wide-enough such that a large packet (e.g., a cache line) can fit in a single flit – thus, interleaving flits in the FlexiShare architecture is not a serious problem. Another drawback of this single-pass token stream arbitration is its daisy-chain-like fixed priority – upstream routers

[1] We will refer to this arbitration as *token-ring* arbitration since the token traverses the waveguide in a circular, ring structure.

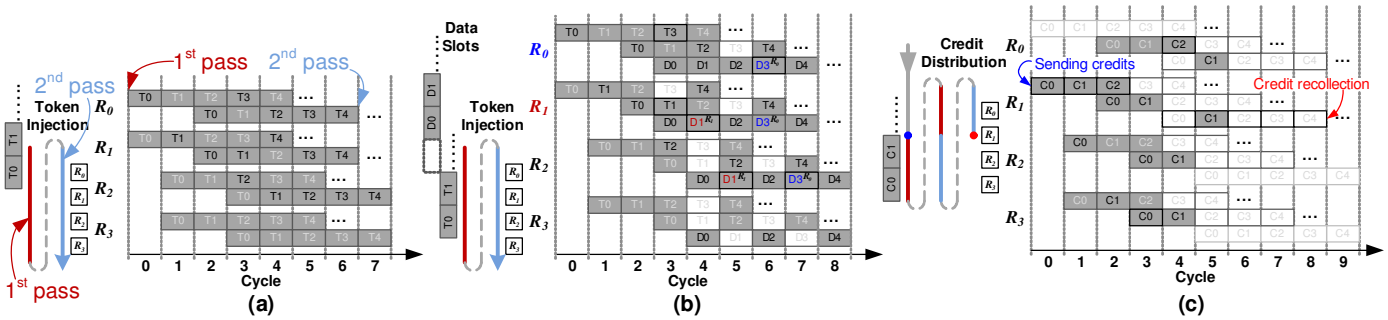

Figure 8. (a) Two-pass token stream arbitration scheme; (b) Two-pass token stream arbitration scheme with requests from R_0 and R_1 in Cycle 3; (c) Two-pass credit distribution scheme where R_1 is distributing credits to the other routers.

always have advantage over downstream routers. In the example shown in Figure 7(c), if R_0 continuously have requests for a data channel, it will always obtain the token and occupy the data slot while downstream routers will be starved. To overcome this limitation, we propose a two-pass token stream arbitration in the next subsection.

3.3.2 Two-Pass Token Stream Arbitration

In two-pass token stream arbitration, the token stream runs past each router twice[2], as shown by the two solid lines on the left of Figure 8(a). In the first pass, each token is dedicated to a different router. During the second pass, all the routers are free to grab any token if the router does not have a dedicated token on the first pass for that cycle.

For example, in Figure 8(a), only routers R_0, R_1 and R_2 transmit on the downstream data sub-channel. In the first pass $T0$ is dedicated to R_0, $T1$ to R_1, $T2$ to R_2, and $T3$ back to R_0 again. R_3 does not need any token for this downstream sub-channel as all the traffic it sends will be on the upstream channel. In the timing diagram, tokens that cannot be grabbed by a router because of lack of priority are greyed out. In general, for a radix-k crossbar, token $T((k-1)\,i+j), (i=0,1,\cdots)$ is dedicated to router R_j in the first pass. Such dedication is implemented by a state-machine at each node – thus each token stream is still 1-bit wide. When a token comes back to R_0 in the second pass, any router can grab it, just like in a single-pass token stream. An example is shown in Figure 8(b), where data slots $D0, D1\cdots$ are shown together with the two pass token stream $T0, T1, \cdots$. In cycle 2, R_1 can request for $T0$ in the second pass even though T_0 was dedicated to R_0 in the first pass. However, R_2 is not allowed to grab T_0 in cycle 3, because in that cycle, R_2 has a dedicated token $T2$ in the first pass and it has to use the dedicated token for its data transfer. Upon grabbing a token, the router is granted the right to modulate on the associated data slot, which comes after the token passes the node the second time. Note that the data channels still only run a single round across each node.

The two-pass token stream arbitration scheme puts a lower bound on the fairness of the sharing of a single data channel as each node are guaranteed (in the first pass) its own share of the channel while unused dedicated data slots can be recycled in the second pass – combining the benefits of both dedicated time slot scheme and daisy-chain priority scheme.

[2]The wrap-around of the stream (dashed lines in Figure 8) can be easily implemented by routing the waveguide as shown in Figure 12.

3.4. Receiver-side Arbitration

In FlexiShare, once a channel is granted by the token stream arbitration, the router will essentially act like a SWMR router to send the data packet to its destination through the temporarily dedicated channel for the sender. We adopt the reservation-assisted scheme in Firefly[18], where the sender will activate its receiver through reservation channels before sending over the data on the data sub-channel. The overhead of latency and power of the reservation channels are fully modeled in our evaluation.

3.5. Credit Stream Flow Control

Buffered flow control is employed in FlexiShare and proper buffer management is required to ensure packets are not dropped. Conventional credit-based flow control maintains a credit count at the sender side for the available downstream buffers. Thus, each credit (and the corresponding buffer space) is implicitly associated with a channel – i.e., obtaining a credit also implies a particular channel to be used for transmission. However, in FlexiShare, because of the global sharing of the channels, channels are detached from the routers and hence the downstream buffers. The conventional credit-based flow control could still be used but would be expensive because of the $O(k^2)$ broadcasting of the credits as well as properly maintaining the credits across all the routers. Hence, we treat the input buffers as a resource shared globally among all the sending routers, just like the channels, and we extend the token-stream mechanism to manage the credits.

