A Reconfigurable Silicon-Photonic Network with Improved Channel Sharing for Multicore Architectures

Sai Vineel Reddy Chittamuru, Srinivas Desai, Sudeep Pasricha
Department of Electrical and Computer Engineering
Colorado State University, Fort Collins, CO, U.S.A.
{sai.chittamuru, srinivas.desai, sudeep}@colostate.edu

ABSTRACT

On-chip communication is widely considered to be one of the major performance bottlenecks in contemporary chip multiprocessors (CMPs). With recent advances in silicon nanophotonics, photonic-based networks-on-chip (NoCs) are being considered as a viable option for communication in emerging CMPs as they can enable higher bandwidth and lower power dissipation compared to traditional electrical NoCs. In this paper, we present *UltraNoC*, a novel reconfigurable silicon-photonic NoC architecture that features improved channel sharing and supports dynamic re-prioritization and exchange of bandwidth between clusters of cores running multiple applications, to increase channel utilization and performance. Experimental results show that *UltraNoC* improves throughput by up to 9.8× while reducing latency by up to 55% and energy-delay product by up to 90% over state-of-the-art solutions.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Network topology*; C.4 [Performance of Systems]: design studies

Keywords

Network-on-chip (NoC), photonic channel sharing, arbitration

1. INTRODUCTION

Advances in technology scaling over the past decade have enabled the integration of billions of transistors on a single die. Such a massive number of transistors has allowed multiple processing cores and more memory to be integrated on a chip, to meet rapidly growing performance demands of modern applications. With hundreds of on-chip cores expected to become a reality in the near future, traditional electrical network-on-chip (NoC) communication fabrics [1], [2] are projected to suffer from cripplingly high power dissipation and severely reduced performance [7]. Crosstalk and electromagnetic interference in metallic interconnects will further worsen the performance and reliability of NoCs with technology scaling.

Recent developments in the area of silicon photonics have enabled their integration with CMOS circuits. On-chip photonic links provide several prolific advantages over their metallic counterparts, including light speed transfers, high bandwidth [3] density by using dense wavelength division multiplexing (DWDM) [3], low power dissipation [4], and low crosstalk [7]. Thus silicon photonics is being considered as an exciting new option for future NoCs. Several photonic devices such as microring resonators, waveguides and photodetectors have already been fabricated and demonstrated at the chip level [5], [6].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GLSVLSI'15, May 20–22, 2015, Pittsburgh, PA, USA.
Copyright © 2015 ACM 978-1-4503-3474-7/15/05…$15.00.
http://dx.doi.org/10.1145/2742060.2742067

These devices have been used as a foundation for several photonic network architectures [8]-[11].

A few prior works emphasize the importance of network resource contention in photonic channels and proposed arbitration techniques to resolve contention [8], [10]. However, a limitation of these approaches is that they do not fully exploit available network bandwidth. Another limitation is that these approaches typically only target single parallel application workloads. In emerging multicore systems where multiple applications execute simultaneously on unique subsets of cores, there is significantly greater variation in temporal and spatial characteristics of network injected traffic. For example, cores running memory intensive tasks can require more network bandwidth than cores running compute intensive tasks [24].

To overcome these shortcomings, we propose a novel photonic NoC architecture called *UltraNoC* that utilizes multiple-write-multiple-read (MWMR) photonic waveguides in a crossbar topology, and supports dynamic performance adaptation to aggressively utilize network bandwidth and meet diverse application demands. We compare *UltraNoC* against architectures with the best-known arbitration mechanisms, for multi-threaded PARSEC [16] workloads on CMP platform sizes ranging from 64-cores to 256-cores. The novel contributions of this paper are:

- A flexible on-chip photonic network architecture (*UltraNoC*) that facilitates selective and reconfigurable prioritization of applications based on their time-varying performance goals;
- A concurrent token stream arbitration that provides multiple simultaneous tokens and increases channel utilization;
- A dynamic bandwidth transfer technique with low overhead, to transfer unused bandwidth among clusters of cores;
- A mechanism to monitor traffic being injected into the network by different co-running applications, to facilitate dynamic arbitration wavelength injection rate modulation.

2. RELATED WORK

A considerable amount of work has focused on the area of photonic NoC design. Several efforts have explored *high-radix low-diameter* photonic crossbar architectures that provide non-blocking connectivity, e.g., [8], [10], [15] and [21]; whereas silicon photonic implementations of *low-radix high-diameter* NoCs have also been investigated in [14], [17], [18], and [20]. Prior work has shown that photonic crossbars are extremely promising architectures to meet future on-chip bandwidths demands, but they can suffer from *(i)* large power dissipation and *(ii)* high contention for resources, especially when using inefficient token-based arbitration schemes. A few techniques to reduce power overhead of photonic NoCs have been proposed in literature. For example, an effective policy for runtime management of the laser source is proposed in [11]. We build on their work to manage power in photonic crossbar NoCs. To reduce contention issues in crossbars, a few improved arbitration techniques have been proposed in [10], [22] that use time division multiplexing (TDM), so that a single data waveguide can be simultaneously used by more than one node in different time slots.

