ENERGY-EFFICIENT PHOTONIC ARCHITECTURES FOR LARGE-SCALE DATA ANALYTICS

A Dissertation

by

DHARANIDHAR DANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Rabi N. Mahapatra
Committee Members,	Duncan M. Walker
	Eun Jung Kim
	Samuel Palermo
Head of Department,	Dilma Da Silva

December 2018

Major Subject: Computer Engineering

ABSTRACT

With silicon technology reaching its physical limit, conventional computing systems are incapable of offering ever-increasing performance requirement with limited power budget. This has propelled semiconductor community to seek for new computing paradigms that can offer high energy-efficiency. Silicon photonics with its ultra-low power characteristics, inherent parallelism, and large multiplexing capability, is one such promising paradigm. The goal of this research is to utilize silicon photonics to design energy-efficient exascale computing architectures.

This study is established through research in a number of directions. First, we propose a non-blocking, 5×5, low-cost on-chip photonic router. It incorporates mode-division-multiplexing in addition to wavelength-division-multiplexing and time-division-multiplexing for high-throughput. It is a first of its kind to the best of our knowledge. We use this router to design high-performance 2D and 3D mesh photonic network-on-chip (PNoC). Further, we introduce a novel laser-multiplexing scheme to further enhance the energy-efficiency of our PNoC designs. Components in a photonic system are highly susceptible to thermal variations. We propose IHDTM, a cross-layer dynamic thermal management techinique which is a combination of device-level optimization and system-level thread migration. After demonstrating a highly reliable energy-efficienct photonic system, we intend to devise a high-performance photonic architecture for exascale data analytic applications. Multicast data dissemination is a major performanec bottleneck for data analytic 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, we propose BiGNoC, a manycore chip platform with a novel application-specific photonic on-chip network architecture. Finally, we intend to utilize the exascale parallelism and ultrafast characteristics of silicon photonics to extend the state of the art in deep learning accelerator architectures. Trainng a deep learning network involves expensive computation overheads. As a result, most of the accelerators use pre-trained weights and focus only on improving the design of inference phase. We propose a novel photonic-based backpropagation accelerator for high performance deep learning training. In ad-

dition, we present a design for convolutional neural network, BPLight-CNN, which incorporates the novel photonic backpropagation accelerator. BPLight-CNN is a first-of-its-kind photonic and memristor-based CNN architecture for end-to-end training and prediction.

DEDICATION

To my parents.

ACKNOWLEDGMENTS

First, I want to thank my advisor, Dr. Rabi N Mahapatra for his generosity wisdom, and guidance throughout my graduate studies. He has continuosuly inspired and challenged me to evolve as a resarcher. This dissertation would not have reached to its fruition without his advice. I would also like to thank my thesis committee members, Dr. Duncan Walker, Dr. Eun Jung Kim, and Dr. Samuel Palermo for agreeing to be on the committee and for providing valuable suggestions and criticisms. I am grateful to Dr. Dilma Da Silva for her kindly advice. I am also thankful to Dr. Vikram Kinra of Aerospace Department for offering me a teaching fellowship which was beneficial to strengthen my teaching skill. The staff at Texas A&M's Department of Computer Science & Engineering have been very nice and helpful during the course of graduate school. Special thanks to Mr. Bruce Veals for providing logistics in a timely fashion.

I am thankful to my collaborators Dr. S.V. R. Chittamuru of Micron, Dr. Martin Fiers of Luceda Photonics, Dr. Karthik Swaminathan of IBM TJ Watson, Dr. Christopher Nitta of University of California, Davis, for giving time and valuable ideas during my PhD. Special thanks to Dr. Sudeep Pasricha of Colorado State University for his critical remarks. I would also like to thank my amazing labmates Jyotikrishna Dass, Syed Ali Hasnain, Karl Ott, Jerry Yiu, Deam Ieong, Biplab Patra for numerous priceless brainstorming sessions.

My work would not have been possible without the support of my family. My parents have always been a continuous source of inspiration. I would also like to thank my younger brother, Subrat Kumar Dang, for taking care of my family in my absense. Special thanks to my sisters Pramodini, Tilottama, Meera, and Bidyutlata for their unconditional love and affection. I am grateful to my mentor Dr. Debashis Sahoo of University of California, San Diego for inspiring and supporting me throughut my stay in the U.S.

I am highly thankful to Texas A&M Cricket Club and Austin Cricket Club to provide me with the opportunity to play cricket regularly. This was highly beneficial to maintain a good balance of professional and personal endeavors.

Finally, I would like to thank my friends Rajarshi, Pulakesh, Abhijit, Hollie, Rushil, Atish, Sanjeev, Roneet, Akash, and Matthew for making my stay in the U.S. an unforgettable journey. Special thanks to Susovita, Artta, and Aurosmita for their affection and encouragement. During time away from home, they have been a constant source of support, motivation, and happiness.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis (or) dissertation committee consisting of Professor Rabi N Mahapatra [advisor], Professor Duncan H. Walker, and Professor Eun Jung Kim of the Department of Computer Science & Engineering and Professor Samuel Palermo of the Department of the Electrical & Computer Engineering.

The experimental analysis in chapter 4 was conducted in part by Dr.Sudeep Pasricha of Colorado State University.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a teaching fellowship from College of Engineering, Texas A&M University and graduate assistantships from the Department of Computer Science & Engineering.

NOMENCLATURE

MRR	Microring Resonator
PNoC	Photonic Network-on-Chip
CMP	Chip Multiprocessor
SOC	System-on-Chip
CNN	Convolutional Neural Network
DWDM	Dense Wavelength-division-multiplexing
TDM	Time-division-multiplexing
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
SOA	Semiconductor Optical Amplifier

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

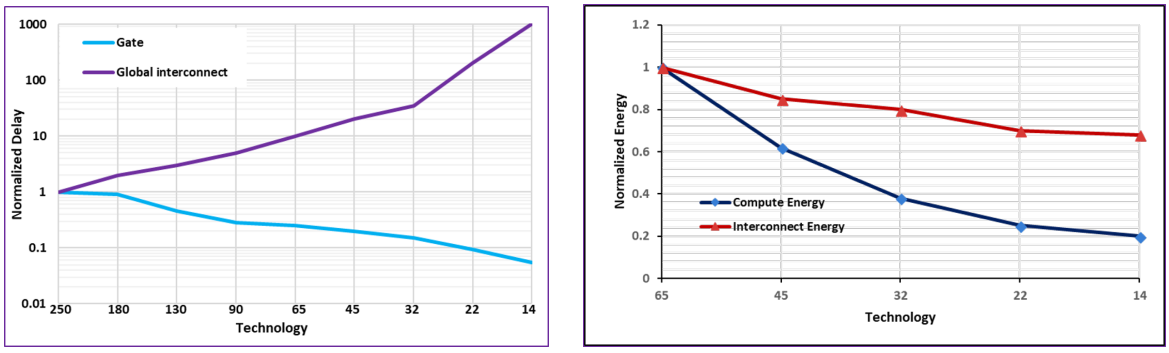
LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 High Performance Computing hits the wall!

The trend of integrating several computing cores into a single die has seen exemplary rise in the last decade. If the same trend continues we may find more than 200 cores in a single chip by 2025. With the rise in core-density in chip-multiprocessors, the on-chip communication fabric, commonly known as Network-on-Chip (NoC), has become a power and performance bottleneck. This is solely due to the technological limitations of metallic interconnects to scale energy and delay at the same rate as that of a core. As illustrated in Fig.1.1(a), the gap between metallic interconnect delay and gate delay rises rapidly as transistor size shrinks [17]. In addition to that, the scaling in energy consumption of metallic interconnects is very low compared to the scaling in computing core as shown in Fig.1.1(b). The metallic interconnect latency could be reduced by using repeaters; however, that incurs high energy consumption.



(a) (b)

Figure 1.1: Comparison of (a) delay of metallic interconnects & gate, and (b) energy-consumption of metallic interconnects & compute core, w.r.t. technology scaling

The speed of metallic interconnects is thus limited by power budget as a result of which further scaling of chip multiprocessors is detrimental. Therefore, it is a general consensus that metallic

interconnects alone will not be able to satisfy the power and performance demands of future many-core processors. This has propelled the research community to explore more efficient interconnect technologies. Silicon photonics with its low-power characteristics and inherent parallism is one such technology.

1.2 Advent of Silicon Photonics

On-chip photonic links, enabled by breakthrough in silicon photonics, provide several advantages over traditional metallic counterparts, such as near light speed data transfer, higher bandwidth density, and low power dissipation. Moreover, photonic links have several times lower data-dependent energy consumption compared to electrical wires, enabling the design of high-radix networks that are easier to program. Silicon photonics is thus becoming an exciting new option for on-chip communication, and has catalyzed much research in the area of photonic NoCs (PNoCs) for manycore systems. Silicon photonics could be fabricated monolithically on the same die or on a separate layer through 3D integration, thus forming the foundation of combining electronic and photonic devices in next-generation chip designs.

Establishing silicon photonics as an ideal alternative to electrical interconnect technologies would require strong cases in support of photonics's superiority in terms of both power consumption and performance. This calls for novel photonic on-chip network designs that efficiently utilize photonic links, advances in photonic devices, and automatic CAD tools to assist engineers in the designing process.

1.3 Silicon Photonic Basics

Photonic interconnects and routers are the building blocks of photonic NoCs. A photonic router consists of high-speed microring resonator(MRR) based switches and extremely low-latency photonic waveguides to provide photonic NoC as an alternative to electrical NoCs. Before diving into the details it is better to be familiar with some of the photonic components which are used and to have a understanding of how a simple communication takes place between two cores in a PNoC.

Silicon Photonic Waveguide:- Silicon photonic waveguide is the building block of several photonic components and lays the foundation for integrating photonics with a Silicon substrate. They are the basis of many optical devices such as MRRs, Directional Couplers, Multiplexers and Demultiplexers, communication channel etc. It basically consists of a Si core having an extremely small cross-section sorrounded by SiO_2 cladding material. Due to the difference in refractive index of the core and the cladding the light travels being confined within the waveguide due to total internal reflection.

MRR:- Micro-ring resonator is one of the priniciple components of a photonic network-on-chip. As quoted in the Laser and Photoics Reviews, A generic Ring Resonator consists of an optical waveguide which is looped back on itself, such that a resonance occurs when the optical path length of the resonator is exactly a whole number of wavelengths. So in this way a MRR is capable of supporting multiple resonances which have a gap equal to the free spectral range(FSR) which in turn depends on the resonator length. The practical application of the MRR is always along with a coupler which is necessary for it to interact with the outside world. It is basically an accessing mechanism. When in the loop the round trip phase shift equals the integer times 2π , resonance occurs and the wave is coupled successfully to or from the MRR.

Silicon photonic Laser:- Laser sources can be of different types depending upon the need of transmission. In the proposed design we have used Mode-Locked Lasers which will be described in the upcoming sections. The important fact is that they are the major source of power in a PNoC.

Photo-detector:- At the reciever end, the light passing through the silicon waveguide must be detected in order to reconvert the modulated data into electronic form. Photodetectors contain PIN photodiodes to convert optical powerinto electric current. Recently graphene is said to be a promising material for ultra-broadband photodetectors[18].

Silicon Optical Modulator:- It is used to modulate a light beam propagating in the optical waveguide with electrical data signals. The most common method of achieving modulation in silicon devices so far has been to exploit the plasma dispersion effect, in which the concentration of free charges in silicon changes the real and imaginary parts of the refractive index[19]. Micro-Ring modulators has been widely explored due to its compact footprint and low drive voltage. The nature resonant frequency of a micro-ring can be shifted by an index change and thus a large modulation depth occurs near the resonance.

Optical coupler:- Its a challenge in silicon photonics to obtain efficient coupling between highly confined mode in a waveguide and the large diamter mode in optical fibre to couple light in and out of the chip. Recently several solutions like surface gratings and tapers in which the thickness and width of silicon layer are increased are proposed.

Silicon photonic transmitter:- The silicon photonic transmitter comprises of the laser source, optical modulator that modulates the light signal with electric signal data and the multiplexer that multiplexes the signal light of multiple wavelengths or modes in this case to be transmitted on a single transmission line.

