BiGNoC: Accelerating Big Data Computing with Application-Specific Photonic Network-on-Chip Architectures

Sai Vineel Reddy Chittamuru, *Student Member, IEEE*, Dharanidhar Dang, *Student Member, IEEE*, Sudeep Pasricha, *Senior Member, IEEE*, and Rabi Mahapatra, *Senior Member, IEEE*

Abstract—In the era of big data, high performance data analytics applications are frequently executed on large-scale cluster architectures to accomplish massive data-parallel computations. Often, these applications involve iterative machine learning algorithms to extract information and make predictions from large data sets. Multicast data dissemination is one of the major performance bottlenecks for such data analytics applications in cluster computing, as terabytes of data need to be distributed frequently from a single data source to hundreds of computing nodes. To overcome this bottleneck for big data applications, we propose *BiGNoC*, a manycore chip platform with a novel application-specific photonic network-on-chip (PNoC) fabric. *BiGNoC* is designed for big data computing and exploits multicasting in photonic waveguides. For high performance data analytics applications, *BiGNoC* improves throughput by up to $9.9\times$ while reducing latency by up to 88 percent and energy-per-bit by up to 98 percent over two state-of-the-art PNoC architectures as well as a broadcast-optimized electrical mesh NoC architecture, and a traditional electrical mesh NoC architecture.

Index Terms—Photonic networks-on-chip, photonic channel sharing, big data computing, multicasting

1 Introduction

Large-scale data analytics applications represent some of the most data-intensive workloads in the emerging domain of big data computing. Most of the high-performance data analytics applications e.g., cancer genome analysis, stock market predictions, consumer product recommendations, disaster forecasting, etc. involve iterative execution of various machine learning algorithms. These iterative machine learning algorithms for large-scale data analytics tasks often run on a MapReduce framework [1] implemented either in the cloud or on commodity clusters in datacenters.

Recently, Hadoop [2] and Spark [3] based distributed frameworks are being increasingly used for MapReduce implementations on cloud services. However, wide-spread security exploits and higher off-loading time with cloud computing have driven several organizations to build their own datacenters for big data processing [2], [3]. Such datacenters are safer from intrusion with lower off-loading time, but according to the Hamilton's cost model [4] the overheads due to power dissipation, power distribution, and cooling in such datacenters with commodity processors can be quite significant. A specialized manycore processor solution in which a large number of cores are interconnected through an efficient on-chip network can reduce such overheads and lead to improved system performance, comparted to commodity processors. *This motivates us to design a customized chip manycore processor (CMP) platform to more efficiently run the iterative machine learning algorithms for big data processing.*

The iterative algorithms in big data processing with MapReduce execute on multiple master and servant cores and take thousands of iterations to produce the desired output. Each iteration typically consists of three phases [5] (Fig. 1). In the initial multicast phase (Fig. 1a) a master node (MN), which consists of one or more master cores, multicasts a large feature set of model parameters to one or more servant nodes (SN; each with one or more servant cores) that perform computations based on the parameters. While computing, these servant nodes may need to exchange or shuffle data with other servant nodes. This phase is called the shuffle phase (Fig. 1b). Lastly, in the aggregation phase (Fig. 1c), all the servant nodes update and send their partial results to the master node. The master node aggregates this partial data to produce the multicasting data for the next iteration.

Multicasting is a performance bottleneck in executing large scale data analytics applications that have large fan-out, big data sizes, and take a large number of iterations to achieve convergence. For example, the K-nearest neighbor algorithm for breast cancer prediction and prognosis [6] requires multicasting of approximately 200 MB of sampled cancer genomic features in each iteration, from 100 image

- *S. V. Reddy Chittamuru and S. Pasricha are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, Colorado 80523. E-mail: {sai.chittamuru, sudeep}@colostate.edu.*
- *D. Dang and R. Mahapatra are with the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas 77843. E-mail: {d.dharanidhar, rabi}@tamu.edu.*

Manuscript received 26 Oct. 2017; revised 24 Apr. 2018; accepted 26 Apr. 2018. Date of publication 8 May 2018; date of current version 10 Oct. 2018.
(Coressponding author: Sai Vineel Reddy Chittamuru).
Recommended for acceptance by X. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2833876

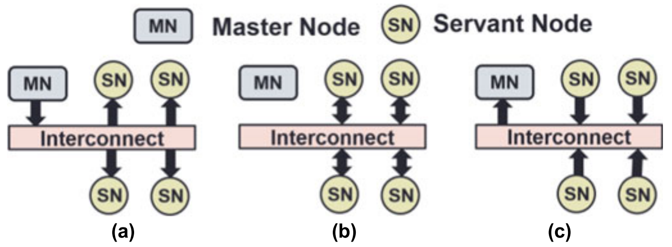


Fig. 1. MapReduce (a) multicast phase, (b) shuffle phase, and (c) aggregation phase of communication while executing iterative machine learning algorithms for large-scale data analytics applications.

samples, each of size 2 MB. As the typical number of iterations is more than 1000, the total multicasting data is in the order of hundreds of gigabytes. Another example is the alternating least squares algorithm for Netflix movie rating prediction, which involves 385 MB of data being distributed to servant nodes per iteration, over hundreds of iterations [7]. This computation thus involves tens of gigabytes of multicast data. *These examples motivate the need for supporting efficient multicasting for big data workload execution scenarios.*

Recent developments in the fabrication of CMOS-compatible on-chip photonic interconnects have opened up the possibility of redesigning emerging manycore processing architectures, especially for big data applications. On-chip photonic interconnects provide several prolific advantages over their conventional metallic counterparts, including the ability to communicate at near light speed, larger bandwidth density by using dense wavelength division multiplexing (DWDM), and lower power dissipation [8]. These advantages motivate us to consider using photonic links for inter-core communication in CMPs that run the iterative algorithms for big data processing. Further, a few prior works [8], [9], [10] have emphasized the importance of multicasting in photonic waveguides to improve data communication rates, and proposed photonic network-on-chip (PNoC) architectures that enable inter-core communication with multicast-enabled waveguides. The multicasting capability of photonic interconnects further inspires us to use them in CMPs optimized for big data processing.

In this paper, we present a novel application-specific PNoC architecture for manycore chips, called *BiGNoC*, to execute large-scale data analytics applications with high throughput and ultra-low latency. To the best of our knowledge, this is the first work that attempts to design PNoCs to tackle iterative machine learning algorithm based large-scale data analytics applications in CMPs. Our novel contributions are:

- We devise a master-servant cluster based communication fabric (MSNoC) with dedicated channels for master-to-servant and servant-to-master communication;
- We design a hierarchical manycore BiGNoC architecture with multiple MSNoCs to execute any combination of high performance large-scale data analytics applications;
- We evaluate BiGNoC by comparing it with two previously proposed PNoCs, as well as a broadcast optimized electrical mesh NoC, and a traditional electrical mesh NoC for multiple real-world big data applications [12], [13], [14], [15].

2 Background and Related Work

Photonic interconnects utilize several photonic devices such as microring resonators (MRs) as modulators, detectors, and switches; photonic waveguides; splitters, and trans-impedance amplifiers (TIAs). Each MR has a unique resonance wavelength in the utilized DWDM spectrum in a waveguide (typically consisting of 64 or less wavelengths) that it can couple to and work correctly with. This resonant nature of an MR allows it to be use as a filter or a switch. A filter MR is used to filter and drop its resonance wavelength on to a photodetector, whereas a switch MR is used to route the propagation of a resonant wavelength signal between two waveguides. Typically, an MR can electro-optically be driven on and off resonance with its resonance wavelength, which allows the MR to modulate 1s (when off-resonance) and 0s (when on-resonance) on its resonance wavelength. The reader is directed to [16] for more discussion on these devices.

Several PNoC architectures have been proposed to date (e.g., [9], [10], [11], [17], [18], [19]) that use on-chip photonic interconnects with MR modulators to modulate electrical signals at the source node on to photonic signals, which then travel through a photonic waveguide, and arrive at MR detectors at the destination node where the photonic signals are detected and electrical signals recovered. Several efforts have explored high throughput crossbar PNoCs that provide non-blocking connectivity, e.g., [9], [10], [11], [17] using different types of photonic waveguides such as Multiple-Write-Single-Read (MWSR), Single-Write-Multiple-Read (SWMR), and Multiple-Write-Multiple-Read (MWMR). Furthermore, multicasting in photonic waveguides enables simultaneous reception of photonic signals in multiple destination nodes. These destination nodes partially de-tune their MR detectors from their resonating wavelengths [8], such that a portion of the photonic energy in the multicast-enabled waveguide continues to be absorbed in subsequent MR detectors of other destination nodes. In the presence of on-chip multicast traffic (i.e., same message transfer from one source node to multiple destination nodes), these multicast-enabled waveguides enable higher data rates compared to ordinary photonic waveguides that inefficiently transfer a single multicast message as multiple unicast messages. Only a few prior works exploit multicasting in SWMR [9] and MWMR [10] waveguides, primarily to improve the performance of PNoC architectures with cache coherence traffic (e.g., in the MOESI coherence protocol, when a shared block is invalidated, an invalidate message must be multicast to all sharers). In addition, a photonic multistage NoC was proposed in [40] that uses photonic-distributed arbitration and concurrent channel reservation mechanisms. This multistage NoC uses an electrical router to interconnect multiple photonic sub-networks and achieves lower latencies. A high-throughput hybrid photonic mesh-diagonal links topology was proposed in [41] with a contention-aware adaptive routing function, and a parallelized photonic channel allocation protocol, to reduce NoC latency. *However, no prior work has attempted to design PNoCs to optimize iterative machine learning algorithm-based large-scale data analytics applications in CMPs.*