In Flexishare [10], a token stream arbitration scheme is proposed. The scheme requires wavelengths corresponding to each data waveguide to be injected serially into different time slots of an arbitration waveguide. A node writes on the data waveguide only when it gets access to the corresponding arbitration wavelength. Subsequently, the node cannot send data again till its arbitration wavelength is injected into the arbitration waveguide, which takes *N* cycles for *N* data waveguides. The scheme leads to channel under-utilization, and performs even worse as the number of nodes and waveguides increase. In [22], the token ring arbitration scheme from Corona [8] was improved with the token channel and token slot arbitration techniques for Multiple-write-single-read (MWSR) crossbars. Token slot arbitration uses TDM and improves upon token channel arbitration by dividing the arbitration waveguide into fixed-size, back-to-back slots, with destination nodes circulating tokens in one-to-one correspondence to slots. A limitation of this approach is that a fixed time gap is required between two arbitration slots to set up data for transmission, which reduces the available time slots to send data. *UltraNoC* improves upon these prior works by utilizing a more powerful concurrent token stream arbitration strategy, together with reconfigurable cluster prioritization and bandwidth re-allocation to improve MWMR photonic channel utilization.

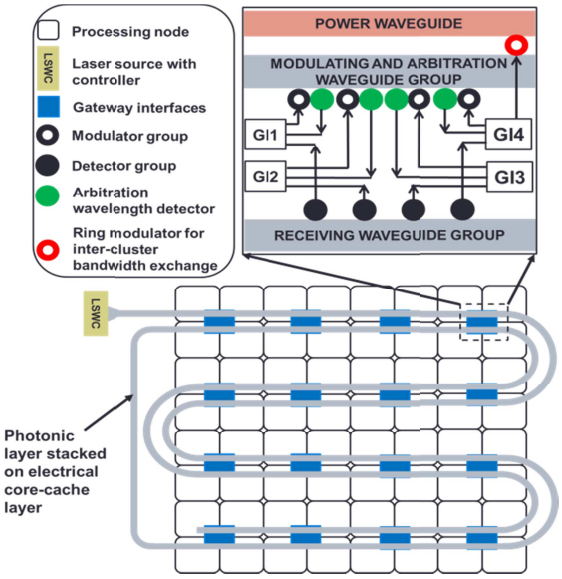

Figure 1: Layout of MWMR crossbar used in UltraNoC along with the arrangement of cores and their respective gateway interfaces.

3. *ULTRANOC* PHOTONIC NOC OVERVIEW

3.1 Architecture and Terminology

The baseline *UltraNoC* architecture is targeted to a 64-core CMP, as shown in figure 1. We also extend the baseline architecture to a 256-core CMP for the purposes of scalability analysis. Each core has a private L1 and shared L2 cache, with a MOESI protocol to maintain cache coherence. In a 64-core CMP, each group of 8 cores has access to main memory via a dedicated memory controller, whereas in a 256-core CMP, each group of 16 cores has a dedicated memory controller. A node is defined as an entity consisting of one and four cores for the 64-core and 256-core CMP, respectively. Every node in *UltraNoC* is attached to a gateway interface (GI) module that facilitates transfers between the CMOS electrical layer and a photonic layer (with photonic waveguides, modulators, detectors, etc.). The entire chip is divided into four clusters (C_0, C_1, C_2, C_3), where each cluster contains 16 cores in a 64-core CMP and 64 cores in a 256-core CMP.

A detailed layout of the *UltraNoC* architecture is shown in figure 1, where 64 nodes are arranged in an 8×8 mesh. Communication between cores within a node for the 256-core CMP uses an electrical 5×5 NoC router, where four of its input and output port pairs are connected to four cores and the fifth input/output port pair is connected to a GI module. A round-robin arbitration scheme is used within each node for communication between cores and the GI.

Inter-node transfers are facilitated by dual-coiled MWMR waveguide groups, where each group has four MWMR waveguides by default, with each waveguide supporting 64 wavelength DWDM, to facilitate simultaneous transfer of a total of 512 bits of data (cache line) with data modulation at both clock edges. Each MWMR waveguide group in *UltraNoC* passes every node twice in the dual coiled structure to enable a two pass inter-node data communication. A node has the ability to write on the first pass using its ring modulators and read from the waveguide group using its ring detectors in the second pass.

As all nodes are capable of modulating (writing) in an MWMR waveguide group during the first pass, there is a need for arbitration (see Section 3.2 for more details) between sending nodes to ensure that the data of different senders does not destructively overlap on the shared waveguide group. Throughout this paper this first pass portion of the waveguide group is referred to as the *modulating and arbitration waveguide group*. In the second pass of the MWMR waveguide group all nodes receive data through their respective ring detectors; hence this portion of the waveguide group is referred to as the *receiving waveguide group*. Additionally there is a *power waveguide* that runs in parallel with the other waveguides, and carries arbitration wavelengths. This waveguide facilitates our bandwidth transfer and priority adaptation techniques (see Sections 3.3, 3.4).