The process of optical communication starts with the laser source producing the light which is coupled into the waveguide on chip with the help of a silicon optical coupler. The Ring modulator along with the multiplexer then modulates the electronic signal from the core on to the appropriate channel. It is transmitted along the waveguide till the destination. The propogation path can be changed or switched by appropriate photonic switches along the path made by the combination of MRRs and waveguides. At the destination the light signal is detected by the photo-detector and is demodulated to feed the information to the destination processor core. The whole process is depicted in the following figure.

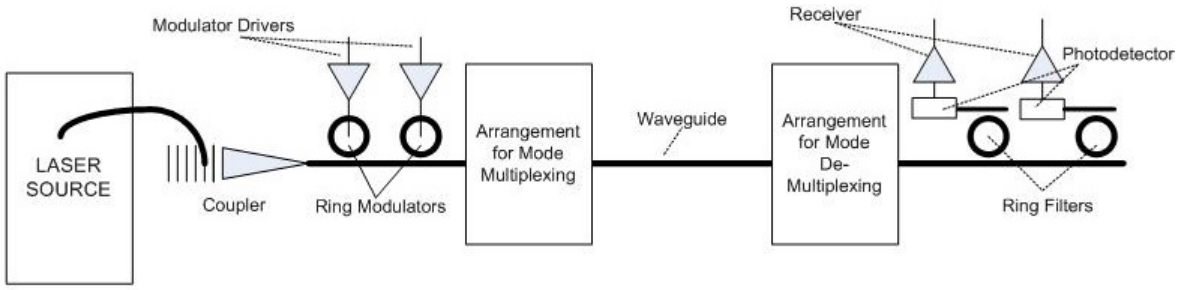


Figure 1.2: Communication flow in Silicon Photonics

1.4 Research Focus

This dissertation focuses on the efficient use of silicon photonics interconnects and devices to design high-performance computing architectures. It begins with the design of a novel ultrafast on-chip photonic router. Such a router is used to design efficient 2D and 3D photonic on-chip network architecture. Further, the dissertation studies the impact of thermal variations in photonic architectures to take necessary measures. In addition to high-throguhput photonic on-chip network architectures, the dissertation also investigates in designing deep learning architectures using photonic-based neuromorphic computing.

1.5 Contributions

The contributions of this dissertation are as follows:

Adaptive Multiplexing in Photonic Network-on-Chip:-In this work, we propose a non-blocking, low power, low-cost, and high performance 5Œ5 photonic router design using silicon microring resonators(MRR). In this router, we introduce wavelength-division-multiplexing (WDM) compatible mode-division-multiplexing (MDM) scheme for maximizing the aggregate bandwidth. The proposed photonic router is utilized to design low-cost and energy-efficient 2D and 3D photonic network-on-chip (PNoC). Laser is found to be the most power hungry element in a photonic system. The proposed PNoCs adopt a novel laser-multiplexing scheme to enhance their energy-efficiency. Our PNoC demonstrates 50% lower energy consumption and 25% lower area overhead compared to the state-of-the-art PNoC designs.

IHDTM: Islands of Heaters based Dynamic Thermal Management in Photonic Network-on-Chip:-the operation of photonic NoCs (PNoCs) is very sensitive to temperature variations that frequently occur on a chip. These variations can create significant reliability issues for PNoCs. For example, microring resonators (MRRs) which are the building blocks of PNoCs, may resonate at another wavelength instead of their designated wavelength due to thermal variations, which can lead to bandwidth wastage and data corruption in PNoCs. This paper proposes a novel run-time framework to overcome temperature-induced issues in PNoCs. The framework consists of (i) a PID controlled heater mechanism to nullify the thermal gradient across PNoCs, (ii) a device-level thermal island framework to distribute MRRs across regions of temperatures; and (iii) a system-level proactive thread migration technique to avoid on-chip thermal threshold violations and to reduce MRR tuningtrimming power by migrating threads between cores.Our experimental results with 64-core Corona and Flexishare PNoCs indicate that the proposed approach reliably satisfies on-chip thermal thresholds and maintains high network bandwidth while reducing total power consumption by up to 64.1%.

BigNoC: Application Specific Photonic Network-on-Chip Architecture Big Data Computing:-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 9.9Œ while reducing latency by up to 88% and energy-per-bit by up to

98% 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.

Neuromorphic Photonic Architecture for Deep Learning:Training deep learning networks involves continuous weight updates across the various layers of the deep network while using backpropagation algorithm (BP). This results in expensive computation overheads while training. As a result, most of the accelerators use pre-trained weights and focus only on improving the design of inference phase. However, the recent trend is to build a complete deep learning accelerator by incorporating the training module. Such efforts require an ultrafast architecture for executing the BP algorithm. In this paper, we propose a novel photonic-based backpropagation accelerator for high performance deep learning training. In addition, we present a design for convolutional neural network, BPLight-CNN, which incorporates the novel photonic backpropagation accelerator. BPLight-CNN is a first-of-its-kind photonic and memristor-based CNN architecture for end-to-end training and prediction. We simulate BPLight-CNN using a standard photonic-based CAD framework such as IPKISS for benchmark models, namely, LeNet and VGG-Net. The proposed design achieves at least 35x acceleration in training in addition to 31x improvement in computational efficiency and 45x energy saving compared to the state-of-the-art designs, without any loss of accuracy.

1.6 Organization

The rest of the dissertation is organized as follows. Chapter 2 illustrates the detailed design of an energy-efficient PNoC architecture with a novel multiplexign scheme. This is followed by demonstration of a dynamic thermal management framework for PNoCs in Chapter 3. Chapter 4 presents an application specific PNoC architecture design for large-scale data analytics applications. Then, Chapter 5 illustrates the design of a complete deep learning accelerator using silicon photonics. Finally, Chapter 6 concludes the dissertation with directions for future research.

2. ADAPTIVE MULTIPLEXING IN PHOTONIC NETWORK-ON-CHIP*

2.1 Motivation

[1]In the contemporary era, information is generated in abundance. In order to process those information, various efforts have been made to increase the processor performance. According to Moore's law the processor performance doubles up every two years. But until 2000 according to the law the processor speed kept on increasing. After that it was tough to increase the overall performance by increasing the efficiency of a single processor. This limitation or in other words constraint was overcome with the dawn of the era of chip multiprocessors(CMP). As was stated in [20][21] performance gain was to be derived by increase in the number of processor cores on chip. This approach has enhanced the throughput performance in CMPs by exploiting paralleilism with less power requirements. Semiconductor roadmap[17] predicts that in the next decade feature size will shrink to sub-10-nm regime. This leap in the process technology will scale the number of cores and threads per devices rather than increase speed or complexity of an indivdual core. This will result in faster processing with the help of thousands of threads running highly parallel codes. But this paradigm shift in the computing world poses newer challenges and issues which needs to be addressed.

Increasing density trend of CMPs involves considerable amount of interaction among the cores on chip. The cores will need to access data from local and distant caches as well as off-chip memory. As a result it requires higher bandwidth and a lower latency to support extensive communication among a large number of cores. Semiconductor NoCs would not provide such a large bandwidth while maintaining an acceptable level of power consumption[17]. The limitation of metallic interconnects in terms of loss, dispersion, cross-talk and speed are becoming increasingly obvious as interconnect density rises. In order to address the growing communication requirements, alternative on-chip interconnect paradigms are required.

Recently, integrated photonic links are being adopted as reliable and attractive alternative to tra-

[1]Adapted with permission from [2] & [3]

ditional metallic interconnects [22]. They hold promise to higher rate data transfer with minimal power dissipation [23]. Photonic links avoid capacitive, resistive and signal integrity constraints and allow efficient realization of physical connectivity. Also, *low loss in optical waveguides* [22] and *bit rate transparency* [24] are added advantages of photonic on-chip communication. The key power savings rises from the fact that once a photonic path is established, the data are transmitted end to end without the need for repeating, regenerating and buffering[25]. On the contrary in electrical NoCs messages are buffered, regenerated and transmitted over several router links en route to the destination. In addition to that over the past several years remarkable advances and breakthroughs have been made in the field of silicon photonics. All these support for a high performance, low power, and low cost photonic NoC (PNoC). PNoC is by far the most promising archetype to meet the needs of the next generation on-chip communication.

2.2 Related Works

In recent years several Photonic NoC (PNoC) architectures have been proposed using topologies like mesh [23], torus [22], crossbar[26], and clos[27].

We can classify photonic NoC architectures into two major categories based on the communication techniques used: (1) deterministic passive networks, and (2) dynamic switching networks. Wavelength selective passive networks utilize deterministic switching in which a fixed routing pattern is defined during the network design and the optical path between the source and destination is established by dynamically selecting a specific wavelength at the source or the destination [27]. Passive networks offer limited scalability and complicated design[28]. Whereas optical networks using dynamic switching are circuit switching networks, where the routing pattern is dynamically set beforehand by an electronic controller [23][22]. It is imperative to combine microelectronic control technology and photnic transmission as it is difficult to realize signal processing in photonics.

Various crosbar topology based PNoC architecture has been examined by the researchers as in [22][27] and [26]. They have proposed several channel sharing crossbar architectures called Single-Write-Multiple-Read(SWMR) or Multiple-Write-Single-Read(MWSR) architectures. Bat-

ten et al[26] implemented an opto-electrical global crossbar between small groups of cores and DRAM models[26]. Vantrease et al proposed Corona, a 3D MWSR channel sharing crossbar architecture where there is a dedicated channel for listening for each node, but several nodes compete to send messages on the same channel[29]. Pan et al also proposed a SWMR based crossbar design[30]. Though these designs correctly addredssed the latency issues faced by several other contenporary architectures they suffer from scalability as fully connected crossbars do not scale well[31].

Shacham et al proposed a hybrid electrical-photonic NoC architecture[25]. They used a electrical control circuit for arbitration and a photonic message transmission circuit. Though the use of a hybrid architecture was a brilliant idea which we have also followed in our design, the placement of MRRs as photnic switching elements involved turning on more number of MRRs. Though their design provides a higher bandwidth than the electrical NoCs the average latency per throughput is not significantly improved.

Hendry et all proposed a TDM based photonic arbitration circuit due to the absence of photonic buffers which addressed the issue of average latency[32]. But the TDM based arbitration makes the whole communication more deterministic rather than adaptive. Also it results in unnecessary turning on of MRRs in setting up paths between a source and destination.

It is reported in [26] that the wavelength selective passive network exhibits low latency as the wavelength selecting time is much shorter than the network configuration time. On the contrary, switching networks offer higher aggregate bandwidth by adopting WDM technology. Also, circuit switching in optical domain is more compact and it offers good scalability [33]. So among the various architectures discussed above almost all of them prefer a hybrid opto-electronic network.

Router is a critical component of NoC. Several MRR based optical routers have been proposed in the literature [33][23][25][34]. In [33], a low power, low cost, and non-blocking 5X5 optical router has been proposed using 16 MRRs. The design appears to be non-scalable due to significant power consumption when network size increases. In [25], the authors have proposed a 4×4

hybrid, blocking router using 8 MRRs. This design is complex and the aggregate bandwidth is limited due to its blocking nature. M. Briere et al proposed the λ router[35]. It uses a passive switching fabric and WDM technology. An $N \times N$ λ-router requires N wavelengths and multiple basic 2×2 switching elements to realize non-blocking switching function. In order to fully utilize all the components it prefers N to be even which may not be feasible in every case. The authors in [34] proposed a 4×4 λ-router using passive switching fabric with 30 MRRs incorporating Wave Division Multiplexing(WDM). Due to use of increasing number of waveguide crossings and MRRs, such a design results in higher insertion loss rendering it non-scalable. A. W. Poon et al proposed a 5×5 optical router based on an optimized crossbar[36]. In this architecture each port of the router is aligned to its corresponding direction to reduce the waveguide crossings around the switching fabric. Therefore a high performance, low power, and low cost photonic router is highly desirable for scalable NoC.