The scheme is shown in Figure 8(c). Unlike conventional credit-based flow control where sending routers maintain credits for receiving buffers, each router keeps its own credit count for the number of available input buffers. As long as there is enough buffer, the router (e.g., R_1 in the figure) will send out optical credit tokens ($C0$, $C1$,...) in a *credit stream* that passes all the other routers twice. Similar to the two-pass token stream described previously, the two passes are needed to provide fairness – credits in the first pass are dedicated to the different

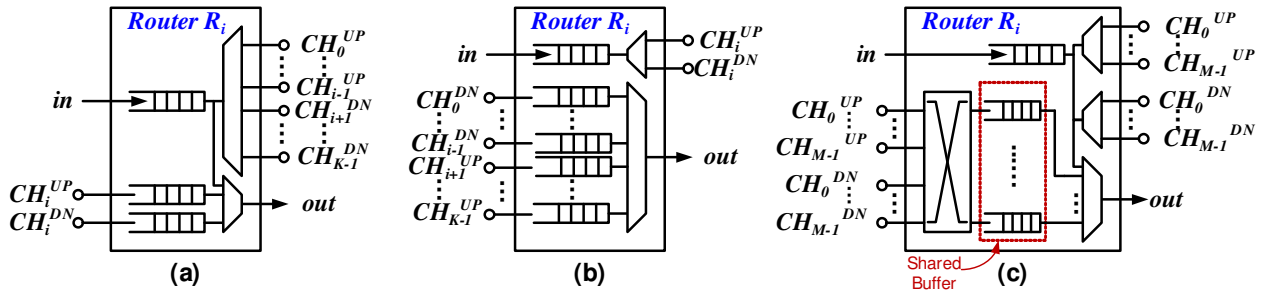

Figure 9. Microarchitecture of router R_i in a radix-k nanophotonic crossbar with concentration $C = 1$ using (a) MWSR, (b) SWMR, and (c) FlexiShare with M channels

routers, and any remaining credits flowing into the second pass can be used by any router.

Different from the token stream for channels, the credit streams have to originate from each router. This requires the laser to traverse slightly longer distance to cover all the node twice after it first passes the credit distributor. When the the tokens come back to its distributor (R_1) after the two passes, they will be re-collected by the distributor to increment its credit count since the credit (or the buffer space) was not utilized. Another difference from token stream is that a single stream is enough for each router and it can run in either upstream or downstream direction, whichever yields the shorter waveguide length.

For example, in Figure 8(c), the two passes are indicated by different colors of the solid lines on the left. R_1 injects credits $C0$, $C1$ and $C2$ but stops afterwards because no more buffer is available. $C0$ is dedicated to R_2 but is grabbed in the second pass by R_3, in cycle 3; while R_0 grabs its dedicated $C2$ in its first pass in cycle 4. $C1$ is not grabbed by any router, and is recollected by R_1, in cycle 5.

3.6. Router Microarchitecture

With the token stream channel arbitration and credit stream flow control, each packet in FlexiShare experiences several pipeline stages. Upon entering the sending router, it generates a credit request for its destination router's input buffer. When the credit is acquired, the packet generates a channel request. Based on the relative location of the sender and receiver, this could be a request for either the upstream or downstream data channel. Any channel in the proper direction is equally feasible. Upon grabbing a token for the data channel, the packet will be distributed to the modulators of the data channel, a reservation will be sent on the corresponding reservation channel to activate the reciever detectors, and the packet will be modulated onto the assigned data slot.

On the receiver end, the packets will be stored in a *shared buffer* before being sent to the ejection ports of the output router which are connected to the network terminal nodes. With a shared buffer, a single credit count is used to represent the availability of the buffer, as described earlier in Section 3.5. The shared buffer is implemented by using a switch organization similar to a Load Balanced Birkoff-von Neumann Switch [7] (Figure 9(c)), where the first switch load-balances the $2(M-1)$ incoming channels onto $2(M-1)$ intermediate queues (the shared buffer) and a second switch[3] connects these queues to the ejection ports. The load-balancing across the intermediate queues enable the use of a single credit count as the queue lengths are balanced. Switch speedup [9] in the first switch here can be implemented to reduce the number of intermediate queues as needed but we do not assume any switch speedup.

The comparison of alternative router microarchitectures is shown in Figure 9, assuming single-round data channels with an upstream and a downstream sub-channel. For simplicity, a concentration of $C = 1$ is shown. Note that in MWSR implementation (Figure 9(a)), the direction of the data channel (either upstream or downstream) is decided by the relative location of the sender and receiver, and the receiver has to buffer the packets from both directions of the data channel. The opposite situation applies to SWMR, where the direction of the receiving sub-channels are decided by the relative location. Because of this restriction of channel usage, each MWSR or SWMR router has access (read or write) to only half the number of data channels. However, for FlexiShare, full access to all of the data channels are allowed, as shown in Figure 9(c). This corresponds to the fact that FlexiShare can match the performance of MWSR or SWMR with half the amount of channels, as discussed in the evaluation section.

3.7. Realistic Token-Stream Latency

For simplicity, the timing diagrams shown in previous sections ignored various latencies for signal conversion, token traversal, switch traversal, arbitration logic, etc. In reality, a more detailed timing diagram for a single-pass token stream would be as shown in Figure 10. In this example, there is a skew for each token to arrive at each router, based on the router location. R_0 indicates its request for a token at the beginning of cycle 0, and will only get the grant at the beginning of cycle 2 due to latency in signal conversion. And it takes another cycle for R_0 to send the data packet to the appropriate modulators to send in cycle 3 on data slot $D0$. However, these skews of latencies are constant for each router and does not impact the arbitration mechanism. The evaluation in the next section accounts for all these realistic latencies.

[3] In Figure 9(c), this switch is reduced to a single $(M-1)$-to-1 multiplexer because concentration $C = 1$ is assumed.