Figure 1 depicts an expanded view of the collection of GIs for four nodes, which shows the modulating and arbitration, receiving, and power waveguide groups, along with their connection to GIs. There are 64 wavelengths that are utilized for data communication in each MWMR waveguide within each waveguide group, so each ring modulator and detector group has 256 ring modulators and 256 ring detectors, respectively. Each of the 64 wavelengths is assigned to a unique receiving node, such that whenever a receiver detects its corresponding wavelength during a clock cycle, it switches its detectors "on" to receive data in the next clock cycle. All other receivers keep their detectors turned off to save power. We also utilize four arbitration wavelengths, each corresponding to a cluster. For powering the waveguides, we use a broadband off-chip laser source with a laser power controller (LSWC). The LSWC has groups of ring modulators capable of injecting different wavelengths in different clock cycles. As we use 68 DWDM in each MWMR waveguide, there are 68 ring modulators in each ring modulator group in the LSWC. The laser controller also has control logic that can alter the rate of injection of arbitration wavelengths into each waveguide group. More details about LSWC are presented in the next subsections and overhead analysis is presented in Section 4.1.

3.2 MWMR Concurrent Token Stream Arbitration

In *UltraNoC* all of the cores on a chip are partitioned into four clusters (C_0 – C_3) and each cluster is assigned a dedicated arbitration wavelength ($\lambda_0 - \lambda_3$). Each MWMR waveguide group is divided into a fixed number of time slots, based on the time taken by light to traverse the waveguide on a die, which is 8 cycles in our architecture. Thus we divide each MWMR waveguide group into 8 time slots. The time slots are classified into three types: *arbitration slot*, *receiver prediction slot*, and *data slot*. For simplicity, figure 2(a) shows an example of the distribution of time slots across 4 MWMR waveguide groups, even though a

minimum of 8 MWMR waveguide groups are used in our architecture. In the arbitration slot, the LSWC injects arbitration wavelengths of clusters, selectively using a modulator group to dedicate the arbitration slot to a particular cluster. After a sending node grabs an arbitration wavelength in the arbitration slot, it gets access to the next receiver prediction slot to release the corresponding wavelength of its receiving node. Subsequently, in the next data slot, the sending node modulates data on the 64 wavelengths in each waveguide group assigned for data transfer.

As an example, suppose N_1 in cluster C_0 needs to send data to N_{32} that has a corresponding receiver prediction wavelength w_{32}. N_1 first grabs arbitration wavelength λ_0 which is dedicated to cluster C_0, in the arbitration slot. N_1 then modulates in the next receiver prediction slot, such that only w_{32} (the dedicated wavelength for receiver prediction of N_{32}) is made available by removing all the wavelengths except w_{32} (using its ring modulators) in that receiver prediction slot. On the receiving end at N_{32}, only the detector for wavelength w_{32} is switched on. Once w_{32} is detected, N_{32} prepares to receive data in the next data slot by switching on the remaining detectors in that node. Figure 2(b) shows the position of different slots in the MWMR waveguide group W1 at time cycle 3 for the example in figure 2(a). As our architecture divides each MWMR waveguide group into 8 slots, each slot covers 8 nodes in a particular time instance. The stream of tokens on concurrent slots in waveguide groups allows multiple nodes to inject packets simultaneously, resulting in extremely high channel utilization in the MWMR waveguides. A round-robin arbiter is used within each cluster to resolve contention among the 16 nodes in a cluster, and avoid starvation.

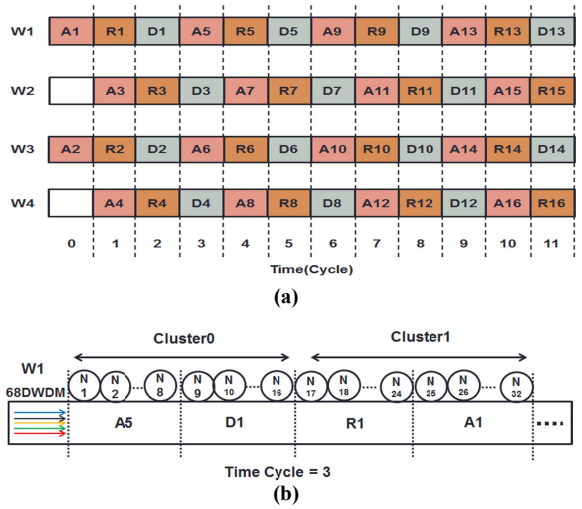

Figure 2: (a) Timing diagram of arbitration in UltraNoC, which shows distribution of arbitration (Ai), receiver prediction (Ri)and data slots (Di) across four MWMR waveguide groups (W1–W4); (b) distribution of different slots within MWMR waveguide group W1 at time cycle 3.

3.3 Inter-cluster Bandwidth Exchange

UltraNoC supports inter-cluster bandwidth transfers to further improve channel utilization and overall performance. As an example, if cluster C_0 does not need to transfer data, it transfers its bandwidth to a subsequent cluster C_1. Similarly any cluster can transfer its unused bandwidth to subsequent clusters. Figure 3 presents an overview of the bandwidth transfer technique. The last node in each cluster is provisioned with an extra modulator that is capable of injecting an arbitration wavelength of the next cluster. Whenever a ring detector of the last node of a cluster detects its own arbitration wavelength, and if this node does not have data to transfer, this is a case where the cluster has not used its bandwidth.