2.3 Contributions

In this chapter, we illustrate the following:-

(1) A novel 5×5 non-blocking photonic router design which incorporates *mode-division-multiplexing* in conjunction with *wavelength-division-multiplexing* for high performance. The proposed approach is the first of its kind to the best of our knowledge. (2) A logical layout of the photonic router. (3) Simulation of the proposed design with standard CAD tools which demonstrates $4\times$ higher throughput and up to 33% improvement in energy-saving compared to the best reported result. (4) A novel laser-multiplexing scheme for energy-efficient 3D PNoC architectures based on the proposed router.

2.4 Basics Of Silicon Photonics

Optical interconnects and routers are the building blocks of photonic NoCs. A photonic router consists of high speed Microring Resonator(MRR) based switches and extremely low-latency optical waveguides to provide photonic NoC as an alternative to electrical NoCs. Before diving into

the details it is better to be familiar with some of the photonic components which are the building blocks of the proposed PNoC and the associated terminologies. It is also necessary to have an understanding of how a simple communication takes place between two cores in a PNoC.

2.4.1 Silicon Photonic Components

Silicon Photonic Waveguide:- Silicon photonic waveguide is the building block of several photonic components and lays the foundation for integrating photonics with a Silicon substrate. They are the basis of many optical devices such as MRRs, Directional Couplers, Multiplexers and Demultiplexers, communication channel etc. It basically consists of a Si core having an extremely small cross-section sorrounded by SiO_2 cladding material. Due to the difference in refractive index of the core and the cladding the light travels being confined within the waveguide due to total internal reflection.

MRR:- Micro-ring resonator is one of the principal components of a photonic network-on-chip. As quoted in the Laser and Photoics Reviews[37], a generic ring resonator consists of an optical waveguide which is looped back onto itself such that a resonance occurs when the optical path length of the resonator is exactly a whole number of wavelengths. So in this way an MRR is capable of supporting multiple resonances with a gap equal to the free spectral range(FSR) which in turn depends on the resonator length. The practical application of MRR is always along with a coupler which is necessary for it to interact with the outside world. It is basically an accessing mechanism. When in the loop the round trip phase shift equals the integer times 2π , resonance occurs and the wave is coupled successfully to or from the MRR.

Silicon photonic Laser:- Laser sources can be of different types depending upon the need of transmission. In the proposed design we have used Mode-Locked Lasers. The important fact is that they are the major source of power consumption in a PNoC.

Photo-detector:- At the reciever end, the light passing through the silicon waveguide must be detected in order to reconvert the modulated data into electronic form. Photodetectors contain PIN photodiodes to convert optical power into electric current. Recently, graphene is said to be a promising material for ultra-broadband photodetectors[18].

Silicon Optical Modulator:- It is used to modulate a light beam propagating in the optical waveguide with electrical data signals. The most common method of achieving modulation in silicon devices so far has been to exploit the plasma dispersion effect, in which the concentration of free charges in silicon changes the real and imaginary parts of the refractive index[19]. Micro-Ring modulators has been widely explored due to its compact footprint and low drive voltage. The nature resonant frequency of a micro-ring can be shifted by an index change and thus a large modulation depth occurs near the resonance.

Optical coupler:- It is a challenge in silicon photonics to obtain efficient coupling between highly confined mode in a waveguide and the large diamter mode in optical fibre to couple light in and out of the chip. Recently several solutions like surface gratings and tapers are proposed in which the thickness and width of silicon layer are increased.

Silicon photonic transmitter:- The silicon photonic transmitter comprises of a laser source, an optical modulator that modulates the light signal with electric signal data and a multiplexer that multiplexes the signal light of multiple wavelengths or modes to be transmitted on a single transmission line.

Mode-Locked Laser:- Mode-locked-laser consists of an active laser resonator, and an optical mirror. Laser resonator produces ultra fast optical pulse circulating around it. Each time the pulse hits the optical mirror, a pulse is emitted out of the laser. As a result, an optical pulse train of

specific wavelength and time period is generated. This phenomenon is called mode-locking as all the modes are trapped inside the laser resonator as a single pulse and hence the term mode-locked-laser.

2.4.2 Multiplexing

As mentioned in the introduction, the advent of silicon photonics as an alternative to conventional CMOS chips was imperative owing to its superior bandwidth capability and lower power dissipation. In order to facilitate such a higher aggregate bandwidth several multiplexing techniques have been proposed. The goal of multiplexing is to boost the aggregrate throughput by utilising the existing communication infrastructure on a chip. Over the recent years, with the increase in technical finesse of fabrication technology, researchers have been able to exploit unique features of light to come up with several multiplexing techniques.

2.4.2.1 Wavelength-Division-Multiplexing

Wavelength-Division-Multiplexing(WDM) is the most commonly used technology to increase the bandwidth of the optical communication system. In this scheme multiple wavelengths are employed to carry optical signals from the source to destination. The data rate in optical communication is limited to the modulation speed but the overall bandwidth can be scaled with the no of wavelengths used in a system acting as parallel communication channels.

Dense WDM(DWDM) technology has been used to enable tens of channels with varying carrier wavelengths over single mode optical fibres in long optical networks. But due to its sensitivity to temperature, the emission wavelengths of the laser need to be aligned and stabilized properly. It makes the switching and routing in DWDM systems complicated, power hungry and expensive rendering it unsuitable for on-chip communication[38]. The alternative is coarse WDM(CWDM) which has less requirements in terms of alignment/control. A 4 -channel CWDM link with 400

GHz channel spacing has been used to realize 50 Gbps communication link between two chips[39].

2.4.2.2 Mode-Division-Multiplexing

A mode can be defined as an electromagnetic field distribution that satisfies the theoritical requirements for propagation in a waveguide or oscillation in a cavity or in other words an electtromagnetic wave travelling in a fiber. Exploiting spatial mode as an independent channel in conjuction with WDM would increase the bandwidth density of an-chip interconnect by manifolds, reduce the number of waveguide crossings and add an additional design degree of freedom in next generation photonic networks. Earlier there have been efforts of implementing mode multiplexing based on Mach-Zehnder interferometers[40][41], Multi-mode interference(MMI) couplers[42][43] etc. But they had larger footprints and supported a limited number of optical modes.

In 2014 Luo et al came up with a micro-ring resonator based on-chip WDM-compatible mode-division multiplexing (and demultiplexing scheme)[1]. They demonstrated the capability of the design to be multiple co-propagating 10Gb/s high-speed communication signals reaching upto 60 Gb/s of aggregate bandwidth.

On the basis of propagation constant matching, an optical mode in a single mode waveguide can be evanescently coupled to a specific spatial mode in an adjacent multimode waveguide, in which the coupling strength to the node depends on the width of the multimode waveguide[1]. The propagation constants of different spatial modes can vary significantly due to the high core-cladding index contrast. In Fig.2.1 taken from [1], the arrangement of the ring resonators and waveguides for facilitating mode multiplexing is shown.

The Ring resonators are formed from a 450-nm wide waveguide. They are designed to support only the fundamental TE mode with effective refractive index(R_{eff}) of 2.46. The multimodal transfer waveguide is tapered at several places. When the waveguide width corresponds to 450 nm, 930 nm, or 1.41μ, the effective indices of TE_0, TE_1 or TE_2 modes respectively, match the effective index of TE_0 mode of the microring to the TE_0, TE_1 or TE_2 modes in the multimode waveguide. By adjusting the coupling gap and coupler length between microrings and waveguides, the insertion loss for the desired mode and the power coupled to other modes can be minimized. All these

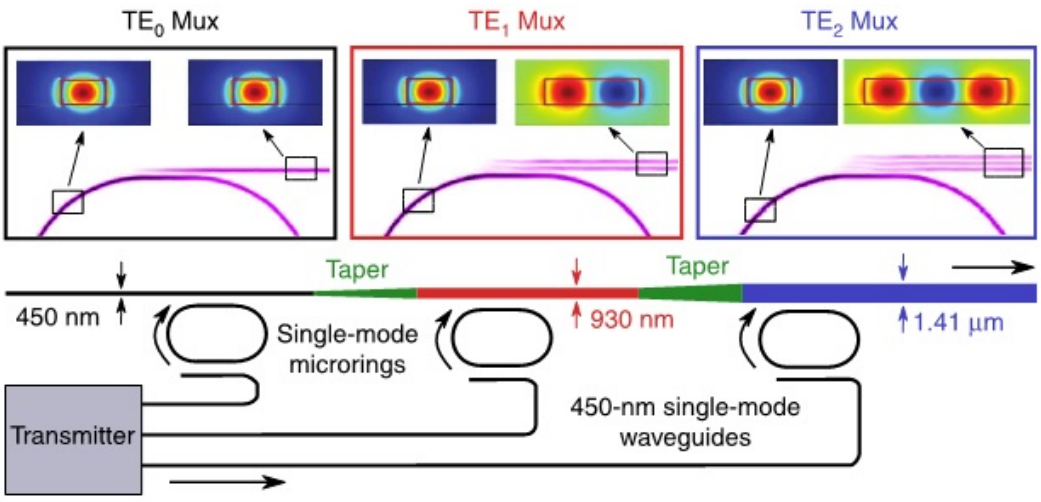


Figure 2.1: Selective coupling of the single-mode microrings to a specific spatial mode in multi-modal waveguide. *Reprinted with permission from* [1]

features along with an integrated heater on top of each microring to tune the Microring resonances to align to the WDM channels optimizes the performance of the whole device.

Looking at the bigger picture, each microring resonator can support 87 WDM channels over the entire C-band(1530-1565nm) keeping the channel spacing as 50GHz. Luo et al demonstrated that tapering the multimode-waveguide upto 2.37μ, 5 spatial modes can be supported. With the above setup, potentially an amazing 4.35Tbps aggregate data rate can be supported.

2.4.3 Proposed Adaptive Multiplexing

In our proposed design we adopt the aforementioned WDM-compatible MDM arrangement to design a high throughput and low power consuming photonic router. For our experiments we used two modes each of the two wavelengths we used for communication. The respective MRRs are turned on with the help of an electrical controller whose functional algorithm will be properly described in chapter 3.

2.5 Communication Flow

The process of optical communication starts with the laser source producing the light which is coupled with the waveguide on chip with the help of a silicon optical coupler. The Ring modulator

modulates the electronic signal from the core on to the appropriate channel. The multiplexer's job is to couple the modulated signal onto the desired wavelength and mode in the multimodal transmission fibre. Data is then transmitted along the waveguide till the destination. The propogation path can be changed or switched by appropriate photonic switches along the path made by the combination of MRRs and waveguides. At the destination the light signal is detected by the photo-detector and is demodulated to feed the information to the destination processor core. The whole process is depicted as a symbolic diagram in the following figure.

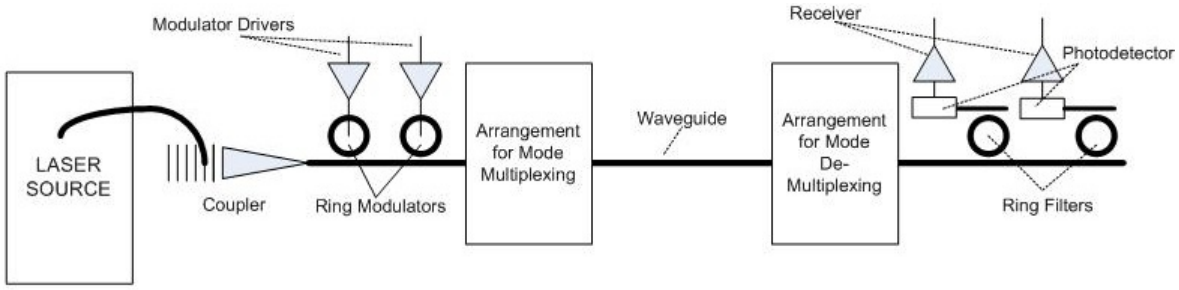


Figure 2.2: Communication flow in Silicon Photonics

2.6 Photonic Router

The proposed photonic router is a fully non-blocking 5×5 photonic router for NoC design. A photonic router in a PNoC consists of a number of I/O ports, a switching fabric connecting the input ports to the output ports, and injection/ejection ports connected to the local IP core via the network interface(NI). Fig.2.3 shows a 2D-mesh PNoC architecture consisting of five-port photonic routers. We zeroed on using a 2D mesh topolgy due to the same reasons that made them popular in electronic NoCs. Their appropriateness to handle a large variety of work-loads and their good layout compatibility with a tiled CMP chip[44] apply in photonic NoCs too.