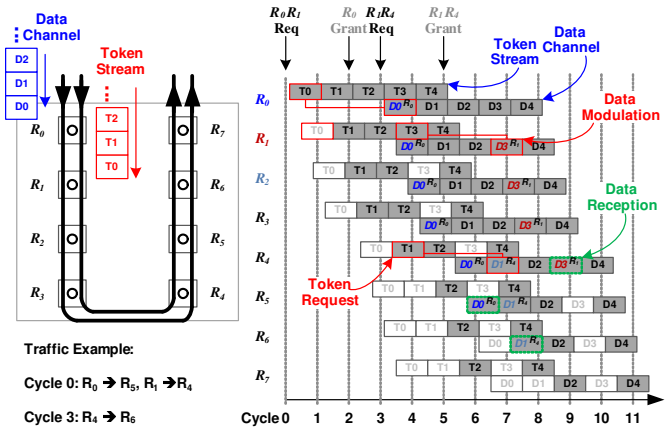

Figure 10. Sample single-pass token stream timing diagram with realistic latencies.

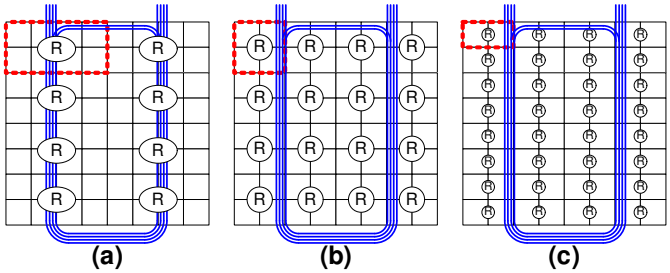

Figure 11. Waveguide layout for varied crossbar radix (k) and concentration (C): (a) $k = 8, C = 8$, (b) $k = 16, C = 4$ and (c) $k = 32, C = 2$.

3.8. Waveguide Layout and Integration

We assume 3-D integration to support FlexiShare such that a separate optical die is stacked on top of a processor die. Such stacking enables specialized fabrication process for the optical die and more freedom for the waveguide routing. 3-D stacking also makes power a more important constraint compared to area for the on-chip network [13]. The same layout of the waveguide for a 64-tile processor is assumed for SWMR, MWSR and FlexiShare, as shown in Figure 11. Each tile has a single interface to the on-chip network and various degrees of concentration can be implemented.

We assume Dense Wavelength Division Multiplexing (DWDM) where up to 64 wavelengths are transmitted in a single waveguide (in both directions). The waveguides run in parallel to avoid crossing. Figure 12(a,b) show the waveguide layout for the token streams (Section 3.3.1) and credit streams (Section 3.5). The token stream waveguide passes each router twice to enable the two-pass token-stream arbitration (Section 3.3.2); while for the credit streams, the laser has to be routed to the router distributing the credits first (R_2 in the figure), and then traverse all the routers twice, as highlighted with different colors.

The different types of channels required in a radix-k FlexiShare network with M channels is shown in Table 1. Because of the need for upstream and downstream sub-channels, two

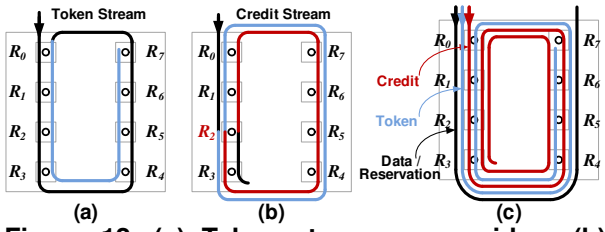

Figure 12. (a) Token stream waveguide. (b) Credit stream waveguide. (c) Waveguides for all the 3 types of channels.

Channel	# of λ	Waveguide	Comments
Data	$2M \times w$	1-round, bi-dir	w-bit datapath
Reservation	$2k \log k$	1-round, bi-dir	broadcast
Token	$2k$	2-round, bi-dir	
Credit	k	2.5-round, uni-dir	

Table 1. Channels in FlexiShare

sets of wavelengths are needed for data, reservation and token channels in opposite directions. Reservation channels need higher laser energy to broadcast the destination information for a packet. Figure 12(c) demonstrates how to layout the waveguides of all these channels without crossing each other. Note that only downstream data/reservation and token waveguides are shown. The upstream counterparts are simply mirrored by y-axis. Here, credit streams of $R_0 \cdots R_3$ are downstream and are shown, while $R_4 \cdots R_7$ credit streams are not shown as they are upstream.

4. Evaluation

In this section, we first describe our system setup for evaluation. Then, the performance of FlexiShare with varied channel provision is analyzed, followed by comparison with alternative designs. Based on the nanophotonic power model in [13], we also estimate the potential of power reduction in FlexiShare.

4.1. System Setup

A cycle accurate network simulator is developed based on the booksim simulator [9, 3] and modified to represent the various topologies. The evaluated alternative topologies with their features are listed in Table 2. We generalize conventional nanophotonic crossbar designs into two categories: MWSR (e.g., Corona [23]) and SWMR (e.g., the optical bus proposed by Kirman et al. [14]). Note TR-MWSR adopts the two-round data channel design (Figure 6(a)). The proposed token-stream arbitration scheme is applied to an MWSR architecture (TS-MWSR) to demostrate the benefit of this arbitration scheme itself. We assume a refractive index of $n = 3.5$ for the waveguide and a conservative 2-cycle latency each for processing an optical token request. The clock frequency is targeted at 5GHz. We focus on network size of 64 ($N = 64$) and each data packet contains a single flit of 512-bits.