To improve channel utilization, the cluster can transfer its unused bandwidth to the next cluster, by having the last node of the cluster inject the arbitration wavelength of the next cluster into the same arbitration slot. To simultaneously remove and inject arbitration wavelengths at the last node of cluster, the arbiter uses information about the current slot of the waveguide and the cluster it belongs to. This information can change across interval windows based on the data received from LSWC (Section 3.4).

Figure 3 illustrates an example of bandwidth transfer. Clusters C_0–C_3 are assigned arbitration wavelengths highlighted with green, yellow, blue, and red respectively. The last node in the first three clusters (node16, node32, and node48) is shown with an extra ring modulator to facilitate the injection of the arbitration wavelength of the next cluster. For this example, nodes in C_0 do not need to transfer any data in the current cycle. Then node16, which is the last node in C_0 removes its cluster's arbitration wavelength (green) from the arbitration slot and injects the arbitration wavelength of C_1 (yellow) so that nodes in C_1 can use this arbitration slot for sending data in the next available data slot. Figure 3 also shows counters at nodes 16, 32, 48, and 64 that are used to count the arbitration wavelength conversions over a time interval. The next subsection presents details about these counters.

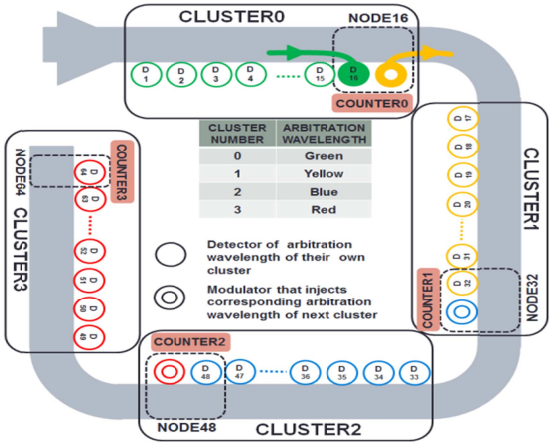

Figure 3: Bandwidth transfer technique: a cluster can transfer its unused bandwidth to the next cluster by absorbing its own arbitration wavelength and releasing arbitration wavelength of next cluster.

3.4 Cluster Priority Adaptation with LSWC Reconfiguration

UltraNoC also supports runtime alteration of allocated bandwidth to each cluster, to closely track changing application bandwidth needs, by altering the number of arbitration slots dedicated to each cluster (i.e., cluster priority). This is essential because while our bandwidth transfer technique can transfer unused arbitration slots from one cluster to another in the direction of concurrent arbitration token stream flow (e.g., C_0 to C_1), it lacks the ability to transfer bandwidth in the opposite direction (e.g., C_1 to C_0). To overcome this limitation, we design a cluster priority adaptation mechanism to more comprehensively manage cluster bandwidth allocations over time. This mechanism also helps to minimize laser power by intelligently reducing the total number of injected arbitration slots for scenarios with low bandwidth requirements. The cluster priority adaptation technique consists of three main steps, as discussed below:

Step 1: Determination of wavelength conversion count: Each cluster C_0 – C_3 has an associated weight w_0 – w_3, which determines the proportion of arbitration slots (and consequently bandwidth or priority) assigned to the cluster. Initially, these weights can be set to be equal, i.e., 0.25 each. At runtime, whenever the last node in a cluster performs a wavelength

conversion from its current cluster arbitration wavelength to the next cluster arbitration wavelength, a counter (shown in figure 3) is incremented. This conversion event represents the case where an unused arbitration slot (bandwidth) is transferred from one cluster to another. Over a time interval *T*, the recorded wavelength conversion counts WCC_0 – WCC_3 from each cluster are then used to determine unused bandwidth of each cluster.

Step 2: Calculation of excess arbitration slots: The wavelength conversion count values of different clusters show the aggregated number of excess arbitration slots, which includes excess arbitration slots of the present cluster along with the excess arbitration slots of predecessor clusters. Therefore the excess arbitration slots for the i^{th} cluster (ES_i) are calculated using equation (1) shown below, by subtracting the cluster wavelength conversion count of the predecessor cluster (WCC_{i-1}) from the wavelength conversion count of the cluster under consideration (WCC_i). ES_i values can also be negative, when a cluster consumes a greater number of arbitration slots (made available by predecessor clusters) than its allocated arbitration slots. Such a cluster has a deficit of arbitration slots.

$$ES_i = \begin{cases} WCC_i, & i = 0 \\ WCC_i - WCC_{i-1}, & i > 0 \end{cases} \quad (1)$$