Several architectures have been explored recently to address large-scale data analytics applications. A PENC manycore architecture consisting of 192 small processing cores was proposed in [21], which can work as a co-processor in tandem with a general-purpose CPU to accelerate big data processing. A low-power manycore architecture for a modern big-data stream mining applications is proposed in [22] that is able to cope with the dynamic nature of the input data stream while consuming limited power. A parallel CMP architecture called SpiNNaker based on a customized electrical NoC to implement spiking neural networks was proposed in [23]. The cores in this architecture are connected by a modified version of the torus topology, whereas the interchip topology is a 2D triangular mesh with 6-port routers. A neural network architecture called EMBRACE is proposed in [24] which integrates a 2D array of interconnected neural tiles surrounded by I/O blocks and adopts a hierarchical mesh-based topology to connect neural tiles. Furthermore, it uses a region-based routing scheme in each network layer to direct messages to destination nodes. Some works have demonstrated reconfigurable neural networks on a broadcast-aware mesh NoC architecture [25], [26]. A theoretical analysis for determining a preferred interconnect architecture for general purpose configurable emulation of spiking neural networks is presented in [25] and shows that mesh NoC using multicast is the most suitable architecture for a wide range of neural network topologies. A cluster-based reconfigurable NoC architecture for neural networks is presented in [26], which employs a reconfigurable communication fabric that efficiently handles multicast communication. In [27], a CPU-GPU architecture was presented with an electrical ring network to better execute large-scale data analytics applications, but this ring interconnect is known to be inefficient for large-scale systems. A hybrid (wired+wireless) on-chip interconnect based CPU-GPU architecture was proposed in [28] for large-scale data analytics applications. The authors in [38] propose Melia, which is an FPGA-based MapReduce architecture. *None of the abovementioned prior works explore the impact of using photonic interconnects for big data processing as part of the on-chip network. Our goal in this paper is to show, for the first time, how PNoC architectures can be designed and customized for manycore chips, to meet the unique communication requirements of big data analytics applications.*

3 Master-Servant Cluster Architecture

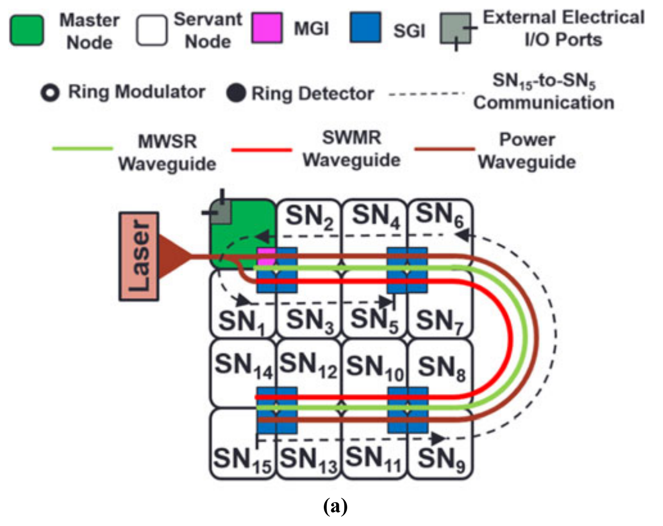
High-performance data analytics applications use a set of iterative machine learning algorithms for data predictions. A machine learning job may take hundreds or thousands of iterations to converge to a solution. On a CMP, each iteration starts with the multicast of a big data set of model parameters from a master core to all the servant cores. Then the servant cores sometimes exchange data among themselves while processing their received data, thus creating inter-servant traffic. Lastly, each servant updates the model parameters partially and sends these model parameters to the master node. These partial results are aggregated at the master node to form the global model parameters for the computations in the subsequent iteration. Thus, execution of large-scale data-intensive applications requires dedicated hardware with master cores, servant cores, and an interconnection fabric between the masters and servants. In this section, we describe the architecture of a new master-servant cluster based communication fabric (*MSNoC*), in which master cores are connected to servant cores via photonic communication channels.

TABLE 1
Micro-Architectural Parameters For MSNoC Cluster

Number of nodes per cluster	16 (1 MN and 15 SNs)
Number of cores	64 (4 per node)
Servant Node (SN):	
Number of servant cores	4
Buffer size of concentrator	10
Servant Core:	
L1 I-Cache size/Associativity	16 KB/Direct Mapped Cache
L1 D-Cache size/Associativity	16 KB/Direct Mapped Cache
Master Node (MN):	
Number of master cores	4
Buffer Size of concentrator	20
Master Core:	
L1 I-Cache size/Associativity	32 KB/Direct Mapped Cache
L1 D-Cache size/Associativity	32 KB/Direct Mapped Cache
L2 Cache size/ Associativity	128 KB/ Direct Mapped Cache
L2 Coherence	MOESI
Frequency	5 GHz
Issue Policy	In-order
Memory controllers	1
Main memory	8 GB; DDR4@30 ns

In our *MSNoC* architecture, a node (N) is defined as an entity consisting of four cores. A node can either be a master node (MN; with four master cores connected using an electrical concentrator) or a servant node (SN; with four servant cores connected using an electrical concentrator). The buffer size of the electrical concentrator of an MN is larger compared to the buffer size of the electrical concentrator of an SN, as the MN node is expected to receive more number of packets compared to a servant node. Our simulation based analysis shows that the buffer size of the electrical concentrator of an MN needs to be larger than the buffer size of the electrical concentrator of an SN. More details about buffer sizes of MNs and SNs are presented in Table 1. Each master core in an MN has a private L1 and L2 cache, whereas each servant core in an SN has only a private L1 cache. The L1 cache size of a master core in the MN is larger than the L1 cache size of a servant core in the SN (see Table 1). As master cores access main memory more frequently compared to servant cores, therefore, the larger L1 size of a master core boosts the *MSNoC*'s performance. Every MN and SN is attached to a gateway interface (GI) module that facilitates transfers between the core-cache layer and the interconnection network layer. A detailed layout of the *MSNoC* is shown in Fig. 2a, where 16 nodes are arranged in a 4×4 grid. Among these 16 nodes, a single node is an MN and the remaining nodes are SNs (i.e., SN_1 to SN_{15}). The master GI (MGI) and servant GI (SGI) are shown in Figs. 2b and 2c, respectively, and discussed further in Sections 3.1, 3.2 and 3.3. Communication between cores within a node (MN or SN) uses a 5×5 on-chip electrical router, where four of its input and output (I/O) ports are connected to four cores (master or servant) and the fifth I/O port is connected to the GI module associated with the node. A round-robin arbitration scheme is used within each node for communication between cores and the GI.

Communication between SNs and MNs is accomplished using SWMR and MWSR waveguides (Sections 3.1, 3.2 and



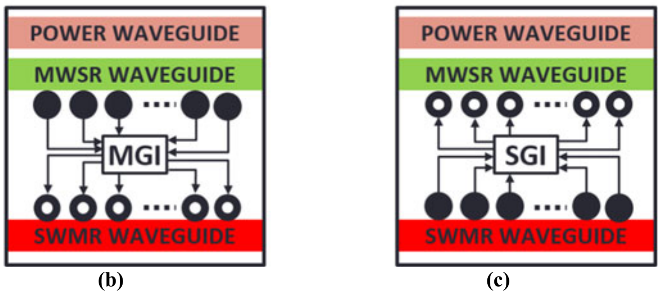


Fig. 2. (a) *MSNoC* layout with SWMR, MWSR, and power waveguides (b) master gateway interface (MGI) (c) servant gateway interface (SGI).

3.3). There is also a power waveguide that runs in parallel with the SWMR and MWSR waveguides. This power waveguide carries all the wavelengths used for data traversal in the waveguides. A 1×2 splitter is used to split power from the power waveguide to SWMR waveguides as shown in Fig. 2a. In addition, a series of 1×2 splitters along the power waveguide are used to supply power to the modulators that are used to write data on to the MWSR waveguides. The splitting losses due to these splitters are considered in the laser power calculations of MSNoC (see Section 6). Our *MSNoC* with a group of 16 nodes (with 64 cores) has dedicated access to main memory via a memory controller at the MN. This is similar to the processor used in Sunway TaihuLight [34], which has dedicated main memory access for every 64 cores. The micro-architectural parameters of nodes and cores in an *MSNoC* cluster are summarized in Table 1. In addition. the functionalities of MNs and SNs are assumed to be correct in the MSNoC. Therefore, this work does not consider the impact of mistakes from MNs and SNs.

In the following three subsections, we present more details about the interconnects that are used to enable communication between the MNs and SNs of an *MSNoC* cluster.

3.1 MN-to-SN Communication in MSNoC Cluster

As discussed earlier, the interconnection network between the master and servant cores plays a crucial role towards achieving faster execution of large-scale data analytics applications on an *MSNoC* cluster. As the communication from master cores to servant cores has significant periods of multicast traffic, this motivates us to use multicast enabled

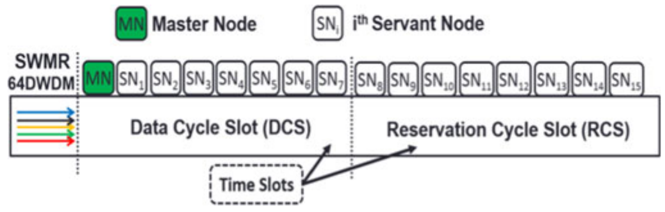

MN Master Node
SNi ith Servant Node
SWMR 64DWDM
MN SN1 SN2 SN3 SN4 SN5 SN6 SN7 SN8 SN9 SN10 SN11 SN12 SN13 SN14 SN15
Data Cycle Slot (DCS)
Reservation Cycle Slot (RCS)
Time Slots

Fig. 3. Distribution of reservation cycle and data cycle slots within SWMR waveguide to enable MN-to-SN communication.

photonic waveguides in our *MSNoC* cluster, to enable faster master-servant communication. As shown in Fig. 2a, in an *MSNoC* cluster we use a multicast enabled Single-Write-Multiple-Read (SWMR) waveguide group to enable communication from a single MN to multiple SNs, where each waveguide group has four SWMR waveguides. The SWMR waveguide group in an *MSNoC* starts from an MN and passes through all of the SNs (i.e., $\mathrm{SN}_1 - \mathrm{SN}_{15}$) in the cluster (Fig. 2a) to enable MN-to-SN communication. An MN has the ability to write on the SWMR waveguide group using its ring modulators (see Fig. 2b, which shows modulators of an MN on SWMR waveguide), and all the SNs are capable of reading from the SWMR waveguide group using their ring detectors (see Fig. 2c, which shows detectors of an SN on SWMR waveguide). To power these SWMR waveguides, we use a broadband off-chip laser source and a 1×4 splitter to split the laser power across the four SWMR waveguides. We also use 64 DWDM wavelengths in each of the four SWMR waveguides of the SWMR waveguide group. Therefore, in an SWMR waveguide group there are 256 modulators and 256 detectors in each MN and SN, respectively.