4.2. FlexiShare Channel Provision

Figure 13 shows the load latency curves for a radix-8, 64-node FlexiShare with varied amount of channels (M). Note

Code Name	Channel Arbitration	Credit Control	Data Channel Type	Comments
TR-MWSR	Token Ring	Infinite Credit	Two-round (Figure 6 (a))	-
TS-MWSR	2-pass Token Stream	Infinite Credit	Single-round (Figure 6 (b))	-
R-SWMR	-	2-pass Credit Stream	Single-round (Figure 6 (b))	Reservation-assisted
FlexiShare	2-pass Token Stream	2-pass Credit Stream	Single-round (Figure 6 (b))	Reservation-assisted

Table 2. Evaluated networks

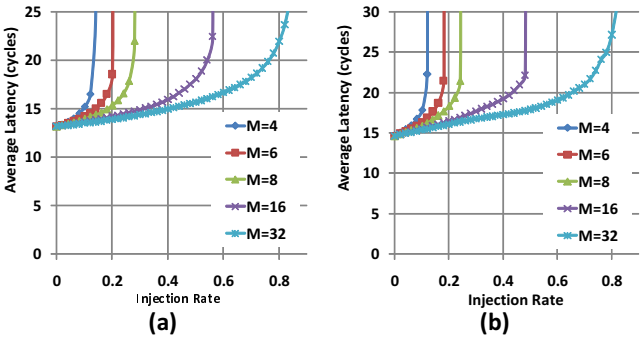

Figure 13. FlexiShare ($C = 8, N = 64, k = 8$) with varied M under (a) uniform random and (b) bitcomp traffic

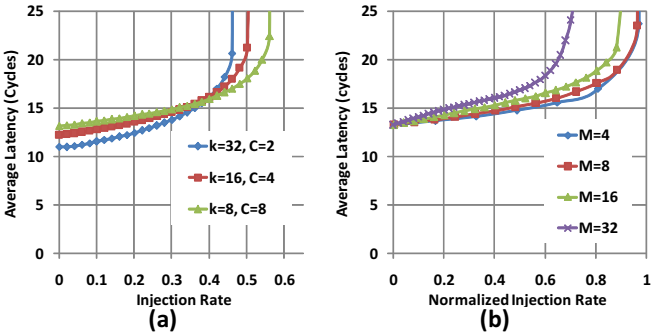

Figure 14. (a) Flexishare ($M = 16, N = 64$) with varied k and C under uniform random traffic. (b) Channel utilization of FlexiShare ($M = 16, N = 64$) under bitcomp traffic

each data channel has two sub-channels running in opposite directions. By varying the number of channels (M) provisioned, the network throughput can be tuned almost linearly – increasing M increases the throughput as the amount of network bandwidth increases with M. This flexibility is important as the amount of provisioned channels directly corresponds to the designed optical power budget. FlexiShare with the 2-pass token-stream arbitration is also insensitive to traffic patterns, showing minimal performance loss with permutation traffic such as bitcomp (Figure 13(b)).

4.3. Channel Utilization

Figure 14(a) shows the performance of FlexiShare with the same number of channels ($M = 16$) for a $N = 64$ network, but with different crossbar radix and concentration. It can be seen that higher throughput is achieved by lower radix. The reason is that in the token stream arbitration scheme, each cycle a

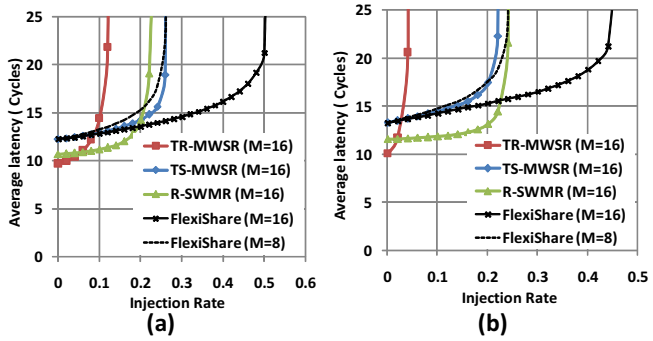

Figure 15. TR-MWSR, TS-MWSR, R-SWMR and FlexiShare ($k = 16, N = 64$) under (a) uniform random and (b) bitcomp traffic.

router speculatively sends a request for *one* of the channels for each packet to be sent. If the request is not granted, it will retry each channel in a round-robin fashion in the following cycles. Thus, if there are a large number of routers requesting on the same stream, the success rate for downstream routers diminishes, which corresponds to degraded throughput. Figure 14(a) shows that a throughput reduction of 18% for a radix-32 FlexiShare compared to a radix-8 FlexiShare.

Channel utilization directly corresponds to the efficiency of the network. In Figure 14(b), we normalize the injection rate by the amount of nanophotonic channels provisioned (x-axis). Thus a normalized injection rate of 1.0 means ideal utilization of the nanophotonic channels. It can be seen that the throughput is around 0.95 if the amount of channels is far lower than the number of tiles (e.g., 8 channels for a 64-tile network). However, as the number of channels increases, this efficiency tends to decrease. This is because, if there are not enough channels, there will be multiple routers trying to grab the tokens of each channel, and thus it is less likely for a token to go un-grabbed – resulting in high channel utilization. When there are more channels, there will be fewer routers trying to grab each token, and if a router makes a wrong speculation on which channel to try, it might be blocked while letting an available token on a different channel go unused – resulting in lower channel utilization. However, even for the full provision of FlexiShare with 32 channels for 64 tiles, the throughput is still above 0.7.

4.4. Comparison with Alternatives

We compare the performance of FlexiShare to the alternative nanophotonic crossbar implementations in Figure 15. The token-ring arbitrated TR-MWSR shows the worst throughput, as the channel is not fully utilized because of the token round-

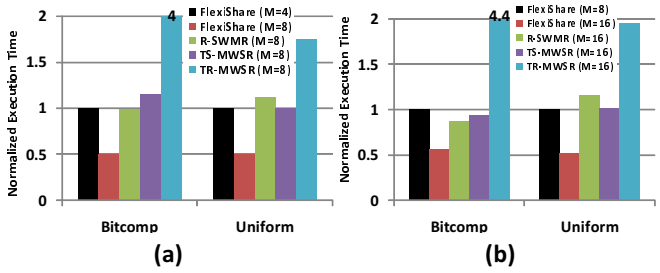

Figure 16. Normalized execution time (with respect to FlexiShare). (a) $k = 8$ **(b)** $k = 16$

trip latency. By applying our proposed token-stream arbitration to a MWSR network (TS-MWSR), the throughput is improved by 5.5 times for bitcomp traffic. R-SWMR and TS-MWSR show similar throughput, while the differences can be attributed to conflict scenarios at the sender and receiver ends.