Step 3: Setting new weight (priority) for each cluster: Based on the estimation of excesses and deficits in arbitration slots assigned across clusters, this final step attempts to adjust weight values of each cluster to eliminate the excesses and deficits. To determine the new weight of the i^{th} cluster w_i(next) for the upcoming time interval, we must subtract the excess weight EW_i of the cluster from its current weight w_i(current). We can calculate EW_i by dividing the excess arbitration slots of the i^{th} cluster (ES_i), calculated in equation (1), by the total number of arbitration slots released in the time interval *T*, which we denote as K. The equations below show these calculations:

$$EW_i = ES_i / K \quad (2)$$

$$w_i(\text{next}) = w_i(\text{current}) - EW_i \quad (3)$$

Based on the values of the new weights, the LSWC changes the distribution of arbitration wavelengths injected for the next time interval *T*, such that a cluster with a higher weight will receive more arbitration wavelengths, and has more opportunities to use the waveguides for data transfer. The weight values are also communicated to all clusters, so that arbiters can adjust their local counters to match the new arbitration slot profile in waveguides.

4. EXPERIMENTS

4.1 Experimental Setup

To evaluate our proposed *UltraNoC* architecture, we compared it to an electrical mesh (EMesh) based NoC as well as to two state-of-the-art photonic crossbar NoCs that use smart arbitration techniques: Flexishare with token stream arbitration [10] and Corona with an enhanced token-slot arbitration [22]. We modeled and simulated the architectures at a cycle-accurate granularity with a SystemC-based NoC simulator, for two CMP platform complexities: 64-core and 256-core. We used the PARSEC benchmark suite [16] to create multi-application workloads, with clusters running parallelized versions of different benchmarks.

Table 1 shows the PARSEC benchmarks we considered, classified into three categories according to their memory intensities. *Compute intensive* benchmarks spend most of the time computing and less time communicating with memory; whereas *memory intensive* applications spend a larger portion of their execution time communicating with memory and less time computing within cores. *Hybrid* intensity benchmarks demonstrate both compute and memory intensive phases. We created 12 multi-application workloads from these benchmarks. Each workload combines 4 benchmarks, and the memory intensity of the workloads varies across the spectrum, from compute intensive to memory intensive. As an example, the SC-BT-BS-VI workload combines parallelized implementations of Streamclusters (SC), Bodytrack (BT), Blackscholes (BS) and Vips (VI), and executes them in clusters C_0, C_1, C_2, and C_3, respectively. Each parallelized benchmark is executed on a group of 16 cores and 64 cores, in the 64-core and 256-core CMP platforms, respectively. Full-system simulation of parallelized PARSEC benchmarks using gem5 [25] was used to generate traces that were fed into our cycle-accurate network simulator. We set a "warm-up" period of 100M instructions and captured traces for 1B instructions.

Table 1: Memory intensity classification of PARSEC benchmarks

Application	Representation	Workload Type
Blackscholes	BS	Compute intensive
Bodytrack	BT	Compute intensive
Vips	VI	Compute intensive
Dedup	DU	Compute intensive
Freqmine	FQ	Hybrid
Ferret	FR	Hybrid
Fluidanimate	FA	Hybrid
X264	X264	Hybrid
Streamclusters	SC	Memory intensive
Canneal	CA	Memory intensive
Facesim	FS	Memory intensive
Swaptions	SW	Memory intensive

We targeted 32nm and 22nm process technologies for the 64-core and 256-core CMPs, respectively. Based on the geometric calculation of the waveguides for a 20mm×20mm chip dimension, we estimated the time needed for light to travel from the first to the last node in a single pass of the MWMR waveguide group in *UltraNoC* as 8 cycles at 5 GHz clock frequency. The same clock and 8 cycle round trip time is also applicable to the waveguides in the Flexishare and Corona photonic crossbar NoCs. Throughout our analysis we use a flit size of 64 bits for EMesh and a total packet size of 512 bits for all photonic NoC architectures. We consider data modulation at both clock edges to enable simultaneous transfer of 512 bits in a single cycle, in the UltraNoC, Flexishare, and Corona architectures.

The static and dynamic energy consumption of electrical routers is based on results obtained from the open-source DSENT tool. Energy consumption of various photonic components for all the photonic NoC architectures are adopted from photonic device characterizations in line with state-of-the-art proposals [19], [23] and shown in Table 2. Here $E_{dynamic}$ is the energy/bit for modulators and photodetectors and $E_{logic-dyn}$ is the energy/bit for the driver circuits of modulators and photodetectors. P_{MWMR} and P_{MWSR} are the static power consumption of an MWMR and an MWSR waveguide group, respectively which includes the power overhead of ring resonator thermal tuning. We consider a ring heating power of 15 µW per ring and detector responsivity of 0.8 A/W [19]. To compute laser power consumption, we calculated photonic loss in components, which sets the photonic laser power budget and correspondingly the electrical laser power.

Finally, based on our gate-level analysis, area overhead due to electrical circuitry (e.g., adders, multipliers) in LSWC is estimated to be $0.011mm^2$ at 32nm. We set the reconfiguration delay overhead in *UltraNoC* to be 20 cycles to account for the time to transfer wavelength conversion counter values from each cluster to LSWC, time to determine new priority weights of each cluster, and to update these values in arbiters in each cluster. We set reconfiguration time interval window size in *UltraNoC* to 300 cycles, to balance reconfiguration overhead and performance.