As all SNs are capable of receiving (reading) from an SWMR waveguide group during MN-to-SN communication, there is a need for receiver selection between SNs to ensure that only the designated receiver will receive data from the shared waveguide group. For receiver selection, each SWMR waveguide group is divided into a fixed number of time slots, based on the time taken by light to traverse the length of the waveguide on a die. Based on the geometric calculations considering a $100\,\mathrm{mm}^2$ chip area for a 64 core CMP at 22 nm technology node, traversal of light through an SWMR waveguide group takes 2 cycles (i.e., 0.4 ns) in an *MSNoC* cluster at 5 GHz clock frequency. Therefore, we divide the SWMR waveguide group into 2 time slots, and each time slot is spread across 8 nodes (the node can either be an MN or SN), as shown in Fig. 3. These time slots are further classified into two types: reservation cycle slots (RCS), and data cycle slots (DCS).

In our reservation assisted MN-to-SN communication process, MNs send data to SNs in two cycles (Fig. 3). In the reservation cycle, the MN reserves the SWMR waveguide group for an SN. Once the reservation is done, the MN sends data to the selected SN in the next cycle (i.e., data cycle). To perform the reservation, the MN uses the first SWMR waveguide in the SWMR waveguide group (this waveguide is shown in Fig. 3). The remaining three SWMR waveguides in the SWMR waveguide group are used only in the data cycle to transfer data. Each $\mathrm{SN_i}$ is assigned a receiver selection wavelength λ_i, that is available in the first SWMR waveguide of the SWMR waveguide group. For example, in an MSNoC cluster with 16 nodes (see Fig 2a),

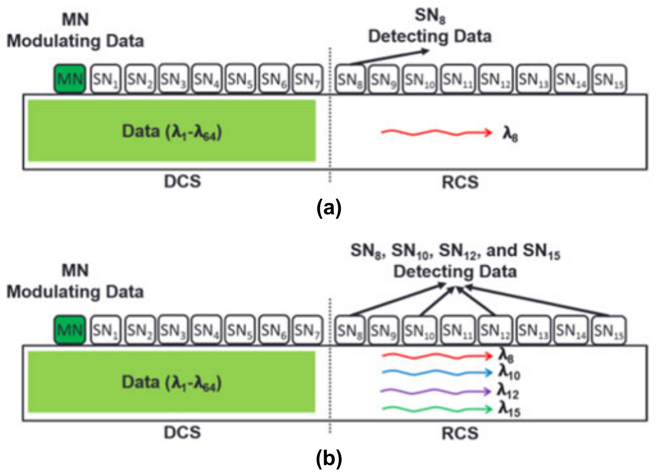


Fig. 4. (a) Transmission of unicast data from an MN to SN_8 in *MSNoC*, which shows receiver selection wavelength λ_8 in RCS of the SWMR waveguide; (b) Multicast of data from an MN to multiple SNs $\mathrm{SN}_8, \mathrm{SN}_{10}, \mathrm{SN}_{12}$, and SN_{15} in *MSNoC*, which shows respective receiver selection wavelengths $\lambda_8, \lambda_{10}, \lambda_{12}$, and λ_{15} in RCS of the SWMR waveguide.

$\lambda_1 - \lambda_{15}$ are assigned as receiver selection wavelengths to $\mathrm{SN}_1 - \mathrm{SN}_{15}$, respectively. When an MN wants to send data to an SN, it gets access to the next RCS, which initially has all of the receiver selection wavelengths from the power waveguide. In this RCS, the MN uses its modulator bank to remove all of the receiver selection wavelengths except the one corresponding to the SN of interest. Subsequently, in the next DCS, the MN modulates data on the 256 wavelengths in four SWMR waveguides (as each SWMR waveguide uses 64 DWDM wavelengths $(\lambda_1 - \lambda_{64})$) of each SWMR waveguide group assigned for data transfer. *Therefore, our receiver selection mechanism prudently reuses the same set of wavelengths in the first SWMR waveguide of an SWMR waveguide group for reservation and data transmission.* On the receiving side of the SWMR waveguide group, whenever an RCS reaches an SN_i, it only switches on the detector which corresponds to its receiver selection wavelength λ_i located on the first SWMR waveguide of the SWMR waveguide group. Whenever an SN_i detects its receiver selection wavelength in the RCS, it switches on its remaining detectors not only on the first SWMR waveguide but also on the remaining three SWMR waveguides of the SWMR waveguide group to receive data in the next DCS.

We illustrate this sending and receiving process with a simple example. In Fig. 4a, suppose an MN needs to send data to SN_8 that has a corresponding receiver selection wavelength λ_8. The MN modulates in the next RCS, such that only λ_8 (the dedicated wavelength for receiver selection of SN_8) is made available by removing all of the wavelengths except λ_8 (using its modulators) in the first SWMR waveguide of the SWMR waveguide group. On the receiving end, all of the SNs which are in the RCS switch-on their detectors for the corresponding receiver selection wavelengths (e.g., nodes SN_8 to SN_{15} switch-on detectors with resonance wavelengths λ_8 to λ_{15}, respectively) in the first SWMR waveguide of the SWMR waveguide group. Therefore, at SN_8 only the detector for wavelength λ_8 is switched on in the RCS. Once λ_8 is detected, SN_8 prepares to receive data in the next DCS by switching on the remaining detectors not only on the first SWMR waveguide but also on the remaining three SWMR waveguides in the SWMR waveguide group in that node.

The receiver selection mechanism presented above can only transmit unicast messages, but while executing big data applications the MN will send not only unicast messages to a single SN but also multicast messages to multiple SNs. One possible solution is to translate these multicast messages into several unicast messages and send them to their respective SNs. But this can cause network congestion and reduce network performance [30]. Therefore, for MN to multiple SN communication in an *MSNoC*, we avoid such repeated unicast messages by providing multicasting support in the *MSNoC's* SWMR waveguides.

Unlike Corona [10] and Firefly [9] PNoCs, where all multicast messages are broadcast and transmitted to all nodes in the network, *MSNoC* enables multicasting to specific nodes in the network. This is realized as follows: the MN in an *MSNoC* releases multiple receiver selection wavelengths into the first SWMR waveguide of the SWMR waveguide group (see Fig. 4b) corresponding to multiple SNs in the next RCS. In the immediately following DCS, the MN modulates the data which needs to be multicast to different SNs on to four SWMR waveguides within the SWMR waveguide group. To enable photonic multicast of data in SWMR waveguides, we partially de-tune the ring detectors from their resonating wavelengths [8], such that a portion of the photonic energy in the SWMR waveguide group continues to be absorbed in subsequent ring detectors. Multicasting thus requires higher laser power compared to unicasting so as to maintain sufficient photonic signal intensity for detection in the worst case, i.e., for the detectors of the last receiving node which receives the multicast data.

Interestingly, the laser power injected in the SWMR waveguide group for multicasting in an *MSNoC* does not change with the number of nodes that need to receive the multicast message. We designed the laser source for the worst-case power loss, which occurs when all of the SNs receive a multicast message (i.e., broadcast message) from an MN. We have considered this extra laser power overhead when presenting energy-delay-product and energy-per-bit results for the *MSNoC* cluster in our experimental results section. In this work, we do not consider optimizing laser power through a laser power management scheme. However, it is possible to integrate previously proposed laser power management schemes [31], [32], as these works are orthogonal to our work.

Figs. 4a and 4b illustrate the difference between transmission of unicast and multicast messages in our *MSNoC* cluster. Suppose an MN needs to multicast data to $\mathrm{SN}_8, \mathrm{SN}_{10}, \mathrm{SN}_{12}$, and SN_{15} whose corresponding receiver selection wavelengths are $\lambda_8, \lambda_{10}, \lambda_{12}$, and λ_{15}, respectively. The MN modulates in the next RCS, such that only $\lambda_8, \lambda_{10}, \lambda_{12}$, and λ_{15} are made available by removing all the wavelengths except $\lambda_8, \lambda_{10}, \lambda_{12}$, and λ_{15} (using the MN's modulators; Fig. 4b) from the first SWMR waveguide of SWMR waveguide group. At the receiver end at $\mathrm{SN}_8, \mathrm{SN}_{10}, \mathrm{SN}_{12}$, and SN_{15}, the detectors for wavelengths $\lambda_8, \lambda_{10}, \lambda_{12}$, and λ_{15} respectively on the first SWMR waveguide of the SWMR waveguide group are switched on when these SNs are in the RCS. At SN_8, once λ_8 is detected in the receiver selection slot, the node prepares to receive data from all of the four SWMR waveguides within the SWMR waveguide group in the next DCS by partially detuning the ring detectors (partial detuning of ring resonators is employed to receive both unicast and multicast data in SN_8) from their corresponding resonating wavelengths in that node. The partial de-tuning of ring detectors of SN_8 will remove a portion of light available in the SWMR waveguide, leaving the remaining portion of light for the other detectors to absorb. Similarly, on detection of $\lambda_{10}, \lambda_{12}$, and λ_{15}, nodes SN_{10}, SN_{12}, and SN_{15} respectively prepare to receive data in the next DCS. Note that our architecture does not differentiate between unicast and multicast transmissions, as it always employs partial detuning to receive both unicast and multicast messages.

3.2 SN-to-MN Communication in MSNoC Cluster

All the SNs send data back to an MN in the aggregation phase, for which our *MSNoC* uses a Multiple-Write-Single-Read (MWSR) waveguide group for SN-to-MN communication, with each waveguide group having four MWSR waveguides. As shown in Fig. 2a, this MWSR waveguide group starts from the last SN (i.e., SN_{15}) and traverses all of the remaining SNs (i.e., $\mathrm{SN}_1 - \mathrm{SN}_{14}$) and finally terminates at the MN. In contrast to the SWMR waveguide group, all SNs have the ability to write on the MWSR waveguide group using their ring modulators (see Fig. 2c which shows modulators of an SN on an MWSR waveguide) and the MN has the ability to read from the MWSR waveguide group using its ring detectors (see Fig. 2b which shows detectors of an MN on an MWSR waveguide).