A striking difference here is that with the same amount of channels ($M = 16$), FlexiShare is able to provide almost *twice* the throughput as TS-MWSR or R-SWMR. This is because FlexiShare fully accesses both the upstream and downstream data channels, while the dedicated channel TS-MWSR or R-SWMR can only utilize half of the data channels, as shown in Figure 9. For example, for a radix-8 network under bitcomp traffic, TS-MWSR only utilizes the downstream sub-channel to send to node 4,5,6,7 and the upstream sub-channels to send to 0,1,2,3, leaving the upstream sub-channels to 4,5,6,7 and the downstream sub-channels to 0,1,2,3 idle. Note that these idle data channels are necessary to guarantee full connectivity of the network. Thus, FlexiShare is able to provide similar performance as TS-MWSR or R-SWMR with half the amount of channels, as shown by the dashed lines (FlexiShare ($M = 8$)) in Figure 15. However, with the token-stream arbitration, the *delayed* data slot increases the zero-load latency of the packets and results in approximately 30% increase in zero-load latency, compared to token-ring arbitration.

4.5. Synthetic Workload

In addition to the load latency curves, we composed a synthetic workload where each tile has a fixed amount of network requests (100K), whose destination observe a certain traffic pattern (bitcomp or uniform). Upon receiving a request, the tile generates a reply packet to the source – in an attempt to mimic cache coherence messages. Each tile is allowed to have at most 4 outstanding requests, at which point it will be blocked from sending more request. The results are shown in Figure 16. Similar to previous findings, token stream arbitration reduces the total execution time on MWSR crossbars by at least a factor of 3.5 on bitcomp traffic as compared to the token ring arbitrated TR-MWSR. In addition, FlexiShare with half the amount of channels achieves similar performance as TS-MWSR or R-SWMR. It is no surprise to find again that lower radix FlexiShare achieves higher performance.

4.6. Trace-based Workload

In the synthetic traffic patterns, we have shown performance results of the various networks under *balanced* traffic loads – that is, each node has similar injection rate. However, in reality, the traffic load distribution on a CMP may not be always balanced. There will be busy nodes and largely idling nodes, as demonstrated in Section 2. Thus, we create trace-based workloads where we use Simics/GEMS [16] to generate the network requests of several SPLASH-2 [25] and MineBench [17] applications. These traces contain time-stamped source/destination information for each request. As a compromise, we calculate the total number of requests for each node. Then, we assume the highest traffic node has an injection rate of 1.0, while the other nodes all have an injection rate proportional to its total number of requests. Each node will send reply packets ahead of its own requests, and will be allowed a maximum of 4 outstanding requests before being blocked. We measure the total execution time of the traces as the performance metric. This maintains the unbalanced nature of the traffic load, and in general stress the network more than the time-stamped trace as an injection rate of 1.0 is assumed for the busiest node. Thus it constitutes a pessimistic and conservative evaluation of FlexiShare, as in reality the traffic load might be even lower – requiring even fewer channels in FlexiShare.

Figure 17 shows the performance of a 64-node Radix-16 FlexiShare with varied number of channels. It is clear that for benchmarks *barnes*, *cholesky*, *lu* and *water*, only 2 channels ($M = 2$) are enough to support the traffic load (in contrast to $M = 16$ for conventional nanophotonic crossbars); while *apriori*, *hop* and *radix* need more channels to be provisioned. This shows that FlexiShare can be provisioned according to the average traffic load.

Comparing FlexiShare with R-SWMR and TS-MWSR with double the amount of channels, we find that similar performance can be achieved for low traffic load benchmarks like *lu* and *water*, while FlexiShare performs significantly better in *hop* and *radix*. This can be attributed to the global channel sharing in FlexiShare. Note the topologies here implement a local concentration ($C = 4$) so that channel resources are shared among neighboring nodes; but it is obvious that the *global* sharing in FlexiShare is more effective.

Component	Loss	Component	Loss
Coupler	1 dB	Waveguide Loss	1 dB/cm
Splitter	0.2 dB	Waveguide Crossing	0.05 dB
Non-linear	1 dB	Ring Through Loss	0.001 dB/ring
Modulator-Insertion	0.001 dB	Filter Drop	1.5 dB
		Photo Detector	0.1 dB

Table 3. Optical loss [13]

4.7. Power Model

To show how the reduction in the number of channels affect the power consumption in the nanophotonic crossbars, we adopt the nanophotonic power model by Joshi et al. [13]. In terms of laser power, we model various losses along the path as listed in Table 3. Modulator insertion and non-resonant rings all pose energy losses to the laser beam, together with the length-dependent loss in the waveguide. Varied values have been assumed for the sensitivity of photodetectors in on-chip networks, from $80\mu W$ [11] to $1\mu W$ [26]. In our model, we

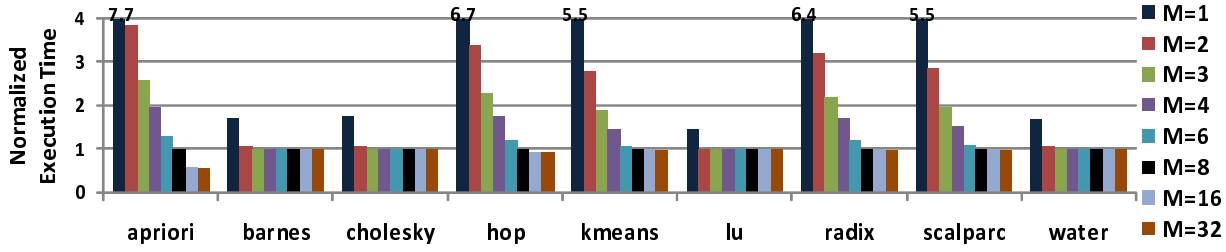

Figure 17. Normalized execution time for FlexiShare ($N = 64, k = 16$) with varied M.