2.6.1 Router Micro-architecture

The hybrid photonic router consists of mainly two circuits :-

1- Photonic circuit switching circuit 2- Electronic packet switching circuit The PNoC microarchi-

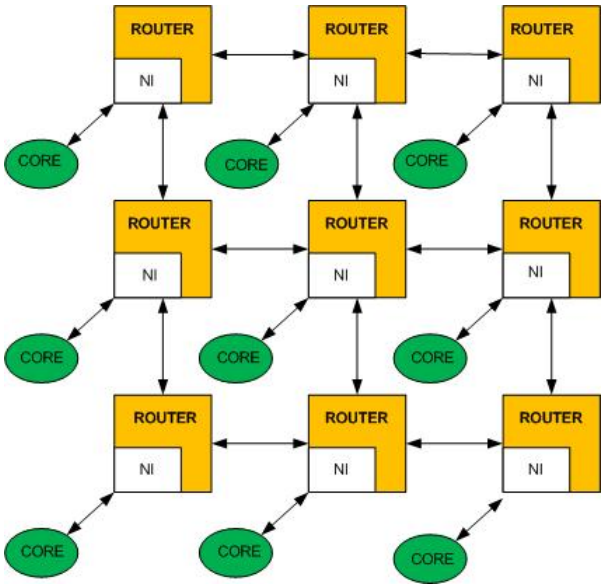


Figure 2.3: 2D PNoC architecture

tecture adopts a hybrid design. It combines a photonic switching fabric for circuit-switched bulk data transmission and an electronic packet-swicthed network for distributed control through *control packet* transmission. Hence the term 'hybrid' refers to the concept of combining electronic and photonic components as well as to the idea of combining a packet-switched network and a circuit-switched network.

This takes advantage of both the technologies i.e. photonic and electronic. Photonic technology provides superior advantages in terms of low power, large bandwidth and high speed communication. On the other hand, electronic control technology offers flexibility to adopt packet-switching. Packet switching requires buffering which is difficult to implement with photonic components. The main objective of employing a hybrid scheme is to address the higher power consumption in electronic NoCs, that scales up with the bandwidth[21].

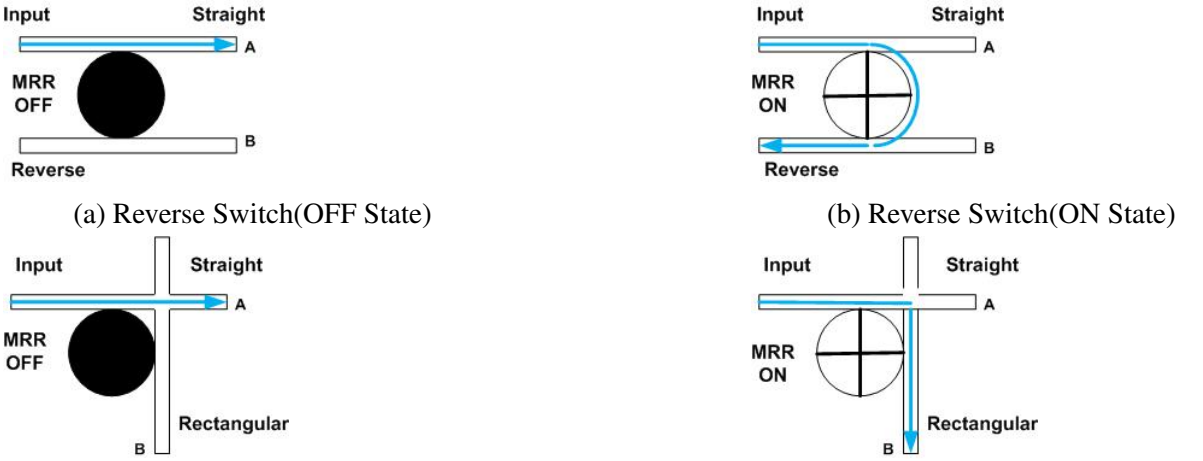
The photonic interconnection network consists of MRR and waveguide based routers and links to communicate large data packets. The electronic control network, comprising of an eletrconic controller integrated with each photonic router controls the operation of the photonic network. In this research we mainly focus on the photonic router micro-architecture rather than the electronic

controller. In the following subsections we discuss the detailed switching elements and router lay-out. We will go through the infrastructure responsible for boosting the performance of the photonic router.

2.6.1.1 Switching Fabric

A switching element in a fabric is composed of MRRs and waveguides as shown in Fig.2.4. The switching fabric consists of a set of parallel and rectangular 1×2 switching elements unlike 2×2 conventional electrical switches where each 1×2 switching element serves the purpose of parallel or orthogonal routing using fewer MRRs. We introduce in brief the working principles of 1×2 switching elements used in router micro-architecture.

Basic 1×2 Switching Element using MRR :-



(a) Reverse Switch(OFF State)

(b) Reverse Switch(ON State)

(c) Rectangular Switch(OFF State)

(d) Rectangular Switch(ON State)

Figure 2.4: MRR Switching. *Reprinted with permission from* [2]

The basic switching element of a photonic router is a micro-ring resonator. A MRR is a circularly coiled waveguide which has the property of rotating the optical signal in the clock-wise direction. During 'OFF' state of MRR, optical signal with wavelength λ_{on} propagates from the input port to the straight port(refer: Fig.3.8a and Fig.2.4c). When turned 'ON' it couples the resonating optical signal (indicated by arrow) in waveguide A and transmits it by coupling it to waveguide

B as shown in Fig.3.8b and Fig.2.4d. One can see that, in Fig.2.4d as the waveguide B is placed orthogonal to the waveguide A, the MRR helps in turning the optical signal in the rectangular direction. However it involves crossing of the two waveguides which may lead to loss due to cross-talk while passing multiple optical signals. It also results in higher insertion loss. To reverse the direction of propagation of the optical signal, one has to use combination of two rectangular switching circuits resulting in two waveguide crossing points. So, in order to decrease the number of crossing points of waveguides and usage of MRRs, a reverse parallel switching arrangement is made as shown in Fig.3.8b. In this arrangement when the MRR is 'ON', it helps in coupling the optical signal from waveguide A to waveguide B in reverse direction. This mechanism makes MRR an 1×2 switching element. With the help of these two types of switching arrangements, we were able to reduce the number of MRRs being used while reducing the no. of waveguide crossing points.

In the proposed scheme, we have incorporated WDM. Hence optical signal of multiple wavelengths can be coupled together through MRR. This will enhance the overall bandwidth of the network communication. The fundamental difference between the two basic 1×2 switches is the position of the two waveguides. The reverse switching element does not have any waveguide crossing unlike the rectangular-switching element. The insertion loss per waveguide crossing is 0.12dB[36]. MRR needs a DC current to switch ON and it consumes power less than 20μW [36]. In the OFF state, there is negligible power consumption by the MRR [35]. The switching time of the MRR is very small and it is 10ps in our casef.

Router layout :-

The 5X5 router layout which is adapted from [23], has been depicted in Fig.2.5. This uses suitable placement of 16 identical MRRs along with various waveguides.The router has five bi-directional ports, viz. East, West, North, South, and NI port. East, West, North, and South ports are connected with other routers to form a 2D NoC whereas the NI port is connected to the network interface. Each photonic router has a controller within NI for selecting wavelength and mode for optical signal transmission.The router can operate on multiple wavelengths simultaneously using WDM with

wavelength spacing equal to the free spectrum range of the MRR. As shown in Fig.2.5, there are optical paths for each of the input-output combination. Complex routing in a 2D photonic layer

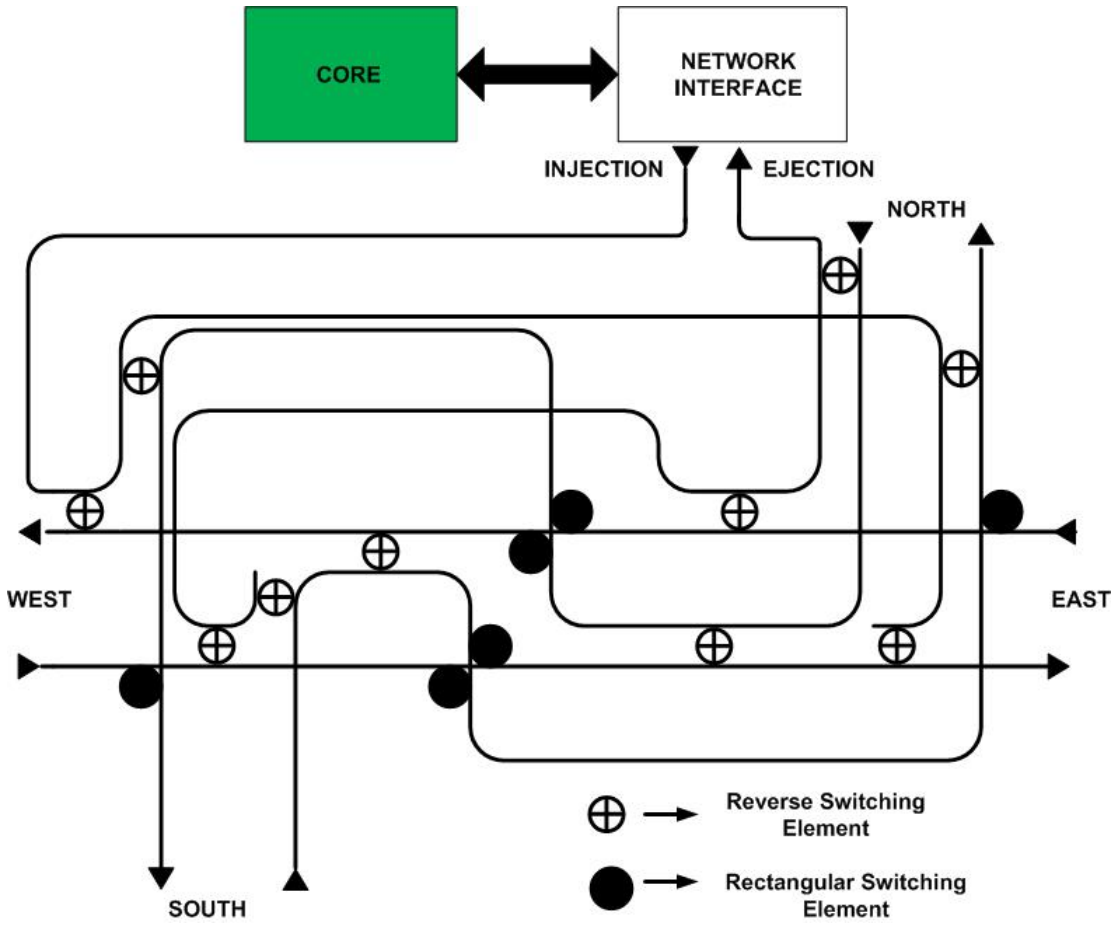


Figure 2.5: Logical layout of 5×5 photonic router. *Reprinted with permission from* [2]

is possible due to MRR based switches and waveguide crossings. However, waveguide crossing incurs optical insertion loss. Hence it's important to design an efficient layout of MRRs in a photonic router with least number of waveguide crossings. The proposed 5×5 non-blocking router consists of 14 waveguide crossings and 16 MRRs as in Fig.2.5 resulting in an optimized design. Apart from that, there are 2 more MRRs within network-interface to facilitate MDM which is explained in the later sections. Insertion loss and crosstalk limit the scalability of the photonic router [45][46]. Analytical results on number of MRRs and various performance parameters of photonic

router are discussed in detail in the later sections.

2.6.1.2 Network Interface

Use of WDM in photonic circuits offers limited performance gain because of its comaparatively higher power consumption [1]. MDM in conjuction with WDM and TDM can provide potentially larger performance gains and higher aggregate bandwidth [1]. Use of MDM technology does not require increasing the number of waveguides which leads to fewer wave-guide crossings. As mentioned in Mode-Division-Multiplexing in chapter 2 by tapering a 2.37μ multimodal waveguide upto 5 spatial modes can be supported. It also uses least area on chip and power. In this design the Network-interface [NI] is deployed with WDM compatible MDM (de)multiplexing technique which provides higher aggregate band-width.