As all SNs are capable of modulating (writing) in an MWSR waveguide group, there is a need for arbitration between SNs to ensure that the data from different SNs does not destructively overlap on the shared MWSR waveguide group. We use a centralized electrical arbiter to avoid contention between SNs when writing to an MWSR waveguide group. This arbiter uses a round-robin arbitration scheme. However, by virtue of being a centralized arbiter, it lacks scalability beyond a certain cluster size. We address this drawback of the centralized arbiter in Section 5. Furthermore, MSNoC exploits the centralized arbiter to enable flow control in the SN-to-MN communication. We employ an Xon/Xoff flow control mechanism to control packet flow from an SN to MN. Whenever, the receiving buffer in the MN is full then a signal is sent to the centralized arbiter, such that this arbiter stops assigning MWSR waveguide groups to the SNs. Otherwise, if the buffer is not full then the centralized arbiter allocates MWSR waveguide groups to SNs to transmit packets to MNs. As per the explanation provide in Section 3, a power waveguide (see Fig. 2a) that runs in parallel with the MWSR waveguide group uses a series of splitters to supply photonic signals to the ring modulators to write data on to the MWSR waveguide group. As each of four MWSR waveguides within this MWSR waveguide group carries 64 wavelengths, therefore, each MWSR waveguide group requires 256 modulators and 256 detectors in the SN and MN to write and read data, respectively. The total amount of photonic hardware required for the MSNoC architecture is quantified in Section 6.

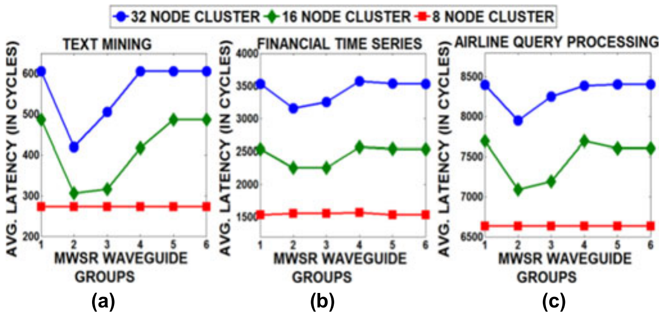


Fig. 5. Variation of average packet latency in *MSNoC* cluster with (a) 32 nodes (b) 16 nodes, and (c) 8 nodes having different MWSR waveguide groups (each group has 4 waveguides) across three big data applications.

3.3 SN-to-SN Communication in MSNoC Cluster

SN-to-SN communication occurs in the *MSNoC* when the execution of high-performance data analytics applications is in the 'shuffle' phase. Our *MSNoC* enables SN-to-SN communication via the MN. We illustrate this SN-to-SN communication with a simple example. When SN_{15} wants to send data to SN_5, first SN_{15} sends data to the MN using an MWSR waveguide group, and then the MN sends the received data to SN_5 using an SWMR waveguide group. We show the $\mathrm{SN}_{15} - \mathrm{to} - \mathrm{SN}_5$ communication path in Fig. 2a as a dotted line. This process thus involves two O/E (optical to electrical) and two E/O (electrical to optical) conversions for each SN-to-SN transfer. The next section presents a performance analysis for an *MSNoC* cluster with different SN counts. In Section 5, we describe how multiple *MSNoC* clusters are combined to form the *BiGNoC* architecture.

4 MSNoC: Sensitivity Analysis

In an *MSNoC* cluster, with the increase in number of SNs, contention between SNs to access an MWSR waveguide group increases. One possible solution to reduce this contention is to increase the number of MWSR waveguide groups in the *MSNoC* cluster. To understand the impact of this change, we performed a sensitivity analysis by varying the number of MWSR waveguide groups within an *MSNoC*, for different cluster sizes (8, 16, 32 nodes; each cluster has 1 MN and the remainder of the nodes are SNs). We modeled and simulated these variants of *MSNoC* at a cycle-accurate granularity with a SystemC-based NoC simulator. We considered three applications: Text Mining [12], Financial Time Series [13], and Airline Query Processing [14]. The goal with these workloads was to emulate an environment with different intensities of MN-to-SN, SN-to-MN, and SN-to-SN traffic with diverse bandwidth needs.

Figs. 5a, 5b, 5c show the variation of average packet latency with increase in number of MWSR waveguide groups (x-axis) for the three sizes of the *MSNoC* cluster, across the three big data applications. It can be observed that for a specific MWSR waveguide group count within an *MSNoC*, increase in cluster size (i.e., increase in node count) increases the average packet latency for all big data applications. Increase in number of nodes within a cluster increases contention between SNs to access the MWSR waveguide groups while sending data to an MN, which increases packet wait time in the buffers of SNs and ultimately increases overall packet latency. From Figs. 5a, 5b, 5c, it can also be seen that with the increase in MWSR waveguide groups, the average packet latency first decreases until the waveguide group count reaches two. When MWSR waveguide group count is increased beyond two, the latency starts increasing. Intuitively, increase in number of MWSR waveguide groups from one to two increases the SN-to-MN data rate (as two MWSR waveguide groups enable two packets to be sent simultaneously from two SNs to an MN), which decreases packet waiting time in the buffers of SNs and reduces the average packet latency. Despite the increase in data rate from SN-to-MN, with the increase in number of MWSR waveguide groups beyond two, there is saturation in the data channel to the MN (as this data channel is capable of sending only one packet per cycle from the concentrator to a master core). This increases the waiting time of packets at the receiving buffers of MGIs and increases average packet latency across all the big data applications.

Based on the analysis presented above, we optimally select two MWSR waveguide groups for *MSNoCs* with cluster sizes of 32 and 16 nodes. Additionally, from the Figs. 5a, 5b, 5c it can also be seen that average latency for an *MSNoC* with 8 nodes remains constant for all MWSR waveguide group counts across all the benchmark applications. From this result, it can be concluded that in an *MSNoC* with 8 nodes, a single MWSR waveguide group is sufficient and optimal for SN-to-MN communication. Furthermore, for these optimal MWSR waveguide group counts used within MSNoC clusters the buffers in concentrators of MNs and SNs will seldom become performance bottlenecks. We use these optimally determined MWSR waveguide group counts for different cluster sizes in our homogeneous and heterogeneous master-servant multi-cluster architecture (*BiGNoC*) which we describe in detail in the next section.

5 BiGNoC Architecture

5.1 Homogeneous BiGNoC Architecture

In Section 3, we presented an *MSNoC* architecture that aims to effectively connect an MN and many SNs within a master-servant cluster using MWSR and SWMR waveguide groups. Typically, large-scale data analytics applications require a greater number of servant cores than can be accommodated in a single *MSNoC* cluster. There are two ways to address the requirement for additional servant cores: increase the cluster size or use multiple interconnected clusters. We prefer the latter solution as increase in cluster size leads to: (i) increase in power dissipation of the SWMR and MWSR waveguide groups (see Table 3 later in the paper), (ii) increase in average packet latency (see Fig. 5), and (iii) increase in MWSR waveguide group arbiter complexity. These drawbacks suppress the power and performance benefits of photonic interconnects. Moreover, increase in cluster size limits the number of available masters within a cluster as the *MSNoC* is designed to have only one master node. Therefore, we propose a homogeneous multi-cluster architecture (*BiGNoC-HOM*) with four uniform clusters represented as C_0, C_1, C_2, and C_3, as shown in Fig. 6a, where each cluster has 16 nodes (i.e., 64 cores).

Each 16-node cluster in the *BiGNoC-HOM* architecture uses one SWMR waveguide group for MN-to-SN communication. As explained in Section 3.1, each SWMR waveguide group is divided into two time slots to enable receiver selection. Furthermore, based on the sensitivity analysis presented in the previous section, we optimally select two MWSR waveguide groups in each cluster for SN-to-MN communication. This architecture considers a single broadband laser source to power all of its SWMR and MWSR waveguides and uses 64 wavelengths in each waveguide for data communication. We add three more splitters to the power waveguide, to distribute laser power to the SWMR and MWSR waveguide groups of the four clusters in *BiGNoC-HOM*.

Each MN has a memory controller to send and receive data from off-chip main memory with dedicated channels for communication. Therefore, *BigNoC-HOM* uses four memory controllers, where each is associated with an MN within a cluster. In addition, as shown in Fig. 6, all the four MNs within the four clusters of *BiGNoC-HOM* are connected to a single 4×4 electrical router using their external electrical I/O ports (shown at the top left of Fig. 2a). This electrical router is used for inter-cluster communication. We have considered a four-stage pipelined electrical router with 4 I/O ports that are connected to four MNs with the following pipeline stages: buffer write/route computation, region validation/switch allocation, switch traversal, and link traversal. This router has an input and output queued crossbar and uses double buffering with an 8-flit buffer size to more effectively cope with the higher photonic path throughput. Each master node is provisioned with an additional buffer which receives and stores packets from other clusters.

Intuitively, inter-cluster MN-to-MN communication occurs in one hop through the electrical router. Inter-cluster MN-to-SN and SN-to-MN communication require two hops: inter-cluster MN-to-SN communication requires MN-to-MN (inter-cluster) and MN-to-SN (intra-cluster) hops, whereas inter-cluster SN-to-MN communication requires SN-to-MN (intra-cluster) and MN-to-MN (inter-cluster) hops. Further, inter-cluster SN-to-SN communication requires three hops: SN-to-MN (intra-cluster), MN-to-MN (inter-cluster), and MN-to-SN (intra-cluster). We illustrate the SN-to-SN communication across different clusters with a simple example. If node N_2 (i.e., SN) of C_0 needs to send a packet to node N_{10} (i.e., SN) of cluster C_1, then N_2 of C_0 first sends data to N_0 (i.e., MN) of C_0 using an MWSR waveguide group. Then from this node the packet is sent to N_0 (i.e., MN) of C_1 through the electrical router that enables inter-cluster communication. Lastly, the packet is sent to N_{10} of C_1 using the SWMR waveguide group in that cluster. Thus, inter-cluster SN-to-SN communication incurs minimal overhead with only two O/E and two E/O conversions, which is similar to intra-cluster SN-to-SN communication.