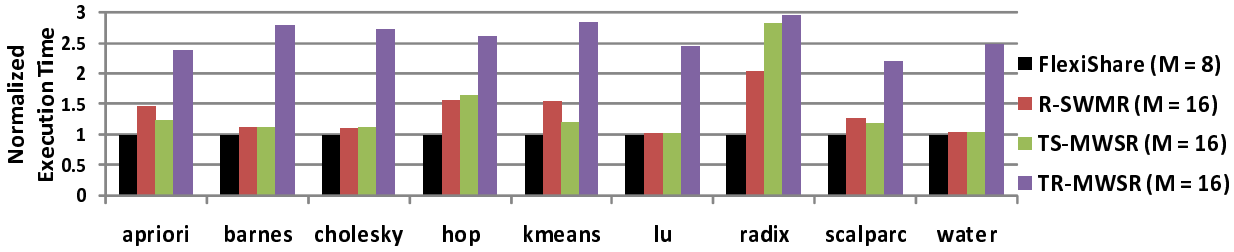

Figure 18. Normalized execution time for various crossbars ($N = 64, k = 16$).

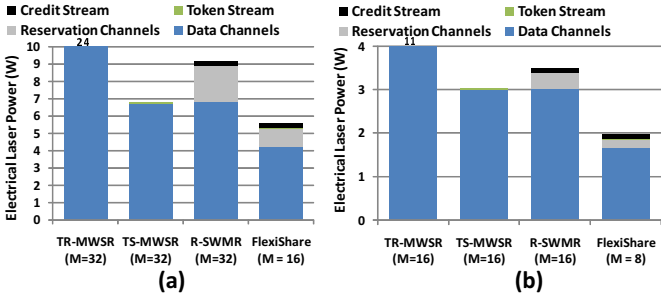

Figure 19. Electrical Laser Power Breakdown for various crossbars (a) $k = 32$ (b) $k = 16$

assume it to be $10\mu W$, in accordance with values adopted by Joshi et al. [13]. Based on these parameters, we layout the waveguides and estimate the per-wavelength laser power needed to activate the farthest detector.

As for ring heating, we also assume $1\mu W$ heating power per ring per Kelvin, and $20K$ tuning range [13]. We estimate the electrical switches in the routers are the dominant part in electrical power, and apply a power model [24] to account for the sizes of the switches. Targeting a $22nm$ technology node based on ITRS [1], we calculate the baseline energy for a 512-bit packet to traverse a 5×5 electrical switch to be $32pJ$.

4.7.1 Laser Power Comparison

As the added flexibility enables FlexiShare to at least match the performance of the alternatives with twice the amount of channels, we compare the power consumption of FlexiShare with half the channels as the alternative designs in Figure 19.

It can be seen that TR-MWSR consumes much higher power, because its twice long waveguides incur more optical loss. The credit streams and token streams consume minimal laser power, due to their respective narrow width compared to the data channel and fewer number of through rings on the path. The broadcasting reservation channels constitutes a major overhead for R-SWMR and FlexiShare. This is especially the case when the crossbar radix is higher. This signifies that a moderate radix is more efficient for FlexiShare and R-SWMR crossbars. Even with all these overheads, FlexiShare with half the amount of channels reduce the total laser power by at least 18% and 35%, for k=32 and k=16, respectively, compared to the best alternatives.

4.7.2 Total Power Comparison

Assuming a uniform average traffic load of $0.1pkt/cycle$, we compare the total energy consumption of different architectures. The results, presented in Figure 20, show that ring heating and laser power are dominant in TR-MWSR, TS-MWSR and R-SWMR, as expected. The added complexity in the electrical routers of FlexiShare results in higher electrical overhead. However, this is incurred for significant power reduction in the dominant laser and ring heating power. The advantage of FlexiShare is especially clear when much fewer channels are provisioned according to the traffic. For example, according to Figure 17 and 18, merely 2 channels ($M = 2$) are enough for a radix-16 FlexiShare to achieve the same performance as the alternatives for *lu* application; and FlexiShare with $M = 4$ performs better for *radix* application. In these cases, radix-16 FlexiShare reduces the total power consumption by 41% and 27%, respectively. The reduction could be up to 72% (when $M = 2$) against a best alternative if radix-32 designs are compared.

4.7.3 Device Requirement

According to Figure 17 and Figure 18, FlexiShare with $M = 4$ outperforms most of the alternative networks. We align their performance and show the device requirements FlexiShare ($M = 4$), R-SWMR ($M = 16$), TS-MWSR ($M = 16$) and TR-MWSR ($M = 16$) would impose under different power budgets, as shown in Figure 21. In this figure, the contour lines are electrical power budgets in watt for generating the laser. By reducing the number of channels provisioned, FlexiShare can

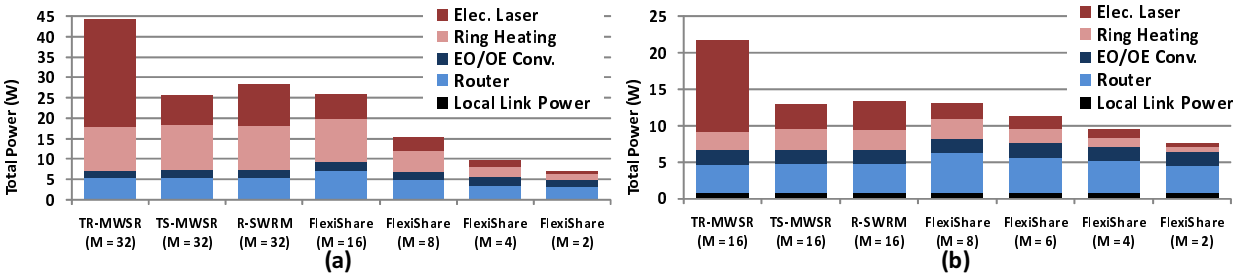

Figure 20. Total Power Breakdown. (a) $k = 32$ (b) $k = 16$.