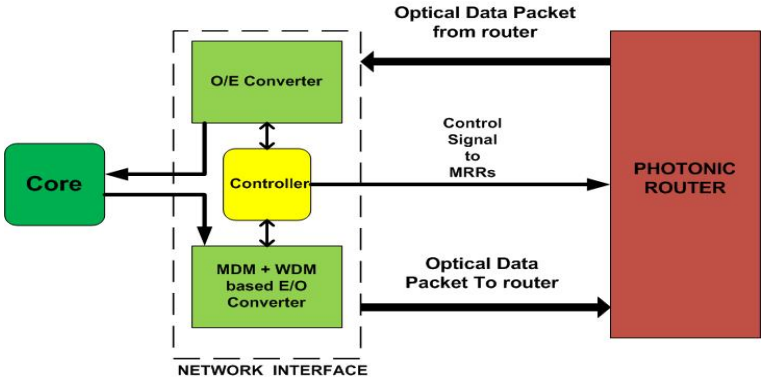


Figure 2.6: MDM integrated Network-Interface

The NI comprises of electrical to optical(E/O) and optical to electrical(O/E) converter, adaptive multiplexer and an electronic controller(Fig.2.6). The E/O converter and the O/E converter are simply ring based modulators and receivers. The Adaptive multiplexer includes the mode locked laser along with the MRR and waveguide arrangement facilitating (Fig.2.7) mode division multiplexing [1]. We have shown two MRRs for MDM as we are using two modes per wavelength in our experiments. As we have mentioned in the previous section the electronic controller controls the adaptive multiplexing too. It controls which MRR need to turn on in the MDM arrangement

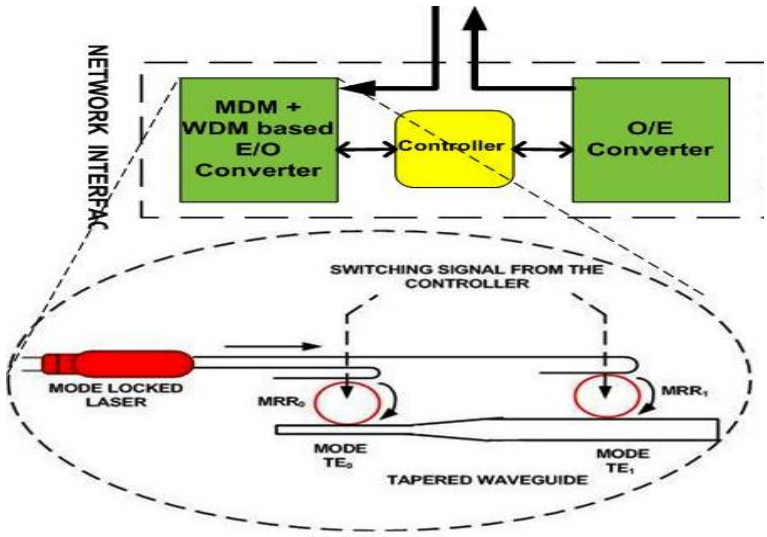


Figure 2.7: Modelocked laser employing MDM

to send the message along the preferred mode. The electrical controller works on the basis of algorithm depicted in Algorithm 1. The algorithm controls the wavelength and mode selection mechanism during message passing as follows. When the route to be followed is empty i.e it is not occupied by any other optical signal, the controller switches on MRR_0 which selects wavelength λ_0 and mode TE_0 transmitted by mode-locked laser for communication. When the route is occupied by certain mode of a specific wavelength then with the help of time-division-multiplexing a certain amount of time lag is incorporated while transmitting the next signal in a different mode so that it won't interfere in the transmission of the previous optical signal occupying the route.

Fig.2.8 illustrates how signals of different modes are transmitted with a time lag as determined by the algorithm, hence facilitating *mode-wavelength-time* division multiplexing. All the signals are carried by the same multimodal transmission waveguide with modulated signals coming from a common single mode fibre. So while selecting the appropriate combination of wavelength and mode it provides a time delay in turning on the corresponding multiplexing MRR. In our proposed design we have taken this time lag to be approx. 15ps as the switching delay of MRR is 5-10ps.

Algorithm 1 Controller Algorithm for adaptive mode division multiplexing

procedure CORE1 WANTS TO SEND DATA TO CORE2
N = no. of signals transmitting fully or partially along the path
Control packet checks for the shortest availale path using electrical routing
if (N=0) **then**
Switch ON MRR_0 with λ_0
Send signal from Mode-Locked Laser
if (N=1) and (mode = TE_0) **then**
Switch ON MRR_1 with λ_0
Send signal from Mode-Locked Laser with a (τ+pulse-width) delay
if (N=2) **then**
Switch ON MRR_0 with λ_1
Send signal from Mode-Locked Laser with a 2(τ+pulse-width) delay
if (N=3) and (mode = TE_0) **then**
Switch ON MRR_1 with λ_1
Send signal from Mode-Locked Laser with a 3(τ+pulse-width) delay

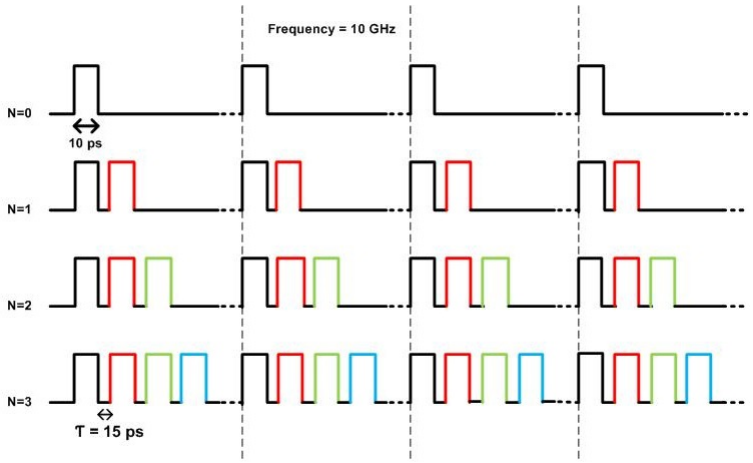


Figure 2.8: Black pulse=TE_0 of λ_0, red pulse=TE_1 of λ_0, green pulse=TE_0 of λ_1, blue pulse=TE_1 of λ_1
Timing diagram of TDM integrated Mode-Division-Multiplexing

2.7 Routing Algorithm

A packet switched 2D mesh electrical circuit acts as the control circuit for the photonic switching fabric. An electronic controller is the fundamental component of the 2D mesh electrical circuit.

There is one electronic controller corrsponding to each router in the photonic circuit. It has two major functions. The first is to set up the path channel from source to destination before the message is transmitted. Second is to configure the ring-resonators corresponding to the (de)modulator and (de)multiplexers to facilitate adaptive multiplexing. Photonic transmission networks are circuit switched networks, so a dedicated communication channel needs to be established before the actual communication takes place.

The electronic controller sets up the path between the source and the destnation nodes in the following way. As soon as the destination address of the message is known the source controller sends a small control packet initially to the destination. The control packet is routed according to the routing algorithm reserving the photonic path for the communication to follow. Among several routing algorithms we chose the adaptive X-Y routing algorithm as the standard routing algorithm for our experiments. Light always propagates in a straight line without any interference. In order to change the direction of propagation by 90° or 180° we need to turn on the MRR acting as switches as described previously. So we make sure that we transmit messages along a straight direction as much as possible without turns. It ensures lesser power consumption during coommunication. Hence, X-Y routing is the appropriate algorithm to be applied in a 2D mesh topology. When message is redirected along the shortest X-Y or Y-X path, it requires turning on only one MRR for a fraction of a second consuming less power during transmission.

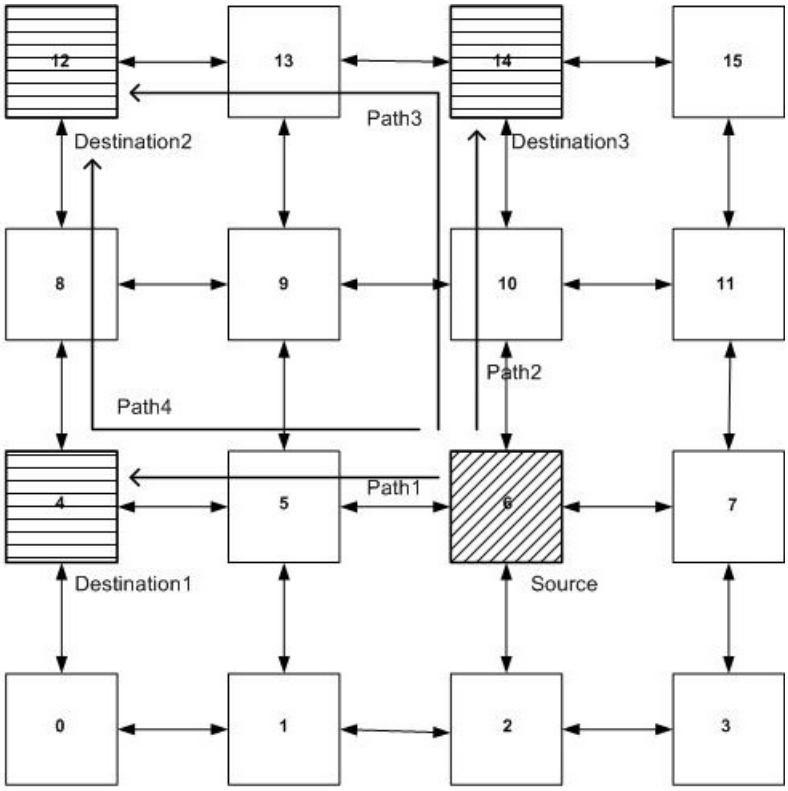


Figure 2.9: Illustrating adaptive X-Y routing between 'Source','Destination 1', 'Destination 2' and 'Destination 3'

Fig 2.9 represents a 4×4 2D mesh. We can see that in order to transmit data from Source(Node 6) to Destination1(Node 4) & Destination3(Node 14), the direction of propagation is along Path1 and Path2 respectively. The Algorithm states that if the destination node is along a straight line path from the source node(same x or y co-ordinate), then the controller must wait till any of the channel along the straight line path to be free before transmiting. For other destination nodes e.g Destination2(node 12) in Fig 2.9, Path3 or Path4 may be chosen according to availability as it involves turning on one MRR for bending during propagation. In the most unlikely case of unavailability of a mode channel along these two shortest paths, a random selection method is applied to select a path with two turns and so on.

Using this algorithm, if the path(in our case a certain mode of a certain wavelength) is successfully set up, an acknowledgement packet is sent back to the source. Upon receiving the acknowl-

edgement the message transmission begins. After the message tramsmission ends, a control packet is sent to free the reserved channel in order to be used by other messages. In case the path-setup packet is not able to set up the whole path and is dropped owing to congestion. Another packet can be sent in the reverse direction backtracking the path reserved by the prevous packet. It releases the reserved hops on the path and notifies the source controller to look for an alternative channel.

2.8 Laser Multiplexing in 2D & 3D Photonic Network-on-Chip

The perviously discussed PNoC uses on-chip lasers. Each core is served by one on-chip laser. However, Laser is found out to be a power hungry photonic component [25]. In fact laser power consumption is up to 50% of total network power consumption in PNoC [25]. This motivates us to design PNoC with reduced laser power. we propose a novel sandwich layered approach to design a 3D PNoC architecture that is able to reduce no of hops, cross over points, and no of laser sources using multiplexing techniques. The 3D hybrid PNoC uses our proposed high performance 5X5 photonic routers incorporating MDM along with WDM and TDM as explained in the previous section. As shown in Fig.2.10, the multiple electronic and photonic layers for building a 3D architecture are connected with each other by through-silicon-vias(TSVs). The proposed multilayer sandwiched architecture uses TSVs for communication between the laser layer and the network layers. TSVs have very small cross-sectional diameter(4um-10um) and extremely low delay(20 ps for a 20 layer 3D stack) [47]. Both the top and the bottom layers are network layers and the middle sandwiched layer is the Laser layer.