5.2 Heterogeneous BiGNoC Architecture

As explained in the previous subsection, *BiGNoC-HOM* with four uniform clusters can enable inter-cluster communication between MNs and SNs. While executing applications with larger servant core count requirements, *BiGNoC-HOM* incurs higher inter-cluster traffic. This increase in inter-cluster traffic via slower electrical links may reduce the performance of the proposed *BiGNoC-HOM* architecture. This motivates us to design a heterogeneous version of

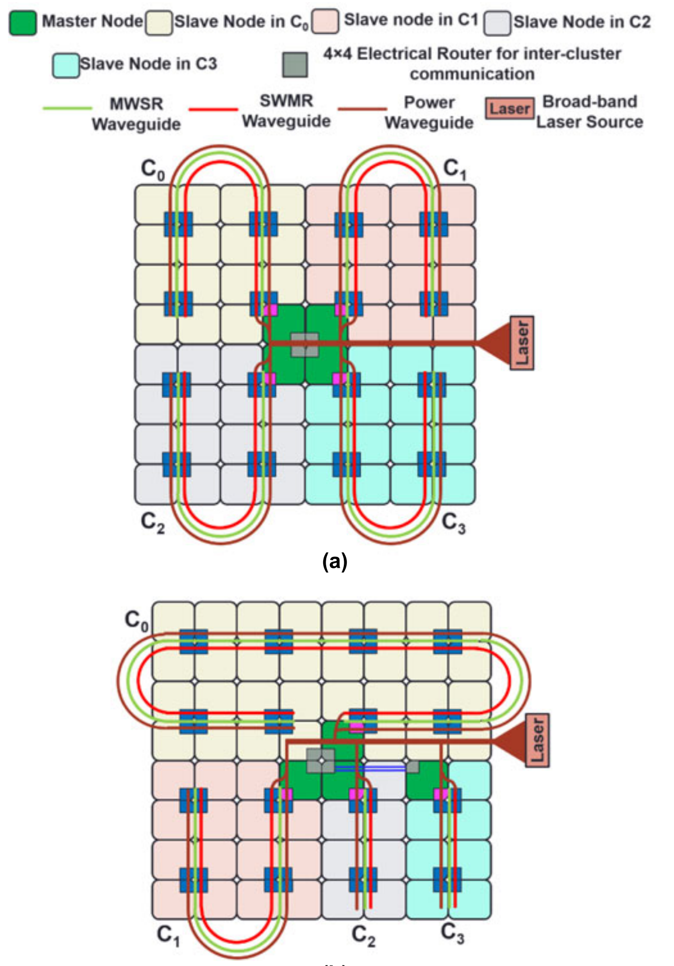


Fig. 6. (a) Homogeneous *BiGNoC* with four uniform clusters C_0, C_1, C_2, C_3, with each cluster having 16 nodes, (b) Heterogeneous *BiGNoC* with four clusters C_0, C_1, C_2, and C_3 having 32, 16, 8, and 8 nodes, respectively.

BiGNoC (*BiGNoC-HET*) with four clusters, but with different cluster sizes.

A larger cluster size with more number of SNs in BiGNoC always improves the performance of a big data application requiring higher number of servant cores. This is because a larger cluster size reduces the inter-cluster traffic through slower electrical links of the 4×4 electrical router. But SWMR waveguides within a BiGNoC cluster cannot support multicasting beyond 32-nodes, due to the limitations in receiver sensitive and TIA circuit bandwidth [8]. Therefore, we have considered a cluster with a maximum of 32 nodes (with 128 cores) for *BiGNoC-HET*. Furthermore, after analyzing the master and servant core requirements of big data benchmark applications, we concluded that these applications have different scales which require different cluster sizes. Therefore, to execute medium and small applications, we have considered a 16-node cluster with 64 cores and an 8-node cluster with 32 cores, respectively. Therefore, in *BiGNoC-HET*, we use clusters C0, C1, C2, and C3 with 32, 16, 8, and 8 nodes, respectively, as shown Fig. 6b. To enable receiver selection in SWMR waveguide groups of these clusters, we divided the waveguides in clusters C_0, C_1, C_2, and C_3 into 4, 2, 1, and 1 time slots respectively, based on the time taken by light to traverse these waveguides on a die. Based on the sensitivity analysis presented in Section 4, we use 2, 2, 1, and 1 MWSR waveguide groups for clusters C_0, C_1, C_2, and C_3 respectively. Similar to *BiGNoC-HOM*, we use four memory controllers to control off-chip memory and an electrical router to connect all four clusters of *BiGNoC-HET*.

In *BiGNoC* (especially *BiGNoC-HET*), scheduling of applications plays a crucial role in enhancing overall performance. For example, *BiGNoC-HET* can achieve better performance when an application with a greater servant core requirement is scheduled to a cluster with more servant cores. In contrast, scheduling a larger application on multiple smaller clusters will increase inter-cluster communication, which in turn may degrade performance. This motivates us to design an application scheduling algorithm for *BiGNoC* which is presented in the next subsection. We perform a detailed comparative study between *BiGNoC-HOM* and *BiGNoC-HET* in Section 6.3.

Algorithm 1. Application scheduling in *BiGNoC*

Inputs: Applications (AP_i) **with master cores** (MA_i) **and servant cores** (SA_i) **requirements, and *BiGNoC* with clusters** (C_j)**, master cores** (MC_j)**, and servant cores** (SC_j)

```
1:  Sort AP_i (highest SA to lowest SA)
2:  Sort BigNoC clusters (highest SC to lowest SC)
3:  for all i do $NSA_i = SA_i; NMA_i = MA_i$;
4:  for all j do $FSC_j = SC_j; FMC_j = MC_j$;
5:  for each $AP_i$ do
6:    for each $C_j$ do
7:      if $FSC_j > 0$ then // Checks for free cores in clusters
8:        if $FSC_j - NSA_i \geq 0$ then
9:          Do_ Scheduling ($AP_i \rightarrow NSA_i$ servant cores of $C_j$) //Map servants
10:         $FSC_j = FSC_j - NSA_i; NSA_i = 0$;
11:         if $FMC_j > 0$ and $FMC_j - NMA_i \geq 0$ then
12:           Do_Scheduling ($AP_i \rightarrow NMA_i$ master cores of $C_j$)//Map masters
13:           $FMC_j = FMC_j - NMA_i; NMA_i = 0$;
14:         else if $FMC_j > 0$ and $FMC_j - NMA_i < 0$ then
15:           Do_ Scheduling ($AP_i \rightarrow (NMA_i - FMC_j)$ master cores of $C_j$)
16:           $NMA_i = NMA_i - FMC_j; FMC_j = 0$;
17:       else
18:         Do_ Scheduling ($AP_i \rightarrow (NSA_i - FSC_j)$ servant cores of $C_j$)
19:         $NSA_i = NSA_i - FSC_j; FSC_j = 0$;
20:         if $FMC_j > 0$ and $FMC_j - NMA_i \geq 0$ then
21:           Do_ Scheduling ($AP_i \rightarrow NMA_i$ master cores of $C_j$)
22:           $FMC_j = FMC_j - NMA_i; NMA_i = 0$;
23:         else if $FMC_j > 0$ and $FMC_j - NMA_i < 0$ then
24:           Do_ Scheduling ($AP_i \rightarrow (NMA_i - FMC_j)$ master cores of $C_j$)
25:           $NMA_i = NMA_i - FMC_j; FMC_j = 0$;
```

Output: Scheduled master-servant cores of app onto clusters of *BiGNoC*

5.3 Application Scheduling in BiGNoC

Algorithm 1 shows the pseudo-code for the application scheduling procedure in *BiGNoC*. Our scheduling algorithm will schedule a combination of applications if the total number of master and servant cores within a BigNoC-HET architecture

TABLE 2
Big Data Application Benchmarks, With Three Variants Each, Based on Their Master-Servant Requirements [7], [12], [13], [14], [15]

Application	Representation	Application variants
Netflix Movie Rating	N (Masters-Servants)	N (1-50), N (1-70), N (1-100)
Text Mining	T (Masters-Servants)	T (1-40), T (1-60), T (1-80)
Gray Sort Contest	G (Masters-Servants)	G (5-200), G (7-200), G (10-200)
Financial Time Series	F (Masters-Servants)	F (2-100), F (3-110), F (4-120)
Airline Query Process	A (Masters-Servants)	A (5-50), A (5-60), A (5-70)

is more than or equal to the required number of master and servant cores for these applications. Applications $(\mathrm{AP_i})$ are assumed to have master core $(\mathrm{MA_i})$ and servant core $(\mathrm{SA_i})$ requirements. The target *BiGNoC* platform is characterized by its clusters $(\mathrm{C_j})$, master cores $(\mathrm{MC_j})$, and servant cores $(\mathrm{SC_j})$. First, the applications and *BiGNoC* platform clusters are sorted in the descending order of their $\mathrm{SA_i}$ and $\mathrm{SC_j}$ counts, respectively (steps **1-2**). In steps **3-4**, the algorithm initializes the required number of master cores $(\mathrm{NMA_i})$ and servant cores $(\mathrm{NSA_i})$ that are to be scheduled for each application, and also initializes the number of available free master cores $(\mathrm{FMC_j})$ and free servant cores $(\mathrm{FSC_j})$ in each cluster of *BiGNoC*, respectively. A nested loop iterates over all applications $(\mathrm{AP_i})$ and clusters $(\mathrm{C_j})$ in steps **5-6**. If $\mathrm{FSC_j}$ are available in cluster $\mathrm{C_j}$ at step **7**, then in steps **8-25**, we assign master and servant cores of *BiGNoC* to applications. We compare the number of available free servant cores within a cluster with the number of servant cores required by an application. If the number of free servant cores within a cluster are greater (steps **8-10**), then we assign the required free servant cores in the current cluster to the current application, else we assign all the free servant cores in the current cluster to the current application (steps **17-19**). For every free servant core assignment to an application in a cluster, we also compare the number of available free master cores within the cluster with the number of master cores required by an application. If the number of free master cores within a cluster are greater (steps **11-13** and **20-22**), then we assign the required free master cores in the current cluster to the current application, else we assign all the free master cores in current cluster to the current application (steps **17-19** and **23-25**). The proposed algorithm is used to schedule applications on both variants of *BiGNoC*.

6 Experiments

6.1 Experimental Setup

To evaluate the proposed *BiGNoC* architecture, we compared it with a traditional electrical mesh NoC (EMesh) and a broadcast optimized electrical mesh NoC (BO-EMesh) [33] as well as with two state-of-the-art photonic crossbar NoCs: Flexishare with token stream arbitration [11] and Firefly with a reservation assisted SWMR (R-SWMR) waveguide groups [9]. We modeled and simulated the NoC architectures at a cycle-accurate granularity with a SystemC-based NoC simulator for a 256-core CMP platform. We used this NoC simulator to emulate the execution of big data benchmarks across different architectures. In Flexishare, Firefly, BO-EMesh, and EMesh architectures with 256-cores, we have considered 16 master cores (similar to the number of master cores in *BiGNoC;* recall that *BiGNoC* has 4 MNs, which corresponds to 16 master cores) and the remaining cores are considered as servant cores for a fair comparison with the *BiGNoC* architecture. We used five big data benchmarks [7], [12], [13], [14], [15] (Table 2) to create multi-application workloads. The goal with these workloads is to emulate an environment that executes future large-scale data analytics applications having different master and servant combinations with diverse bandwidth needs.