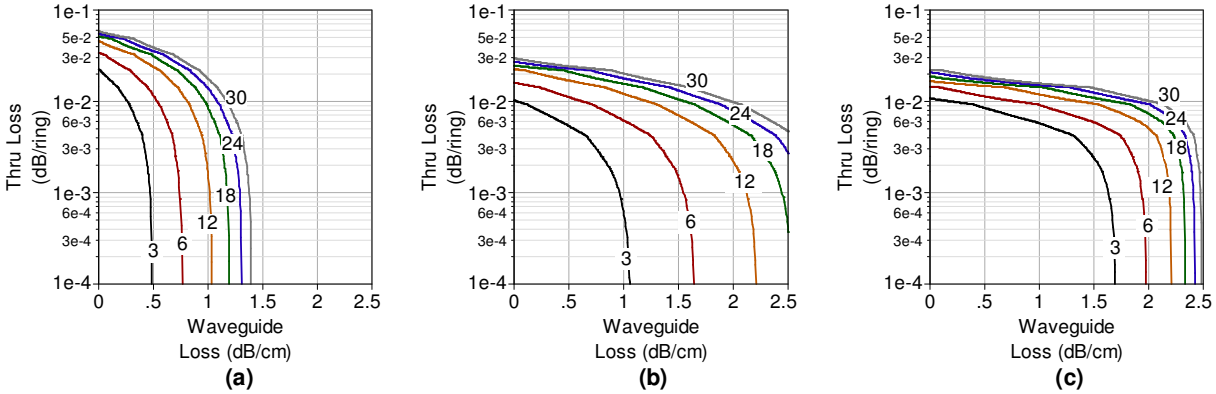

Figure 21. Electrical laser power (W) for (a) TR-MWSR ($k = 16, C = 4, M = 16$), (b) TS-MWSR ($k = 16, C = 4, M = 16$) and (c) FlexiShare ($k = 16, C = 4, M = 4$).

meet an electrical laser power budget as low as 3W with ring through loss of up to 0.011 and waveguide loss of $1.7dB/cm$.

5. Related Work

Recent advances in optical signaling [2, 4, 19, 21] have made the use of on-chip optical signaling a possibility. Different on-chip network architectures have been proposed to exploit silicon nanophotonics. Kirman et al. [14] proposed a 64-node CMP architecture which takes advantage of nanophotonics to create an on-chip bus and results in a hierarchical, multi-bus interconnect. Shacham et al. [20] proposed using an electrical layer for control signals while the channels and the switches for data transmission use optical signaling. The Corona [23] architecture implements a monolithic crossbar topology to support on-chip as well as off-chip communication. Another crossbar structure was proposed by Batten et al. [5] to connect a many-core processor to the DRAM memory using monolithic silicon. Recently, Joshi et al. [13] proposed to implement a nanophotonic clos network that provides uniform latency and throughput while consuming low power. Nanophotonic switching is explored in the Phastlane [8] work. The hierarchical Firefly architecture [18] was proposed to use partitioned nanophotonic crossbars to connect clusters of electrically connected mesh networks, so as to improve power efficiency. Zheng et al. [26] implemented a hybrid nanophotonic-electric network where latency-critical messages are broadcast in a planar optical waveguide using a unique optical antenna structure. The FlexiShare architecture proposed in this paper is an efficient implementation of a nanophotonic crossbar, that can be leveraged by existing topologies to improve energy efficiency.

Resource sharing is a common technique that has been used in interconnection network and different aspects of sharing has been proposed for on-chip networks. XShare [10] switch architecture was proposed to combine multiple short packets and send them together on a wide channel. For on-chip networks, concentrated mesh [3] topology was proposed to reduce the network diameter and improve channel utilization by sharing routers among multiple nodes. Kumar et al. [15] studied different implementations of concentration and its the impact on the network efficiency. In this work, we extend the concept of concentration to provide *global* concentration such that the resources (channels and buffers) are shared across the network.

Load-balanced Birkhoff-von Neumann switches were proposed by Cheng et al. [7], where back-to-back crossbars are used with the first crossbar for load-balancing and a buffer stage inserted in between. Flexishare has a similar logical structure but our main objective is reducing the amount of global bandwidth (i.e., optical channels) to improve the power efficiency of nanophotonics. In addition, the arbitration of the crossbar in FlexiShare is not a centralized scheduler but distributed and leverages nanophotonics with novel token-stream arbitration schemes.

6. Conclusion

Recent advances in nanophotonics have presented opportunities for architects to exploit the benefits of nanophotonics for

on-chip communication in future many-core processors. However, nanophotonics poses different constraints compared to conventional, electrical on-chip network as the channels become an expensive resource because of the static power overhead. Motivated by this constraint and the unbalanced traffic observed in different applications, we propose FlexiShare – an energy-efficient crossbar design which provides the flexibility to provision the amount of channel resource independent of the network size. By sharing the channels among all of the nodes in the network – creating *global* concentration, the number of channels required can be reduced to mitigate the static power overhead of nanophotonics. FlexiShare decouples the allocation of buffers from the channels to enable global sharing of the buffers as well. A novel token-stream mechanism is used for channel arbitration to better utilize the channels as well as distributed credit information for efficient buffer management. By applying this arbitration scheme to a conventional MWSR crossbar, the throughput is improved by up to $5.5\times$ compared to a token-ring arbitrated network. Flexishare introduces additional complexity to support the global sharing and results in moderate increase in electrical power consumption compared to a conventional nanophotonic crossbar. However, because of the significant reduction of the amount of channels we can achieve an overall power reduction of at least 27% (up to 72%) while providing similar or better performance than the conventional nanophotonic crossbars under unbalanced trace traffic loads.