In the proposed multilayer design, each of the network layer consists of a 2D mesh network of 16 cores interconnected by 16, 5X5 non-blocking photonic routers as described in section 2.2. Apart from that therre are multimodal waveguides with two channels incorporating WDM and four 4X1 MUXs each dedicated to each of the 4 routers. The Laser layer consists of several waveguides which act as channels to relay the information from the source router to the destination router through the mode-locked laser in the same layer, selected by the corresponding 4X1 MUX in the network layer. Each laser serves four routers contrary to one laser per router as in almost all

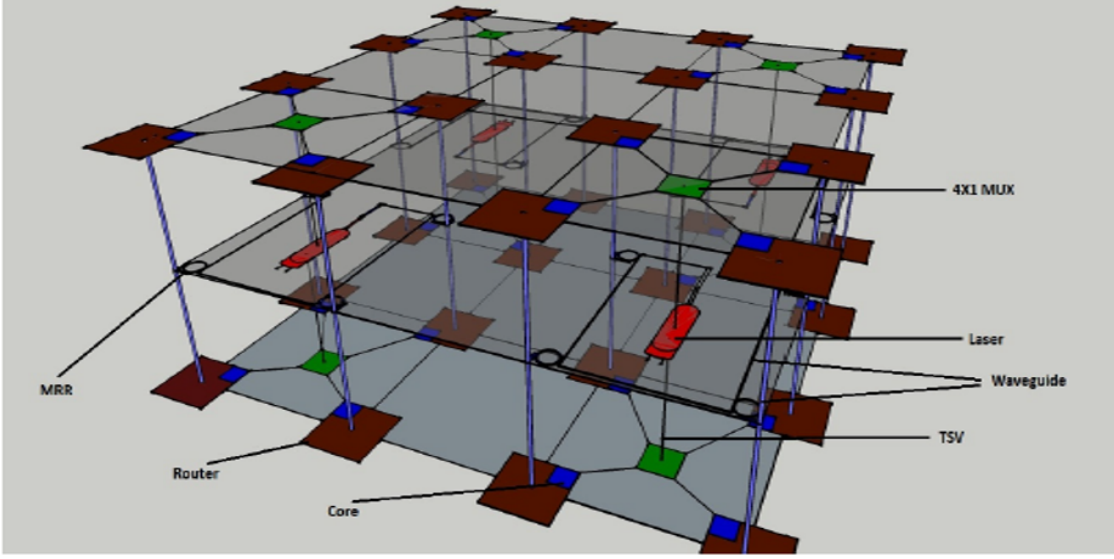


Figure 2.10: Multilayer PNoC with Laser Multiplexing. *Reprinted with permission from* [3]

Photonic NoC architectures recently proposed.

2.8.1 Mechanism and Control

If a source core 's' wants to send the message to the destination core 'd', it first sends a READY signal to the corresponding 4X1 multiplexer (MUX) to check its availability. As it follows the circuit switching mechanism so the route to be followed from source to destination is figured out as soon as the READY signal reaches the MUX. After receiving the acknowledgement from MUX, the core sends the message to the mode locked laser through MUX via TSV. The mode locked laser is controlled by the electronic controller present in the router. It adaptively selects the wavelength and the mode on which the message is to be relayed until the destination core. The controller works on the algorithm as depicted in Algorithm I. To relay messages from the top to the bottom layer and vice versa we have one MRR in the laser layer corresponding to each router which helps in tapping the message as it is relayed in the common waveguide channel from the corresponding mode locked laser. Fom the above arrangement we can see that with only four lasers we are able to serve 32 cores simultaneously.

2.9 Experiments & Results

2.9.1 Experimental Methodology

IPKISS [48] platform has been used for the design and simulation of the photonic router. The tool allows the photonic component layout design, virtual fabrication of components in different technologies, physical simulation of components, and optical circuit design and simulation. We custom-designed some photonic components such as MRR, waveguide, mode-locked laser, tapered waveguide, and photodiode required for the router. After checking the design rules, we proposed the design of Photonic NoC in this work. In order to validate the proposed micro-architecture we built a cycle accurate network simulator with required infrastructure to run both synthetic workloads and PARSEC benchmark traffic.

2.9.1.1 Microarchitecture Simulation on IPKISS

We optimized the desired parameters of the optical components and validated them prior to using them in the experiments. The design parameters adopted to carry out various experiments are depicted in TABLE 2.1. We restricted the width of the multimodal transmission waveguide such as to support two optical modes TE_0 and TE_1 along with multiple wavelengths. Though with varying width the wavwguide can support more no. of modes, but we're considering modes TE_0 and TE_1 as the proposed MDM scheme supports 2 modes (Fig.2.7).Using the fundamental optical components, we built the router in IPKISS and performed the physical simulation.

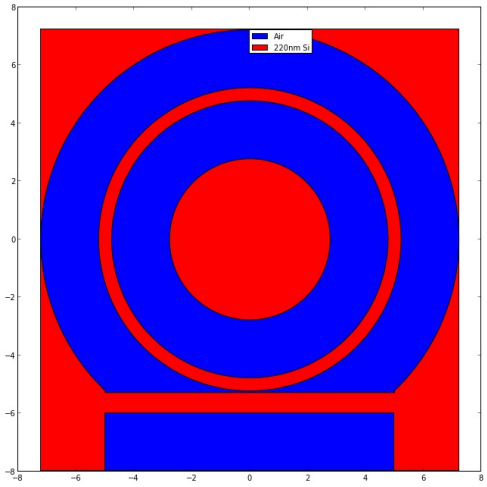
Table 2.1: Design parameters for experimental setup. *Reprinted with permission from* [2]

Design Parameters	Value
MRR diameter(D)	5μm
Waveguide(MRR) width	1.5μm
Photodetector Area(A_{det})	$20\mu m^2$
Refractive index Of waveguides	2.46
Pulse-width of Optical Signal	10ps
Frequency(mode-locked laser)	10GHz
Wavelength	1547.5nm and 1550nm

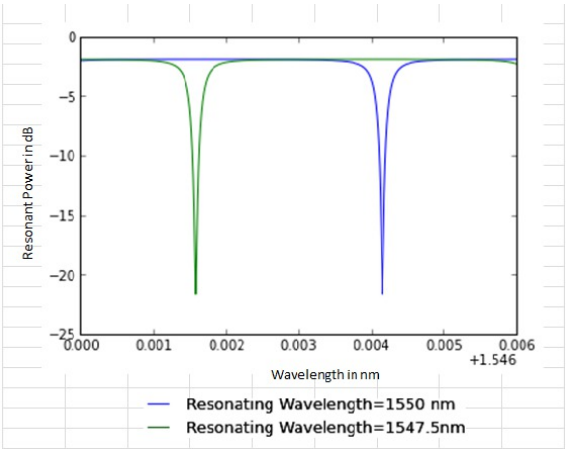
Virtual fabrication of various optical components e.g the MRR(Fig.2.11a) were done taking into consideration the design rules. MRRs and waveguides were integrated to virtually fabricate the router. After fabrication, we carried out simulation using CAMFR [48]. It provided the refractive index profile and optical transmission profile of the fabricated design. Uniform refractive index across the router is a must for ripple free photonic transmission. After testing the uniformity of refractive index across the router, we simulated the router in CAPHE [48]. CAPHE is an optical circuit simulator for time-domain and frequency-domain analysis. It is also used to evaluate insertion loss in an optical circuit. The insertion loss in the MRR was found to be 0.12dB. Each MRR can be tuned to multiple wavelengths. The proposed router takes into account the wavelengths of 1547.5nm and 1550nm(Fig.2.11b) for MRR switching as these are the most suitable wavelengths(in terms of performance) for silicon photonic waveguides [49].

2.9.2 Microarchitecture validation under traffic

In order to validate the proposed router microarchitecture we built a systemC based cycle accurate network simulator inspired from widely used Noxim simulator[50]. We implemented the MDM+WDM+TDM based photonic router along with circuit switching scheme in the simulator



(a) Reverse Switch(OFF State)



(b) Reverse Switch(ON State)

Figure 2.11: MRR Switching

called Photonoxim. We took 2D mesh as the network topology to study the behavior of the network architecture under various kinds of synthetic traffics. We also validated our micro-architecture under PARSEC benchmark traces. We fixed the PNoC chip frequency at 1GHz. We compared our circuit-switched proposed PNoC with packet-switched 45nm electronic NoC in a 2D mesh topology for same amount of simulation cycles under various kind of synthetic as well as benchmark traffics.

2.9.3 Comparitive Analysis and Results

A comparative analysis of the proposed router with the most recent designs such as the λ-router, crossbar router, cygnus router [23], and the columbian router [51] was carried out. We analyzed important parameters of all these routers like the required number of MRRs, optical insertion losses, and power consumptions. We adopted the information on crossbar router from [23]. 2D topology based NoCs generally require 5×5 routers. But λ-router does not support odd number of input and output ports. Hence we have considered a 6×6 λ-router in which one pair of input output ports remain idle.

2.9.3.1 *Number of MRRs*

The number of MRRs used in a router microarchitecture determines the area overhead of router. Reducing the number of MRRs will lower the chip area and also decrease the resultant energy consumption. This will eventually reduce the chip cost. We compared the number of MRRs used to design each of the photonic routers. The proposed router uses 18 MRRs [16 for routing and 2

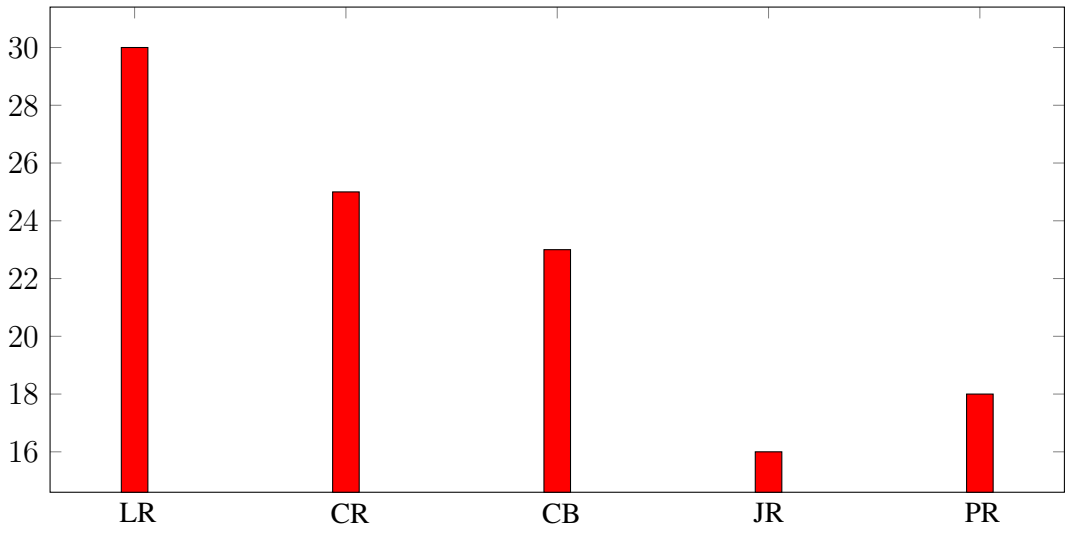


Figure 2.12: Comparing number of MRRs/Router for different PNoC architectures; LR=λ-Router CR=Columbian Router, CB=Crossbar Router, JR= Ji-router, PR=Proposed Router. *Reprinted with permission from* [2]

for MDM]. It is 40% lesser than the λ-router. Fig.2.12 represents the number of MRRs used in each of the routers. We have denoted router proposed in [33] as Ji-router for simplicity. The graph clearly shows that the proposed router outperforms all its counterparts except Ji-router in-terms of number of MRRs. Though Ji-router uses fewer number of MRRs than the proposed architecture, it is not scalable because of its higher insertion loss due to more no. of waveguide crossings and it does not support higher throughput performance.