Table 2: Energy and losses for photonic devices [19], [23]

Energy consumption type	Energy
$E_{dynamic}$	0.42 pJ/bit
$E_{logic-dyn}$	0.18 pJ/bit
Static power per waveguide group	**Power**
P_{MWMR} (with 68 DWDM)	3.86 W
P_{MWMR} (with 64 DWDM)	3.73 W
P_{MWSR} (with 64 DWDM)	2.35 W
Photonic loss type	**Loss (in dB)**
Microring through	0.02
Waveguide propagation per cm	1
Waveguide coupler/splitter	0.5
Waveguide bending loss	0.005 per 90^0

4.2 Experimental Results

Our experiments target a 64-core CMP platform and compare network throughput, average packet latency, and energy-delay product (EDP) of *UltraNoC* with the electrical mesh (EMesh), Flexishare with token stream arbitration [10], and Corona with token-slot arbitration [22]. We consider two variants of our architecture: *UltraNoC-8* which uses 8 waveguide groups and *UltraNoC-16* which uses 16 waveguide groups. Figures 4(a)-(c) show the results of this study, with all results normalized with respect to the EMesh results. From the throughput comparison in Figure 4(a), it can be observed that, not surprisingly, all photonic NoCs provide better throughput than EMesh, due to the presence of higher bandwidth photonic links. *UltraNoC* with 8 MWMR waveguide groups (*UltraNoC*-8) has nearly 3× greater throughput compared to Flexishare, with the same number of MWMR waveguides. Even though Flexishare uses MWMR waveguides and time division multiplexing (TDM) as in *UltraNoC*, there are significant differences between the architectures.

In Flexishare, arbitration wavelengths corresponding to MWMR data waveguides are injected serially into the arbitration waveguide and a node that grabs a token in the arbitration waveguide gets exclusive access to the corresponding MWMR data waveguide; whereas in *UltraNoC,* multiple arbitration, reservation and data slots are available concurrently in an MWMR waveguide, such that each MWMR waveguide can be accessed simultaneously by multiple nodes. Moreover, unlike *UltraNoC*, Flexishare does not support priority reconfiguration and bandwidth exchange mechanisms.

UltraNoC-8 also provides higher throughput than Corona. This is because of instances in Corona, where multiple sender nodes attempt to communicate with a single receiver node (e.g., memory controller). Such instances result in the sender nodes attempting to access the single MWSR waveguide connected to the receiver, creating a significant imbalance among MWSR waveguides, with the other waveguides being underutilized while packets get queued waiting for the waveguide connected to the receiver. *UltraNoC* avoids such an imbalance with its use of more efficient MWMR waveguides and improved arbitration.

The *UltraNoC* configuration with 16 waveguide groups (*UltraNoC*-16) with approximately twice the photonic hardware of *UltraNoC*-8 provides even better throughput than the other architectures. On average *UltraNoC*-8 has 2.8×, 1.6×, and 3.8× higher throughputs compared to Flexishare, Corona and EMesh, respectively. *UltraNoC*-16 has 5.3×, 3.2×, and 7.7× higher throughputs compared to Flexishare, Corona and EMesh, respectively. The improvement is somewhat higher for memory intensive workloads than for compute intensive workloads. The large throughput improvement for *UltraNoC* is a direct consequence of avoiding unused bandwidth by transferring it to cores that need it the most, using the bandwidth transfer and priority alteration mechanisms at runtime. These mechanisms also improve the average packet latency in *UltraNoC*, as shown in Figure 4(b), by reducing the time spent waiting for access to the photonic waveguides. On average *UltraNoC*-8 has 17%, 32% and 47% and *UltraNoC*-16 has 25%, 40% and 51% lower average packet delay over Flexishare, Corona and EMesh, respectively for the different multi-application workloads.

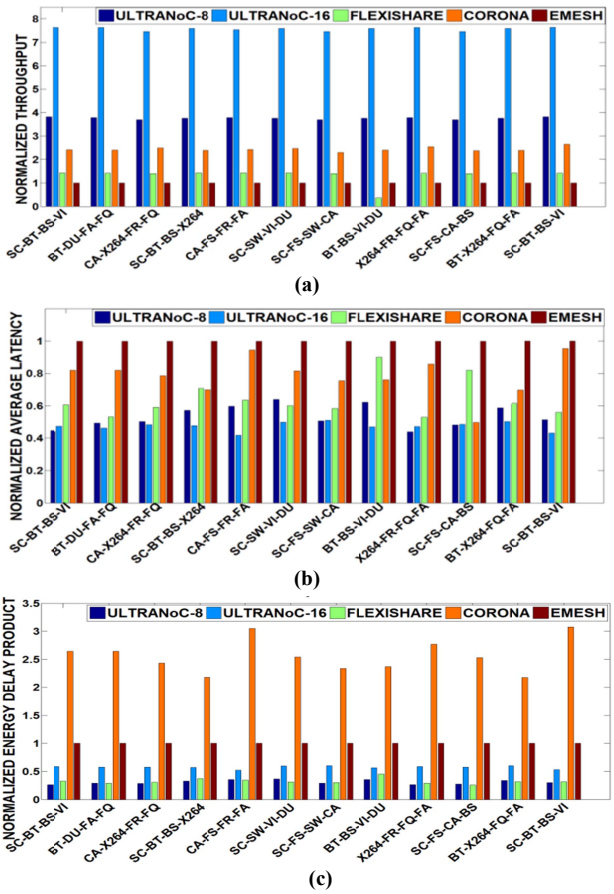

Figure 4: (a) throughput (b) latency (c) EDP comparison of UltraNoC-8 and UltraNoC-16 with other architectures for a 64-core CMP. Results are normalized wrt EMesh.