Acknowledgements

This work is supported by NSF grants CCF-0916746 and CCF-0747201. Special thanks to Yu Zhang for generating the NoC traffic traces in our evaluation and Dr. Ajay Joshi and Dr. Christopher Batten for their help with the nanophotonic power models. We would also like to thank all the anonymous referees for their helpful comments.

References

[1] *2007 International Technology Roadmap for Semiconductors(ITRS)*.

[2] V. Almeida, C. Barrios, R. Panepucci, M. Lipson, M. Foster, D. Ouzounov, and A. Gaeta. All-optical switching on a silicon chip. *Optics Letters*, 29:2867–2869, 2004.

[3] J. Balfour and W. J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proc. of the Int'l Conference on Supercomputing (ICS)*, pages 187–198, Carns, Queensland, Australia, 2006.

[4] C. A. Barrios, V. R. Almeida, and M. Lipson. Low-power-consumption short-length and high-modulation-depth silicon electro-optic modulator. *Journal of Lightwave Technology*, 21(4):1089–1098, 2003.

[5] 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 *Proc. of Hot Interconnects*, pages 21–30, Stanford, CA, 2008.

[6] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. Mackay, M. Reif, L. Bao, J. Brown, M. Mattina, C.-C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 processor: A 64-core soc with mesh interconnect. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE Int'l*, pages 88–598, 2008.

[7] C.-S. Chang, D.-S. Lee, and C.-M. Lien. Load balanced Birkhoff-von Neumann switches with resequencing. In *ACM SIGMETRICS Performance Evaluation Review*, volume 29, 2001.

[8] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi. Phastlane: A rapid transit optical routing network. In *Proc. of the Int'l Symposium on Computer Architecture (ISCA)*, Austin, TX, 2009.

[9] W. J. Dally and T. B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishing Inc., 2004.

[10] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *Int'l Symposium on High-Performance Computer Architecture (HPCA)*, pages 175–186, Raleigh, NC, USA, Feb. 2009.

[11] R. K. Dokania and A. B. Apsel. Analysis of challenges for on-chip optical interconnects. In *GLSVLSI '09: Proceedings of the 19th ACM Great Lakes symposium on VLSI*, pages 275–280, New York, NY, USA, 2009. ACM.

[12] P. Gratz, C. Kim, R. McDonald, S. Keckler, and D. Burger. Implementation and evaluation of on-chip network architectures. In *Int'l Conference on Computer Design (ICCD)*, pages 477–484, San Jose, CA, 2006.

[13] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic. Silicon-photonic clos networks for global on-chip communication. In *IEEE Int'l Symposium on Network-on-Chip (NOCS)*, San Diego, CA, 2009.

[14] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins, and D. H. Albonesi. Leveraging optical technology in future bus-based chip multiprocessors. In *IEEE/ACM Int'l Symposium on Microarchitecture (MICRO)*, pages 492–503, Orlando, FL, 2006.

[15] P. Kumar, Y. Pan, J. Kim, G. Memik, and A. Choudhary. Exploring concentration and channel slicing in on-chip network router. In *IEEE Int'l Symposium on Network-on-Chip (NOCS)*, San Diego, CA, 2009.

[16] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *ACM SIGARCH Computer Architecture News*, 33(4):92–99, Sep. 2005.

[17] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary. Minebench: A benchmark suite for data mining workloads. In *IEEE Int'l Symposium on Workload Characterization (IISCW)*, pages 182–188, San Jose, CA, 2006.

[18] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary. Firefly: Illuminating future network-on-chip with nanophotonics. In *Proc. of the Int'l Symposium on Computer Architecture (ISCA)*, Austin, TX, 2009.

[19] L. Pavesi and D. J. Lockwood. *Silicon Photonics, Volume 94 of Topics in applied physics*. Springer-Verlag, 2004.

[20] A. Shacham, K. Bergman, and L. P. Carloni. The case for low-power photonic networks-on-chip. In *Proc. of Design Automation Conference (DAC)*, pages 132–135, San Diego, CA, 2007.

[21] J. Tatum. Vcsels for 10 gb/s optical interconnects. In *IEEE Emerging Technologies Symposium on BroadBand Communications for the Internet Era*, pages 58–61, Richardson, TX, 2001.

[22] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-Tile sub-100-w teraflops processor in 65-nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(1):29–41, 2008.

[23] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. L. Binkert, R. G. Beausoleil, and J. H. Ahn. Corona: System implications of emerging nanophotonic technology. In *Proc. of the Int'l Symposium on Computer Architecture (ISCA)*, pages 153–164, Beijing, China, 2008.

[24] H.-S. Wang, L.-S. Peh, and S. Malik. A power model for routers: Modeling alpha 21364 and infiniband routers. *IEEE Micro*, 23(1):26–35, 2003.

[25] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proc. of the Int'l Symposium on Computer Architecture (ISCA)*, pages 24–36, Santa Margherita Ligure, Italy, Jun. 1995.

[26] L. Zheng, A. Mickelson, L. Shang, M. Vachharajani, D. Filipovic, W. Park, and Y. Sun. Spectrum: A hybrid nanophotonic-electric on-chip network. In *Proc. of Design Automation Conference (DAC)*, San Francisco, CA, Jun 2009.