Table 2 shows the variants of big data benchmarks with different master-servant requirements considered for our analysis. We created 12 multi-application workloads from these benchmarks. Each workload combines 2 to 4 benchmarks, such that the summation of all the master cores and servant cores within the multi-application workload is lower than the number of available cores (i.e., 256) in the CMP. As an example, the T (1-40)-A (5-50)-F (2-100)-N (1-50) workload combines variants of Text Mining with 1-master and 40-servants (T (1-40)), Airline Query Processing with 5-masters and 50-servants (A (5-50)), Financial Time Series with 2-masters and 100-servants (F (2-100)), and Netflix Movie Rating with 1-master and 50-servants (N (1-50)), and schedules them to clusters $\mathrm{C_0}, \mathrm{C_1}, \mathrm{C_2}$, and $\mathrm{C_3}$ of *BiGNoC-HOM* and *BiGNoC-HET* using the application scheduling algorithm presented in Section 5.3. We analyzed the actual execution characteristics of the big data applications presented in Table 2 (such as the master processing time, servant processing time, etc.) that are measured using an Amazon's Elastic Compute Cloud (EC2) instance [35], to generate traces that were fed into our network simulator. We set a "warm-up" period of 1M cycles and executed the applications for 100M cycles.

We targeted a 22 nm process technology for the 256-core system. Based on geometric calculations of the waveguides for a $20\,\mathrm{mm} \times 20\,\mathrm{mm}$ chip dimension, we estimated the time needed for light to travel in a photonic waveguide with a length of 12 cm from the first to the last node in a single pass of the MWMR waveguide group in Flexishare as 8 cycles (i.e., 1.6 ns) at 5 GHz clock frequency. Throughout our analysis we use a flit size of 64 bits for BO-EMesh and EMesh and a total packet size of 512 bits for all PNoC architectures. We consider data modulation at both clock edges to enable simultaneous transfer of 512 bits in a single cycle, in the *BiGNoC-HOM*, *BiGNoC-HET*, Flexishare, and Firefly PNoCs. We considered an on-off switching time of 3.1 ps for a ring modulator and ring detector [11], which is less than one clock cycle (i.e., 200 ps) at 5 GHz frequency.

The static and dynamic energy consumption of the electrical routers is based on results obtained from the DSENT tool [29]. 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 [16], [36], [36], [37] and shown in Table 3. Here $\mathrm{E_{dynamic}}$ is the energy per bit for modulators and photodetectors and $\mathrm{E_{logic-dyn}}$ is the energy per bit for the driver circuits of modulators and photodetectors. $\mathrm{P_{SWMR-FY}}$ and $\mathrm{P_{MWMR-FX}}$ are the static power dissipation of SWMR and

TABLE 3
Energy and Losses For Photonic Devices [16], [36], [37]

Cluster-wise static power per waveguide group of BiGNoC			
Waveguide Type	**32-Node Power**	**16-Node Power**	**8-Node Power**
P_{MWSR}	1.54W	0.62 W	0.21W
P_{SWMR}	5.72 W	2.69 W	1.26 W
Static power per waveguide group			**Power**
$P_{SWMR\text{-}FY}$			1.15 W
$P_{MWMR\text{-}FX}$			3.73 W
Energy consumption type			**Energy**
$E_{dynamic}$			0.42 pJ/bit
$E_{logic-dyn}$			0.18 pJ/bit
Photonic loss type			**Loss (in dB)**
Microring through			−0.005
Waveguide propagation per cm			−0.274
Waveguide coupler/splitter			−0.2
Waveguide bending loss			0.005 per 90°

MWMR waveguide groups in Firefly and Flexishare architectures, respectively. Further, the P_{MWSR} and P_{SWMR} rows in Table 3 show static power dissipation of MWSR and SWMR waveguide groups of clusters in *BiGNoC* with sizes 32, 16, and 8 nodes, respectively. Also, we calculate power dissipation overheads of 75 mW, 35 mW, and 15 mW in the electrical circuits of the SWMR waveguide groups in clusters of *BiGNoC* with sizes 32, 16, and 8 nodes, respectively, to realize partial detuning based on estimates from prior work [8]. All the static power dissipation values for waveguides presented in Table 3 include the power overhead of MR thermal tuning. We consider an MR heating power of 15 μW per MR and receiver sensitivity of -20 dBm [42], [43]. To compute laser power dissipation, we calculated photonic loss in components, which sets the photonic laser power budget and correspondingly the electrical laser power. Lastly, based on our gate-level analysis, we estimate area overheads of $0.0065\ \text{mm}^2$ and $0.008\ \text{mm}^2$, and power overheads of 0.12W and 0.16W in the electrical arbiters for the MWSR waveguide groups of *BiGNoC-HOM* and *BiGNoC-HET*, respectively.

6.2 BiGNoC: Sensitivity Analysis

Our first set of experiments presents a sensitivity analysis to explore the optimal buffer size of the electrical router that is used for inter-cluster communication in two variants of our *BiGNoC* architecture with 256 cores: *BiGNoC-HOM* and *BiGNoC-HET*. *BiGNoC-HOM* has four homogeneous clusters with each cluster having 16 nodes; and *BiGNoC-HET* has four clusters with 32, 16, 8, and 8 nodes, respectively.

Figs. 7a and 7b show the average packet latency for three multi-application big data workloads on *BiGNoC-HOM* and *BiGNoC-HET*, with buffer depth of the electrical router varying from 8 to 40. The range of buffer depths (i.e., from 8 to 40) explored in this sensitivity analysis is decided based on the buffer depths used in the prior works [9], [11]. In this analysis, to compute average packet latency we have considered the delay incurred by the packet to move from the source node to the destination node along with the queuing delays in routers

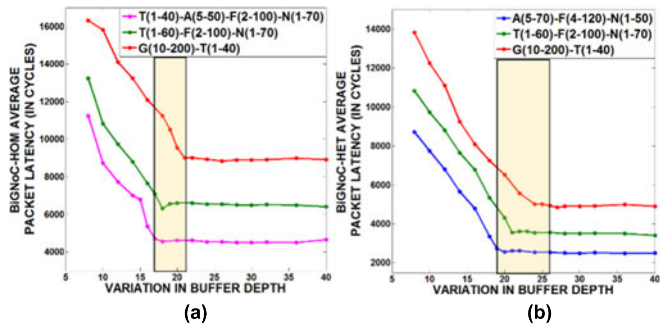
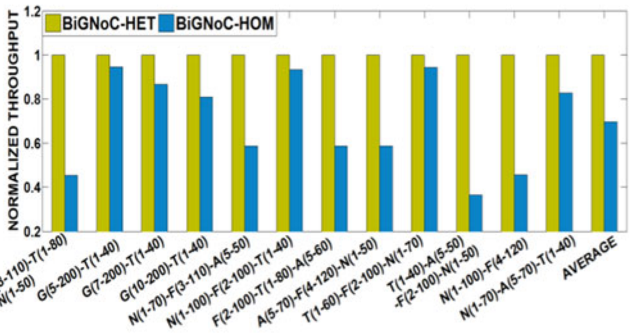


Fig. 7. Average packet latency comparison for (a) *BiGNoC-HOM* and (b) *BiGNoC-HET* in a 256-core CMP with different buffer depths (8-40).

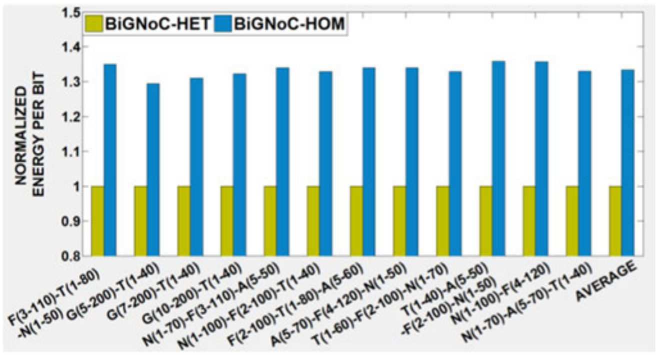
and interfaces. The three workloads were chosen to possess high, medium, and low aggregate inter-cluster traffic, to explore the impact of application traffic on buffer depth. We characterized inter-cluster traffic of an application by counting the number of transfers through the electrical router, which is used for inter-cluster communication.

At a particular buffer depth for both *BiGNoC-HOM* and *BiGNoC-HET*, Fig. 7 shows higher average packet latency for workloads with higher inter-cluster traffic (i.e., G(10-200)-T(1-40)) compared to workloads with lower inter-cluster traffic (i.e., T(1-40)-A(5-50)-F(2-100)-N(1-50) for *BiGNoC-HOM* and A(5-70)-F(4-120)-N(1-50) for *BiGNoC-HET*) as queuing of packets occurs at the master nodes for workloads with higher inter-cluster traffic, which increases their queueing delay and average packet latency. Also, for all workloads executing on both *BiGNoC-HOM* and *BiGNoC-HET*, a smaller buffer size should intuitively result in higher average packet latency, as the buffer in the electrical router becomes more frequently full and creates back pressure on the buffers in the MN of each cluster of *BiGNoC-HOM* and *BiGNoC-HET*. As a result, the centralized arbiter within each cluster stops assigning MWSR waveguide groups to SNs (due to Xon/Xoff flow control mechanism used within each cluster; for explanation see Section 3.2) in that cluster, which are used to transfer packets to MN, which in turn increases packet queuing delay within each SN and incurs higher average packet latency.