Figure 4(c) shows the energy-delay product (EDP) comparison between the architectures. It can be observed that on average *UltraNoC*-8 has 90%, 70% and 10% lower EDP compared to Corona, EMesh, and Flexishare respectively. Further *UltraNoC*-16 has 79% and 45% lower EDP compared to Corona and EMesh. Most of the energy in the photonic architectures was consumed in the form of static energy. A comparison of the photonic hardware between the network architectures (detailed results for modulator and detector ring counts are omitted due to lack of space) shows that *UltraNoC-8* and *UltraNoC-16* have 75% and 50% less photonic hardware compared to Corona but *UltraNoC-16* uses more hardware than Flexishare. This explains why *UltraNoC-16* has a higher EDP than Flexishare, because *UltraNoC-16* uses more photonic hardware (and thus consumes more static energy) than Flexishare, even though the average packet delay for *UltraNoC-16* is lower than Flexishare.

Our final set of experiments explores architecture scalability. We considered a larger 256-core CMP platform by increasing the core concentration in each tile to four to enable higher traffic injection into the network for this scalability study. We evaluated NoC throughput, average packet latency, and EDP for all photonic NoCs. In addition to *UltraNoC*-8 and*UltaNoC*-16, we also

considered *UltraNoC*-32, a variant of our architecture with 32 waveguide groups. Figures 5(a)-(c) show the results of these experiments. From the throughput results it can also be inferred that on average *UltraNoC*-8 has 3×, 2.8× and 2.6×, *UltraNoC*-16 has 5.7×, 5.3× and 5.7×, and *UltraNoC*-32 has 9.8×, 9× and 9.7× greater throughput compared to EMesh, Corona and Flexishare. The improvements in throughput for *UltraNoC* over Flexishare and Corona are even better for the 256-core CMP, than in the 64-core CMP case. From figure 5(b), it can be seen that on an average *UltraNoC*-8 has 23%, 25% and 48%, *UltraNoC*-16 has 29%, 30% and 53% and *UltraNoC*-32 has 34%, 36% and 55% lower latency compared to Corona, Flexishare and EMesh architectures respectively. Finally, Figure 5(c) shows that on average *UltraNoC*-8 has 88%, 87% and 20% lower EDP compared to EMesh, Corona and Flexishare respectively. Further *UltraNoC*-16 has 77% and 73% and *UltraNoC*-32 has 55% and 51% lower energy delay product compared to EMesh and Corona respectively. The lower EDP of Flexishare compared to *UltraNoC*-16 and *UltraNoC*-32 is due to the lower photonic hardware used in Flexishare, which lowers its power footprint, but also reduces its throughput and increases its latency.

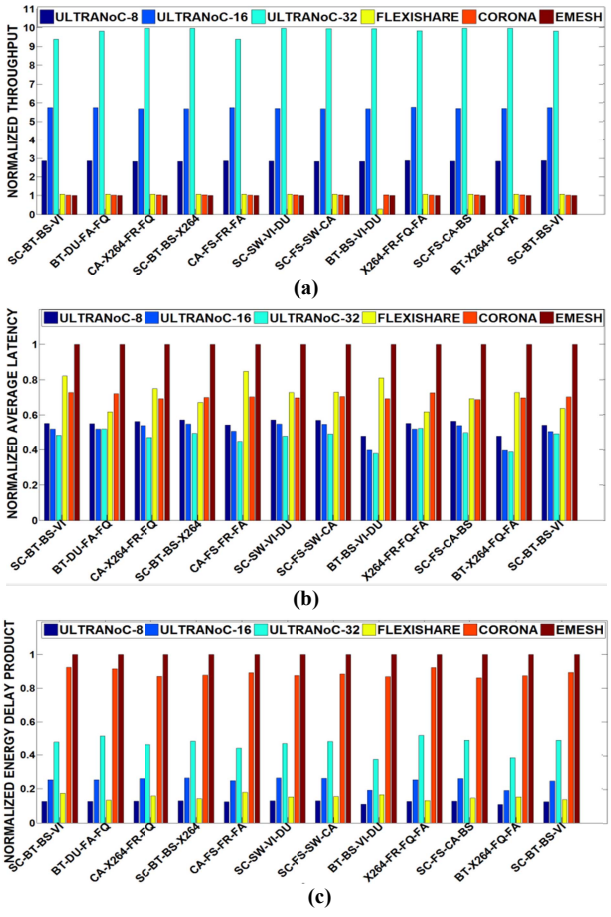

Figure 5: (a) throughput (b) latency (c) EDP comparison of UltraNoC-8, UltraNoC-16 and UltraNoC-32 with other architectures for a 256-core CMP. Results are normalized wrt EMesh.