2.9.3.2 Photonic area Overhead

There are several optical components involved in silicon photonics NoC. These includes modulators, waveguides, switches, filters and photodetectors. The area occupancy of a given photonic NoC can be calculated as the sum of the area of each of the individual component on the silicon die. For our analysis we assume the following :-

1- All the ring resonators are of the same size

2- The ring modulators in the transmitter are made up of one active ring resonator

3- The (de)multiplexers consist of two active ring resonators(as for our experiments we have taken two modes into consideration)

4- The reciever includes a passive ring resonator based filter and a photodetector.

We used the area overhead calculation formula proposed by Abadal et al in [52]. According to it the area of a given PNoC architecture given the previous assumptions can be approximated as:

$$A \approx N_{ring} A_{ring} + N_{det} A_{det} + \sum_{i} A_{wg,i} \tag{2.1}$$

Here, A represents area of chip occupied by photonic components, N_{ring} represents total number of MRRs, A_{ring} is the area of each MRR, N_{det} is the total number of photodetector, A_{det} represents area of each photodetectors, and A_{wg} is area of each waveguide. A_{ring} can be calulated using equation 2.2. We took the parameter values mentioned in Table.3.1 for calculating the area overhead.

$$A_{ring} = \frac{\pi D^2}{4} \tag{2.2}$$

Table 3.1 shows the design parameters used for this analysis. We have taken into account ATAC[53], CORONA[29], DCOF[54] and P-MESH[22] along with the proposed design for area-overhead analysis. Fig.2.13 shows the area footprint of each configuration evaluated as a function of the number of cores. It has been normalized to the area of a 400 mm^2 chip. From the graph, it is clearly evident that the proposed PNoC architecture is the most scalable in terms of area evolution

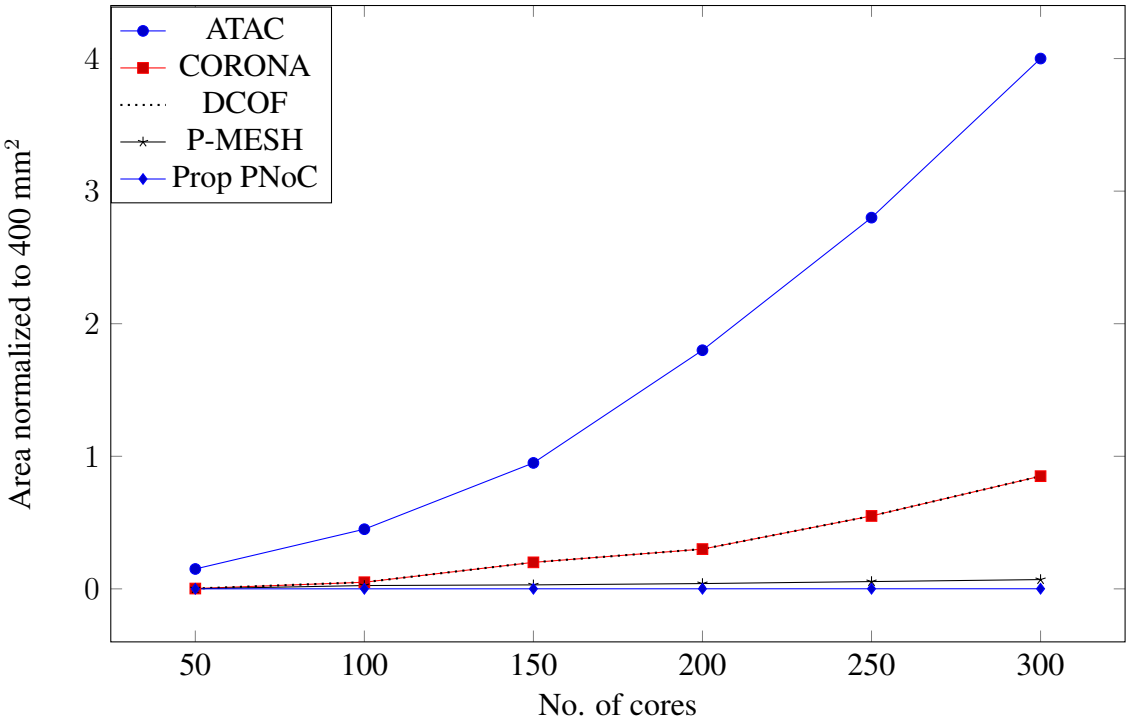


Figure 2.13: Area overhead of different PNoC architectures with varying sizes

as compared to all others. The number of modes which we can support in a multimode fibre for increasing the aggregate bandwidth depends on the width of the transmission waveguide. So with increase in the no of modes, the area overhead increases too, but we can always make a trade off between the area overhead and the performance we seek.

2.9.3.3 Average Throughput

Throughput is a parameter that measures the rate at which message traffic can be sent across a communication network. According to [55] it is defined as:-

$$Average \quad Throughput = \frac{(number \quad of \quad messages \quad transmitted) * (message \quad length)}{(number \quad of \quad IP \quad blocks) * (total \quad time)} \tag{2.3}$$

In order to validate the performance of the proposed PNoC against other NoC architectures, we simulated some synthetic traffics and PARSEC benchmark applications in Photonoxim.

Comparison between Electrical NoC and Proposed PNoC:-

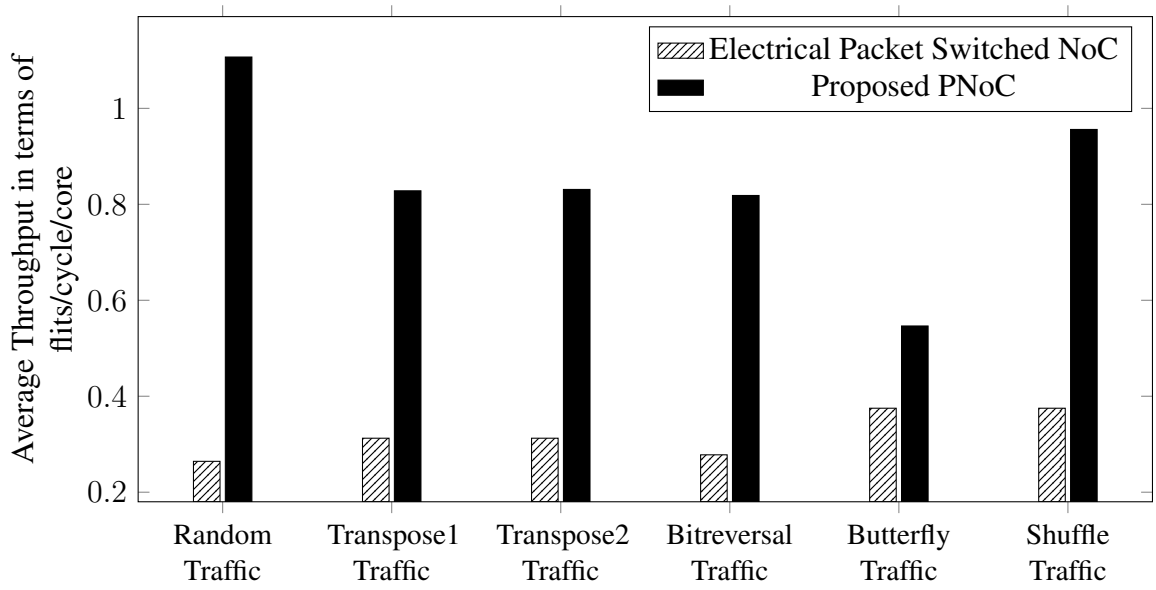


Figure 2.14: Comparison of Average Throughput between Electrical noC and Proposed PNoC with various synthetic traffics

As mentioned before for transmission of messages we adopt WDM+MDM+TDM based circuit switching photonic circuit. Since electrical signals are fundamentally limited with their bandwidth, a larger capacity can be achieved only by increasing no of parallel wires between the source and destination. With the help of mode division multiplexing in photonic circuits we can have multiple channels with the help of multiple modes and wavelengths. From the analysis it is quite clear that from the average throughput point of view electrical NoCs are easily trumped by Photonic circuits. We still ran some synthetic traffics both through a packet switching 45nm electrical NoC and the proposed PNoC. We kept the number of simulation cycles, packet injection rate and the no of cores same for both experiments.

Fig.2.14 represents the average-throughput/cycle/core for different types of synthetic traffics. From Fig.2.14 it is quite evident that the average throughput of proposed PNoC is about 4 times better than that of electrical NoCs. It is not only for the fact that there are more parallel channels(modes) but also that the average latency of the proposed PNoC is less than the traditional electrical NoC. For testing under some real world traffic we built the infrastructure in both NOXIM and PHOTONOXIM to run some PARSEC benchmark applications. Fig.2.15 reiterates the fact that

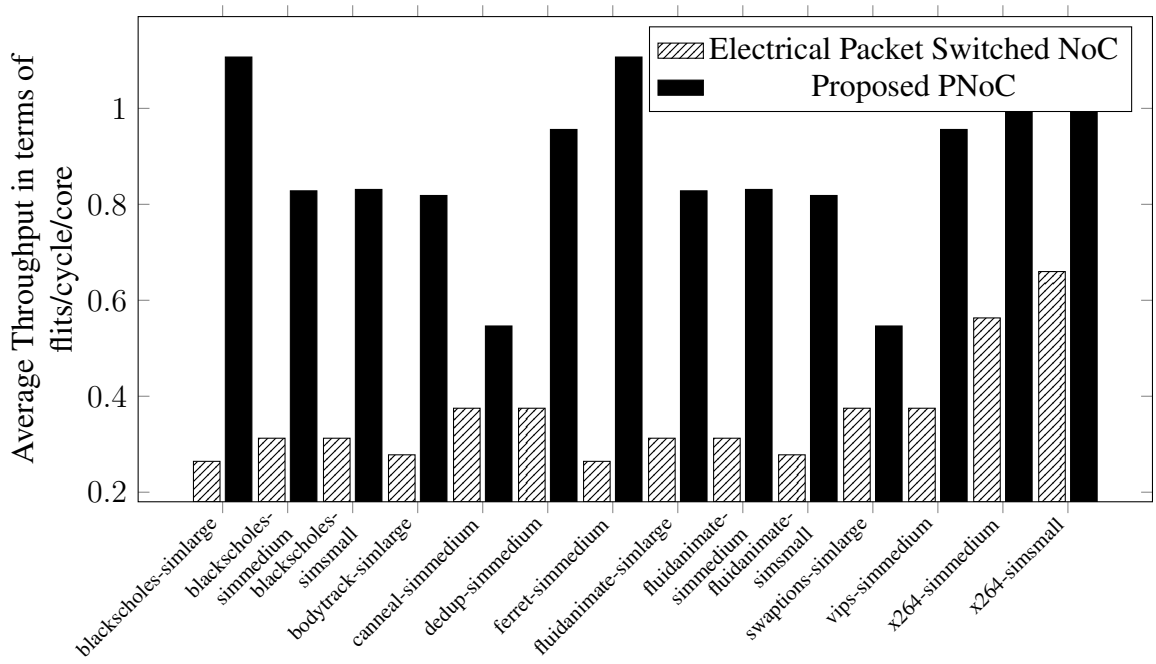


Figure 2.15: Comparison of Average Throughput between Electrical noC and Proposed PNoC with PARSEC benchmark

we inferred from running synthetic traffics earlier. This proves Photonic NoCs to be the biggest contenders in replacing electrical NoCs for high throughput on-chip communication.

Throughput Comparison of proposed PNoC architecture vs other PNoC architectures:-
Several PNoC architectures have been proposed in the recent past[22][27][26]. But none of them have demonstrated detailed simulation results except [56].

The authors in [56] have shown significant improvements over other recently reported PNoC architectures like CORONA[29], Firefly[30], [22]. Because of its detailed description on simulation and analysis, we are taking into account [56] for a fair comparison. In [56] uses WDM technology with a WDM degree[1] of 32 and provides results for a 8×8 CMP. As Meteor is based on crossbar architecture which is different from the proposed PNoC, we need to consider a different throughput metric for a fair comparison. Crossbar architectures in general use more number of photonic components (ref: Table 2.2) in order to provide higher bandwidth. But larger number of

photonic components leads to larger area over head.