On the other hand, beyond a particular buffer depth in both *BiGNoC-HOM* and *BiGNoC-HET* the average packet latency of all the applications saturate. After a particular buffer depth, the buffer in the electrical router of both variants of BiGNoC seldom gets full, which is the main reason for this saturation. A careful observation of the plots in Fig. 7 shows that for workloads with lower inter-cluster traffic (i.e., T(1-40)-A(5-50)-F(2-100)-N(1-50) for *BiGNoC-HOM* and A(5-70)-F(4-120)-N(1-50) for *BiGNoC-HET*) latency saturation occurs at a small buffer depth, whereas for workloads with higher inter-cluster traffic (i.e., G(10-200)-T(1-40) for both *BiGNoC-HOM* and *BiGNoC-HET*) latency saturation occurs at a large buffer depth. However, as shown in Figs. 7a and 7b, there is a region (light yellow shaded region) between saturation points of low inter-cluster traffic application and high inter-cluster traffic application, where both *BiGNoC-HOM* and *BiGNoC-HET* archive optimal performance. Therefore, we chose to use 21 and 26 as the optimal buffer depth for *BiGNoC-HOM* and *BiGNoC-HET*,



(a)



(b)

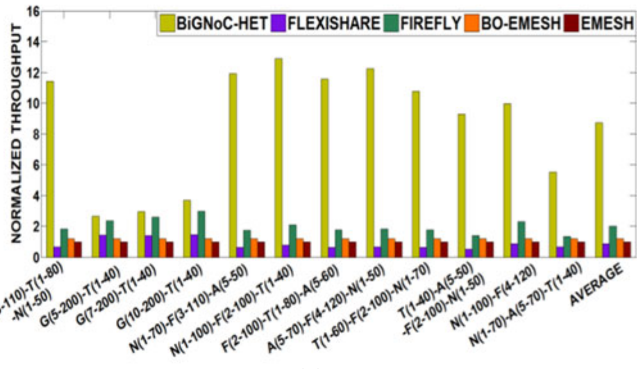
Fig. 8. (a) Normalized throughput, (b) normalized EPB comparison of *BiGNoC-HOM* with *BiGNoC-HET* for 256-core CMP. Results are shown for multi-application workloads and normalized w.r.t. *BiGNoC-HET.*

respectively, which are the highest buffer depths of the optimal performance regions shown in Figs. 7a and 7b. We use these optimal buffer depths for *BiGNoC-HOM* and *BiGNoC-HET* in the rest of our analysis.

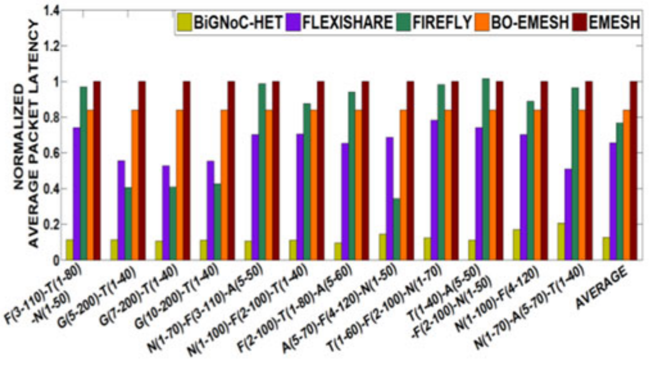
6.3 Experimental Results

Our next set of experiments presents a comparative study between *BiGNoC-HOM* and *BiGNoC-HET*. We used the optimal buffer depth of 21 and 26 for *BiGNoC-HOM* and *BiGNoC-HET*, respectively (determined as per the previous subsection) in this comparative study. Figs. 8a and 8b present detailed simulation results that quantify the average throughput and energy-per-bit (EPB) for *BiGNoC-HOM* and *BiGNoC-HET*, for twelve multi-application workloads. Results are normalized with respect to the *BiGNoC-HET* results.

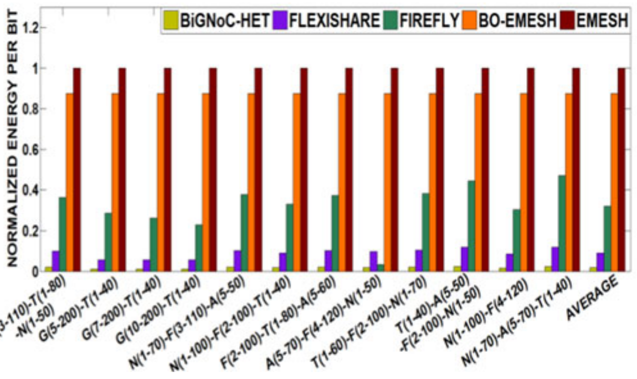
From Fig. 8a it can be seen that on an average *BiGNoC-HET* has 30.4 percent higher average throughput compared to *BiGNoC-HOM*. Variable cluster sizes in *BiGNoC-HET* help reduce the inter-cluster traffic while executing big data workloads involving different master-servant combinations. This decrease in inter-cluster traffic improves utilization of MWSR and SWMR waveguides within a cluster and increases the throughput of *BiGNoC-HET* compared to *BiGNoC-HOM*. Also, from Fig. 8b it can be observed that on an average *BiGNoC-HET* has 33.3 percent lower EPB compared to *BiGNoC-HOM*. The increase in data rate and decrease in trimming energy (due to decrease in number of detectors) decreases the EPB of *BiGNoC-HET* compared to *BiGNoC-HOM* even though there is increase in laser energy for *BiGNoC-HET*. From the average throughput and EPB results presented in Fig. 8, we can summarize that *BiGNoC-HET* achieves better performance with lower EPB compared to *BiGNoC-HOM*, which highlights its viability for executing future large-scale data analytics applications. Therefore, for our next set of experiments we have used only *BiGNoC-HET* to estimates benefits over electrical and photonic NoC architectures from prior work.



(a)



(b)



(c)

Fig. 9. Normalized (a) throughput (b) latency (c) EPB comparison of *BiGNoC-HET* with other architectures for a 256-core CMP. Results are for multi-application workloads and normalized w.r.t. EMesh.

In the next set of experiments, we compare network throughput, average packet latency, and energy-per-bit (EPB) of *BiGNoC-HET* with the EMesh, BO-EMesh, Flexishare with token stream arbitration [11], and Firefly with R-SWMR waveguide [9] architectures. Figs. 9a, 9b, 9c show the results of this comparative analysis, where all the results are normalized with respect to the EMesh results. From the throughput comparison in Fig. 9a, it can be observed that, not surprisingly, *BiGNoC-HET* provides 8.7× and 7.2× higher throughput than EMesh and BO-EMesh, respectively, due to the presence of higher bandwidth photonic links for data communication. Furthermore, as shown in

TABLE 4
Photonic Hardware Comparison

Architecture	Waveguides	Modulators	Detectors
BiGNoC-HOM	12	31,744	17,408
BiGNoC-HET	10	33,280	11,776
Flexishare	33	131,080	131,648
Firefly	64	4,096	28,672

Fig. 9a, throughput improvements of BiGNoC-HET are significantly higher for most of the application combinations, except the application combinations that have the Gray Sort Contest (G) application. As this single large application utilizes a significant portion of the BiGNoC architecture (by utilizing 200 servant cores) for which a major portion of the traffic traverses the 4×4 electrical routers (for inter-cluster communication), the bottlenecks at these routers limit BiGNoC-HET performance.

BiGNoC-HET has nearly $9.9\times$ greater throughput compared to Flexishare. Even though Flexishare uses MWMR waveguides and time division multiplexing (TDM), its token stream arbitration reduces its waveguide utilization and overall throughput compared to *BiGNoC-HET*. 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, which limits Flexishare's ability to perform simultaneous data transfers. In contrast, *BiGNoC-HET* has dedicated photonic paths (MWSR waveguide group for SN-to-MN communication and SWMR waveguide group for MN-to-SN communication) between the master node and servant nodes within each cluster. This helps in increasing simultaneous data transfers in *BiGNoC-HET* with increase in number of clusters. *BiGNoC-HET* also facilitates efficient multicasting to improve throughput over Flexishare by using its SWMR waveguide groups from MN to SNs, whereas in Flexishare, multiple unicast packets are sent from the master core to servant cores instead of a single multicast packet.

BiGNoC-HET has $4.4\times$ higher throughput compared to Firefly. This is due to the near light speed communications for a majority of the path traversed by the data in *BiGNoC-HET* using photonic links, whereas Firefly being a hybrid network, utilizes slower electrical links for a significant portion of the path traversed by the data. These mechanisms also improve the average packet latency in *BiGNoC-HET*, as shown in Fig. 9b, by reducing the time spent waiting for access to the photonic waveguides. On average *BiGNoC-HET* has 81 percent, 84 percent, 85 percent, and 88 percent lower average packet delay over Flexishare, Firefly, BO-EMesh, and EMesh, respectively for the different multi-application workloads.

Fig. 9c shows the EPB comparison between the architectures. It can be observed that on average *BiGNoC-HET* has 88 percent, 90 percent, 96 percent, and 98 percent lower EPB compared to Flexishare, Firefly, BO-EMesh, and EMesh, respectively. *BiGNoC-HET* has lower EPB compared to BO-EMesh and EMesh, as it uses energy efficient photonic links for data transfer instead of power hungry electrical links. Most of the energy in the photonic architectures was consumed in the form of static energy. Table 4 shows the photonic hardware comparison between the PNoC architectures. It can be seen that *BiGNoC-HET* has 82 percent less photonic hardware compared to Flexishare. This reduction in photonic hardware reduces its overall static energy consumption and its EPB. Although both *BiGNoC-HET* and Firefly use multicasting in their SWMR waveguides, the lower EPB of *BiGNoC-HET* compared to Firefly is due to the higher energy consumption in the electrical network of the Firefly architecture.

7 Conclusion

We presented a new application-specific *BiGNoC* architecture that features master-servant clusters with efficient utilization of SWMR and MWSR waveguides to improve performance while executing large-scale data analytics applications. *BiGNoC* exploits efficient multicasting in photonic waveguides to achieve high data rates. In particular, we showed how *BiGNoC-HET*, a variant of *BiGNoC*, improves performance due to improved photonic channel utilization and its ability to adapt to time-varying application performance goals while co-running multiple large-scale data analytics applications. *BiGNoC-HET* improves throughput by up to $9.9\times$, packet latency by up to 88 percent, and energy-per-bit by up to 98 percent over traditional EMesh, broadcast optimized EMesh, and state-of-the-art photonic NoC architectures (Flexishare and Firefly). These results corroborate the excellent capabilities of our proposed *BiGNoC* architecture towards executing large-scale data analytics applications.

Acknowledgments

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

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Comm. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[2] Hadoop. (2016). [Online]. Available: https://hadoop.apache.org

[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. USENIX Conf. Hot Topics Cloud Comput.*, 2010, Art. no. 10.

[4] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): low cost low power servers for internet-scale services," in *CIDR*, 2009, pp. 1–8.

[5] Y. Xia, T. S. E. Ng, X. S. Sun, "Blast: Accelerating high-performance data analytics applications by optical multicast," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1930–1938.

[6] KNN Refinement. (2013). [Online]. Available: https://www3.nd.edu/~steve/computing_with_data/17_Refining_kNN/refining_knn.html.

[7] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM*, 2011, pp. 98–109.