From the above results, we can summarize that there are variants of *UltraNoC* that can achieve better performance with less hardware compared to existing photonic NoCs, for CMP platforms with low as well as high core counts. The results are a good indicator of the scalability of our proposed *UltraNoC* architecture.

5. CONCLUSIONS

In this work we presented the *UltraNoC* photonic NoC architecture that features improved channel sharing among cores by using an aggressive concurrent token stream-based arbitration strategy. *UltraNoC* also supports the ability to dynamically transfer bandwidth between clusters of cores and re-prioritize multiple co-running applications to further improve channel utilization and adapt to time-varying application performance goals. *UltraNoC* improves throughput by up to 9.8×, latency by up to 55% and EDP by up to 90% over traditional electrical and state-of-the-art photonic NoC architectures with the best-known arbitration mechanisms. *UltraNoC* also scales well with increasing core counts on a chip.

6. ACKNOWLEDGMENTS

This research is supported by grants from SRC, NSF (CCF-1252500, CCF-1302693), and AFOSR (FA9550-13-1-0110).

REFERENCES

[1] R. Ho, et al., “The future of wires,” in *Proc. IEEE*, Apr 2001.
[2] W. J. Dally, B. Towles, “Route packets, not wires,” in *DAC*, 2001.
[3] S. Bahirat et al., "OPAL: A Multi-Layer Hybrid Photonic NoC for 3D ICs," in *ASPDAC*, Yokohama, Japan, Jan 2011.
[4] D. A. B. Miller, “Device requirements for optical interconnects to silicon chips,” *in IEEE, Special Issue on Silicon Photonics*, 2009.
[5] Q. Xu et al., “12.5gbit/s carrier-injection-based silicon micro-ring silicon modulators,” in *Optics Express*, vol. 15,no. 2, Jan. 2007.
[6] S. J. Koester et al., “Ge-on soi-detector/si-cmos-amplifier receivers for high-performance optical communication applications,” in *JLT*, vol. 25,no. 1, pp. 46–57, Jan. 2007.
[7] J. D. Owens, et al., “Research challenges for on-chip interconnection networks,” in *IEEE Micro*, September-October 2007.
[8] D. Vantrease et al., “Corona: System implications of emerging nanophotonic technology,” in *ISCA*, 2008.
[9] Y. Pan et al.,“Firefly: Illuminating future network-on-chip with nanophotonics,” in *ISCA*, 2009.
[10] Y. Pan, J. Kim, G. Memik, “Flexishare: Channel sharing for an energy efficient nanophotonic crossbar,” in *HPCA*, 2010.
[11] C. Chen, A. Joshi, “Runtime management of laser power in silicon-photonic multibus NoC architecture,” in *IEEE JQE*, 2013.
[12] V. Almeida et al., “All-optical switching on a silicon chip” in *Optics Letters*, 29:2867–2869, 2004.
[13] C. A. Barrios, et al., “Low-power-consumption short-length and high-modulation-depth silicon electro-optic modulator” in JLT, 2003.
[14] R. Morris et al., “Power-efficient and high-performance multilevel hybrid nanophotonic interconnect for multicores,” in *NOCS* 2010.
[15] J. Psota et al., “ATAC: On-chip optical networks for multicore processors,” *Boston Area Archit. Workshop*, pp. 107– 108, Jan. 2007.
[16] C. Bienia et al., "The PARSEC Benchmark Suit: Characterization and Architectural Implications," *in PACT, Oct. 2008.*
[17] M. J. Cianchettiet et al., “Phastlane: a rapid transit optical routing network,” in *ISCA,* 2009.
[18] N. Kirman et al., “A power-efficient all-optical on-chip interconnect using wavelength-based oblivious routing,” in *ASPLOS* 2010.
[19] X. Zheng et al., “Ultra-efficient 10Gb/s hybrid integrated silicon photonic transmitter and receiver,” in *Opt. Express*, Mar 2011.
[20] M. Petracca, et al., “Design exploration of optical interconnection networks for chip multiprocessors,” in *HOTI*, Aug. 2008,
[21] A. Joshi et al., “Silicon-Photonic Clos Networks for Global On-Chip Communication”, *in NOCS 2009*
[22] D. Vantrease et al., “Light speed arbitration and flow control for nanophotonic interconnects,” in *IEEE/ACM MICRO*, Dec. 2009.
[23] P. Grani and S. Bartolini, “Design Options for Optical Ring Interconnect in Future Client Devices,” in *ACM JETC, May, 2014.*
[24] T.Pimpalkhute et al., “An application-aware heterogeneous prioritization framework for NoC based chip multiprocessors,” in *ISQED 2014.*
[25] N. Binkert et al.,"The gem5 Simulator," in *CA News*, May 2011.