[8] C. Li, M. Browning, P. V. Gratz, and S. Palermo, "Energy-efficient optical broadcast for nanophotonic networks-on-chip," in *Proc. Optical Interconnects Conf.*, 64–65, 2012.

[9] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating future network-on-chip with nanophotonics," in *Proc. Int. Symp. Comput. Archit.*, 2009, pp. 429–440.

[10] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. Beausoleil, and J. Ahn, "Corona: System implications of emerging nanophotonic technology," in *Proc. Int. Symp. Comput. Archit.*, 2008, pp. 153–164.

[11] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel sharing for an energy efficient nanophotonic crossbar," in *Proc. Int. Symp. High Perform. Comput. Archit.*, 2010, pp. 1–12.

[12] Text Mining. (2007). [Online]. Available: https://www.cs.umb.edu/~smimarog/textmining/datasets/

[13] R. S. Tsay, *Analysis of Financial Time Series*. Hoboken, NJ, USA: Wiley, ISBN 0-471-41544-8, 2002.

[14] Airline Dataset. (2010). [Online]. Available: http://www.stat.purdue.edu/~sguha/rhipe/doc/html/airline.html

[15] Sort Benchmark. (2009). [Online]. Available: http://sortbenchmark.org/

[16] L.H.K. Duong, M. Nikdast, S. Le Beux, J. Xu, X. Wu, Z. Wang, and P. Yang, "A case study of signal-to-noise ratio in ring-based optical networks-on-chip," *IEEE Des. Test*, vol. 31, no. 5, pp. 55–65, Oct. 2014.

[17] R. Morris and A. K. Kodi, "Power-efficient and high-performance multilevel hybrid nanophotonic interconnect for multicores," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, 2010, pp. 207–214.

[18] J. Psota, J. Miller, G. Kurian, H. Hoffman, N. Beckmann, J. Eastep, and A. Agarwal, "ATAC: On-chip optical networks for multicore processors," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 3325–3328.

[19] E. Fusella, J. Flich, and A. Cilardo, "Path setup for hybrid NoC architectures exploiting flooding and standby," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 5, May 2017.

[20] L. Yang, W. Liu, W. Jiang, M. Li, P. Chen, and E. H. Sha, "FoToNoC: A folded torus-like network-on-chip based many-core systems-on-chip in the dark silicon era," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, Jul. 2017.

[21] A. Kulkarnl, T. Abtahi, E. Smith, and T. Mohsenin, "Low energy sketching engines on many-core platform for big data acceleration," in *Proc. Int. Great Lakes Symp. VLSI*, 2016, pp. 57–62.

[22] K. Kanoun, M. Ruggiero, D. Atienza, and M. Schaar, "Low power and scalable many-core architecture for big-data stream computing," in *Proc. IEEE Comput. Society Annu. Symp. VLSI*, 2014, pp. 468–473.

[23] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, Aug. 2013.

[24] S. Carrillo, J. Harkin, L. J. McDaid, F. Morgan, S. Pande, S. Cawley, and B. McGinley, "Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 22, pp. 2451–2461, Dec. 2013.

[25] V. Dmitri and R. Ginosar, "Network-on-chip architectures for neural networks," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, 2010, pp. 135–144.

[26] A. Firuzan, M. Modarressi, and M. Daneshtalab, "Reconfigurable communication fabric for efficient implementation of neural networks," in *Proc. 10th Int. Symp. Reconfigurable Commun.-Centric Syst.-on-Chip*, 2015, pp. 1–8.

[27] J. Lee, S. Li, H. Kim, and S. Yalamanchili, "Design space exploration of on-chip ring interconnection for a CPU-GPU heterogeneous architecture," *J. Parallel Distrib. Comput.*, vol. 73, no. 12, pp. 1525–1538, 2013.

[28] W. Choi, K. Duraisamy, R. G. Kim, J. R. Doppa, P. P. Pande, R. Marculescu, and D. Marculescu, "Hybrid network-on-chip architectures for accelerating deep learning kernels on heterogeneous manycore platforms," in *Proc. Int. Conf. Compliers Archit. Sythesis Embedded Syst.*, Oct. 2016, pp. 1–10.

[29] C. Sun, C. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, and V. Stojanovic, "DSENT - A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, 2012, pp. 207–214.

[30] N. E. Jerger, L. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Proc. Int. Symp. Comput. Archit.*, 2008, pp. 229–240.

[31] I. Thakkar, S. V. R. Chittamuru, and S. Pasricha, "Run-time laser power management in photonic NoCs with on-chip semiconductor optical amplifiers," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, 2016, pp. 1–4.

[32] C. Chen and A. Joshi, "Runtime management of laser power in silicon-photonic multibus NoC architecture," *IEEE J. Select. Topics Quantum Electron.*, vol. 2, no. 19, Mar./Apr. 2013, Art. no. 3700713.

[33] T. Krishna, L. Peh, B. M. Beckmann, S. K. Reinhardt, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2011, pp. 71–82.

[34] J. Dongarra, "Report on the sunway taihulight system," in Technical Report UT-EECS-16-742, Jun. 2016.

[35] Amazon EC2. (2017). [Online]. Available: http://aws.amazon.com/ec2.

[36] X. Zheng, D. Patil, J. Lexau, F. Liu, G. Li, H. Thacker, Y. Luo, I. Shubin, J. Li, J. Yao, P. Dong, D. Feng, M. Asghari, T. Pinguet, A. Mekis, P. Amberg, M. Dayringer, J. Gainsley, H. F. Moghadam, E. Alon, K. Raj, R. Ho, J. E. Cunningham, and A. V. Krishnamoorthy, "Ultra-efficient 10Gb/s hybrid integrated silicon photonic transmitter and receiver," *Opt. Express*, vol. 19, no. 6, pp. 5172–5186, Mar. 2011.

[37] P. Grani and S. Bartolini, "Design options for optical ring interconnect in future client devices," *ACM JETC*, vol. 10, no. 4, May 2014, Art. no. 30.

[38] Z. Wang, S. Zhang, B. He, and W. Zhang, "Melia: A mapreduce framework on open CL-based FPGAs", *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3547–3560, Dec. 2016.

[39] S. V. R. Chittamuru, S. Desai, and S. Pasricha, "SWIFTNoC: A reconfigurable silicon-photonic network with multicast enabled channel sharing for multicore architectures," *ACM J. Emerging Technol. Comput. Syst.*, vol. 13, no. 4, Feb. 2017, Art. no. 58.

[40] C. Li, M. Browning, P. V. Gratz, and S. Palermo, "LumiNOC: A power-efficient, high-performance photonic network-on-chip," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 33, no. 6, pp. 826–838, Jun. 2014.

[41] E. Kakoulli, V. Soteriou, C. Koutsides, and K. Kalli, "Design of high-performance, power-efficient optical NoCs using silica-embedded silicon nanophotonics," in *Proc. IEEE Int. Conf. Comput. Des.*, Oct. 2015, pp. 1–8.

[42] R. Hendry, D. Nikolova, S. Rumley, N. Ophir, and K. Bergman, "Physical layer analysis and modeling of silicon photonic WDM bus architectures," in *Proc. HiPEAC Workshop*, 2014.

[43] I. Thakkar, S. V. R. Chittamuru, and S. Pasricha, "A comparative analysis of front-end and back-end compatible silicon photonic on-chip interconnects," in *Proc. of SLIP*, 2016, pp. 1–8.

Sai Vineel Reddy Chittamuru (S'14) received the BTech degree in electrical engineering from the Indian Institute of Technology Kharagpur (IIT-KGP) in 2011, and the PhD degree in electrical and computer engineering, from Colorado State University, Fort Collins, Colorado, in 2018. His research interests include embedded system, systems-on-chip, and optical networks-on-chip. He is the recipient of best paper awards from IEEE/ACM SLIP 2016 and ACM GLSVLSI 2015 conferences, and a best paper nomination from the IEEE ISQED 2016 conference for his research contributions. He is Student Member of the IEEE.

Dharanidhar Dang (S'15) working toward the PhD degree in computer engineering at Texas A&M University, College Station, Texas. His thesis advisor is Dr.Rabi N Mahapatra. Dang's research area includes silicon photonics, neuromorphic computing, computer architecture, deep learning, and VLSI. He has seven publications in top EDA events and several under reviews in IEEE and ACM transactions. He has been honored as a teaching fellow by College of Engineering, TAMU. He won multiple national and international competitions such as gold medal in Intel Embedded Challenge, 5000$ cash prize in Youth Enterprise, 2500$ in Schneider Electric Innovation Challenge. In his PhD thesis, Dang is investigating silicon photonic architecture for exascale computing. He is Student Member of the IEEE.

Sudeep Pasricha (M'02, SM'13) received the BE degree in electronics and communication engineering from the Delhi Institute of Technology, India, in 2000, and the MS and PhD degrees in computer science from the University of California, Irvine, in 2005 and 2008, respectively. He is currently a Monfort and Rockwell-Anderson professor of Electrical and Computer Engineering, Colorado State University, Fort Collins, Colorado. His research interests include energy efficiency and fault tolerant design for network and memory architectures in multicore computing. He is in the editorial board of various journals such as *IEEE Transactions on Computer-Aided Design*, *IEEE Transactions on Multi-Scale Computing Systems*, *IEEE Transactions on Embedded Computing Systems*, etc. He is or has been an Organizing Committee Member and/or Technical Program Committee member of various IEEE/ACM conferences such as DAC, DATE, CODES+ISSS, NOCS, ISQED, VLSID, and GLSVLSI. He is the recipient of several awards for his research contributions, including the 2015 IEEE TCSC Award for Excellence for a mid-career researcher and the 2013 AFOSR Young Investigator Award, as well as five Best Paper Awards. He is Senior Member of the IEEE.

Rabi Mahapatra (M'94, SM'02) is a professor in the Department of Computer Science and Engineering, Texas A&M University, Texas. He was a faculty with the Indian Institute of Technology, Kharagpur, India and a faculty fellow wih IBM T. J Watson Research Center. His principal areas of research are Embedded Systems, System on Chip, Low-power system design, and Data analytic accelerators. He directs the Embedded System Research Group at Texas A&M University. He has been on the Editorial board of *ACM Transactions on Embedded Computing*, *IEEE Transactions on Parallel Distributed Systems*, and *EUROSIP Journal on Embedded Systems*. He is a ford fellow, BOYS-CAST fellow, and was a distinguished visitor of IEEE Computer Society. He received Undergraduate Teaching Excellence award in 2010. He is Senior Member of the IEEE.