Table 2.2: Microring resonator requirement for a 8X8 CMP

Component	Meteor	Corona	Firefly	Optical Mesh	Proposed PNoC
Transmission	8192	262144	32768	294912	1408
Reservation	1024	2048	1024	6400	0
Arbitration	1024	2048	1024	6400	0
Clock	4	64	16	64	64
Total	10244	266208	34832	307776	1472

To have a fair analysis, we introduce two new metrices namely 1) Throughput/photonic-area and 2) bandwidth/photonic-area.

Throughput/photonic-area comparison:-

We used normalized-throughput/photonic area-overhead as the metric for comparing the proposed PNoC with Meteor architecture where,

$$Throughput = \frac{Actual \quad throughput}{Ideal \quad Throughput * PhotonicArea} \tag{2.4}$$

From Table.2.2(*Courtesy*:-[56]), we can get the MRR requirement of various 8×8 core photonic architectures. We calculated the area overhead of different PNoC architectures from the MRR requirement information as mentioned in Table.2.2 and normalised it for the plot to fit in. We implemented some synthetic traffics and PARSEC benchmark traffics to evaluate the proposed architecture against Meteor. From Fig.2.16 and Fig.2.17, it is clearly evident that the proposed PNoC trumps Meteor in almost all the cases. The variations in throughput for various traffics are a bit different depending on the type of application and traffic. But in each case the proposed PNoC fares better than the Meteor PNoC.

[1]No of wavelengths acting as channel

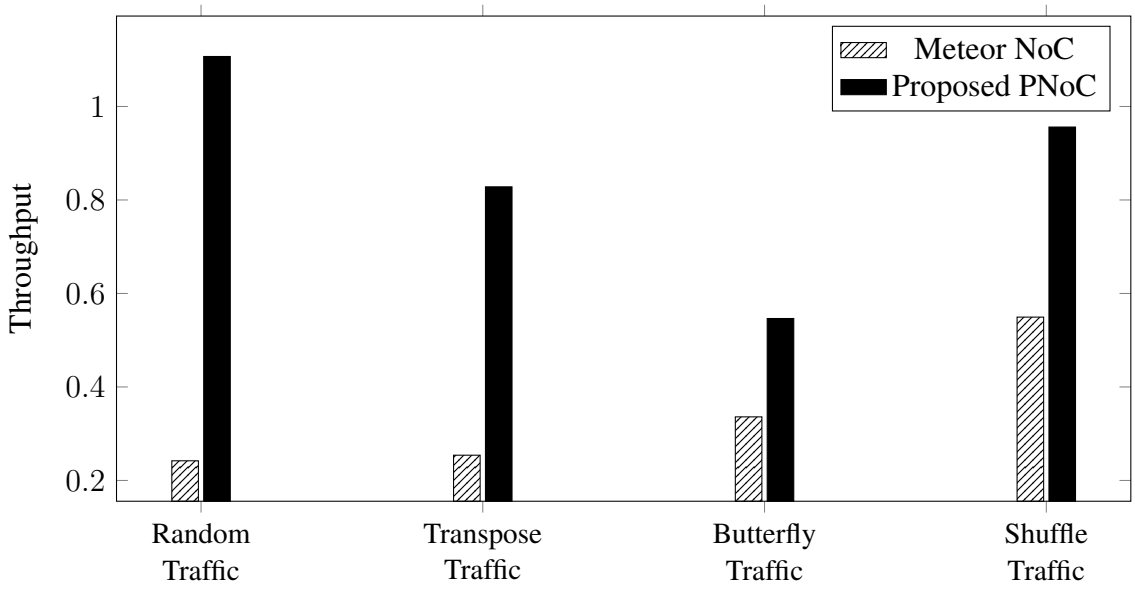


Figure 2.16: Throughput(Synthetic Traffic): METEOR NoC vs Proposed PNoC

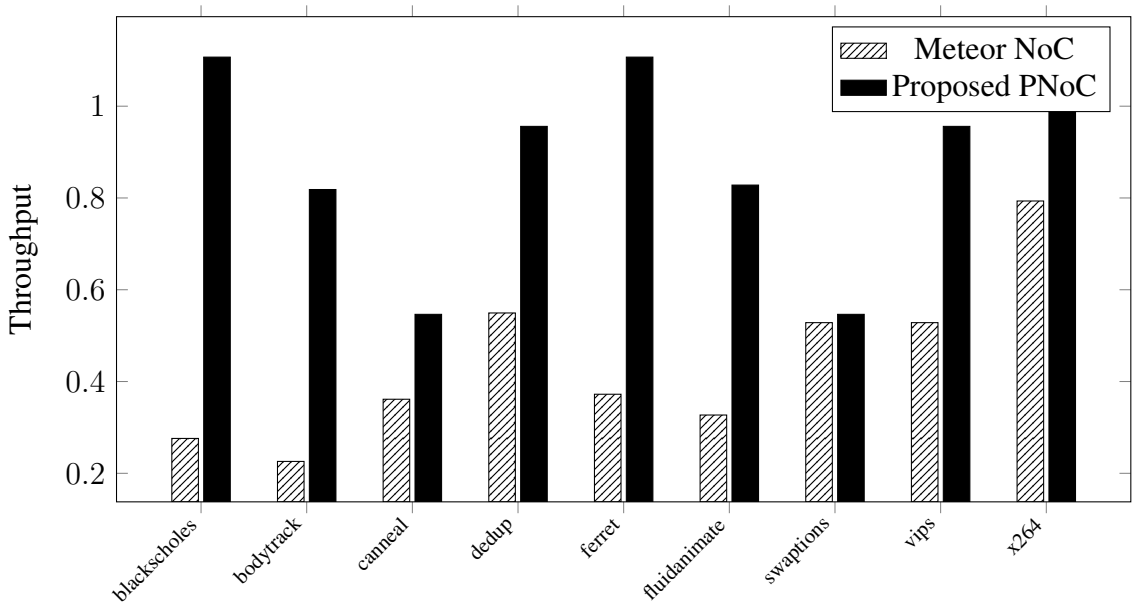


Figure 2.17: Throughput(PARSEC Benchmark): METEOR NoC vs Proposed PNoC

Bandwidth/Photonic-area comparison:-

The comparison metric bandwidth/area-overhead quantifies the performance in terms of design efficiency. The crossbar architecture performance can be boosted with an increase in the no. of photonic components. But a high performance scalable architecture is necessary to cater to the needs of future generation CMPs. The proposed PNoC architecture can provide multiple transmission channels within the same multimodal waveguide facilitating higher bandwidth with a smaller footprint as compared to other proposed PNoC architectures.

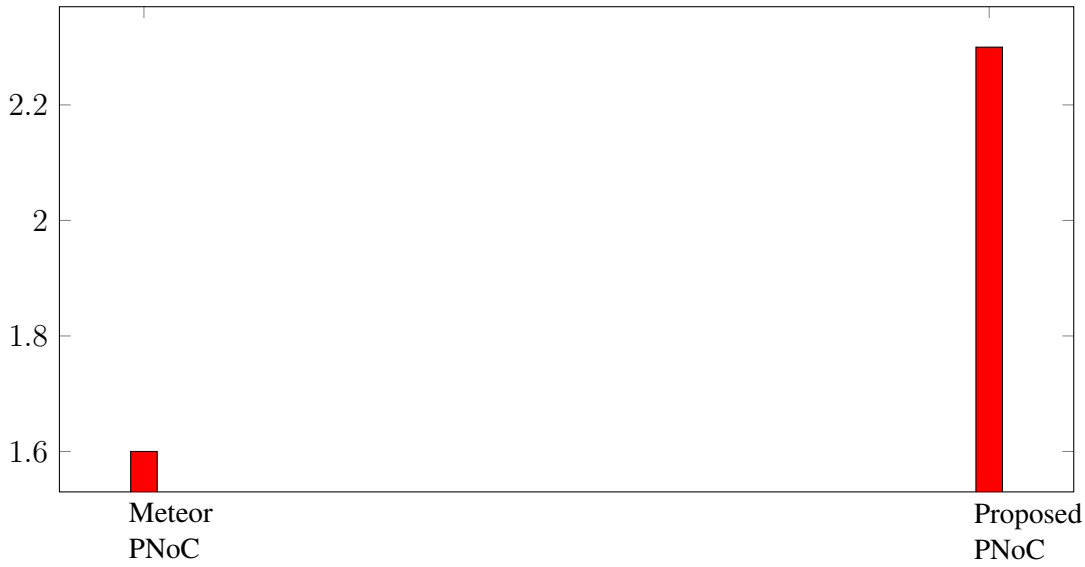


Figure 2.18: Bandwidth/Photonic area(Meteor PNoC vs Proposed PNoC)

2.9.3.4 Energy Consumption

The total energy consumption in a PNoC architecture is the sum of power consumed by the electrical controller and the power dissipated in the photonic switching fabric. The total power can be expressed as follows:

$$E_{PNoC-total} = E_{Electrical} + E_{Optical} \tag{2.5}$$

The total optical energy over a full network can be governed by Equation 2.6.

$$E_{Optical} = E_{Laser} + \sum_{j=1}^{N}[(E_{E/O_i} + E_{O/E_j}) + K_i \times E_M] \tag{2.6}$$

Here, E_{Laser} is the total laser energy consumption over the full network cycles, K_i denotes total number of MRR switched on for packet 'i', and E_M represents energy consumed to swicth on an MRR. E_{Laser} and E_M are given by Equations 2.7 and 2.8 respectively.

$$E_{Laser} = \sum_{j=1}^{N} P_{Laser_j} \times T_j \tag{2.7}$$

$$E_M = P_M \times T_M \tag{2.8}$$

T_j in Equation 2.7 refers to total ON time of $Laser_j$ whereas T_M represents time to switch on an MRR.

In order to compare different photonic routers we need to compare the energy consumption per bit of communication. Followings are the assumption to determine the total power consumption :-
1) We take E/O and O/E conversion time as 50 ps. Though we can take a lesser time but they will be bottlenecked by the electrical components of the control circuit. So we took a longer time duration in order to compensate for that.
2) ON duration for Mode (de)multiplexing MRR: 20ps
3) ON duration for Switching MRR: 20ps
4) Energy consumption in E/O and O/E : 2*[50ps * 20μ W] = 2.0 fJ
5) One MRR turning on each for multiplexing and demultiplexing: 2* [20ps * 20μ W] = 0.8 fJ
6) One MRR switching in XY: 20ps * 20μ W = 0.4 fJ
7) Total energy consumption in communicating one bit of data: 2.0+0.4+0.4=3.2 fJ

Laser is the most significant power consuming unit in a PNoC architecture. While ensuring the power of the laser we have to take into consideration all the losses that an optical signal incurs

while propagating from source to the destination. We have to ensure a reasonable strength of the signal at the receiver end after signal degradation. The losses include on-chip coupling loss, MRR coupling loss and pass loss and waveguide bending and propagation loss. The various loss values and the photodetector sensitivity is given in Table 2.3. We won't be considering waveguide propagation loss as it is negligible.

Table 2.3: Optical losses. *Reprinted with permission from* [2]

Parameters	Value	Units
MRR drop loss	1	dB
MRR pass loss	0.01	dB
Waveguide bending loss	0.15	dB/bend
(De)modulation loss	3	dB
On-chip coupling loss	1	dB
Photodetector sensitivity	-30	dBm
Laser efficiency	8	%

$$Loss\ in\ dB = 10 \log \frac{P_{in}}{P_{out}} \tag{2.9}$$

$$P_{in} = P_{out} * 10^{\frac{Loss\ in\ dB}{10}} \tag{2.10}$$

For the whole trasmission path, in Equation.2.9 P_{in} is the laser power and P_{out} is the receiver power. The photodetector sensitivity is -30 dBm = 1μ W. From the router layout, in the worst case scenario on an optical path there will be 16 waveguide bendings, 3 drop MRRs, 42 pass MRRs(for a 8×8 network), one MRR each for modulation and demodulation. After calculations and taking into consideration the worst case scenario, in a 8×8 mesh network the laser power needed to trans-

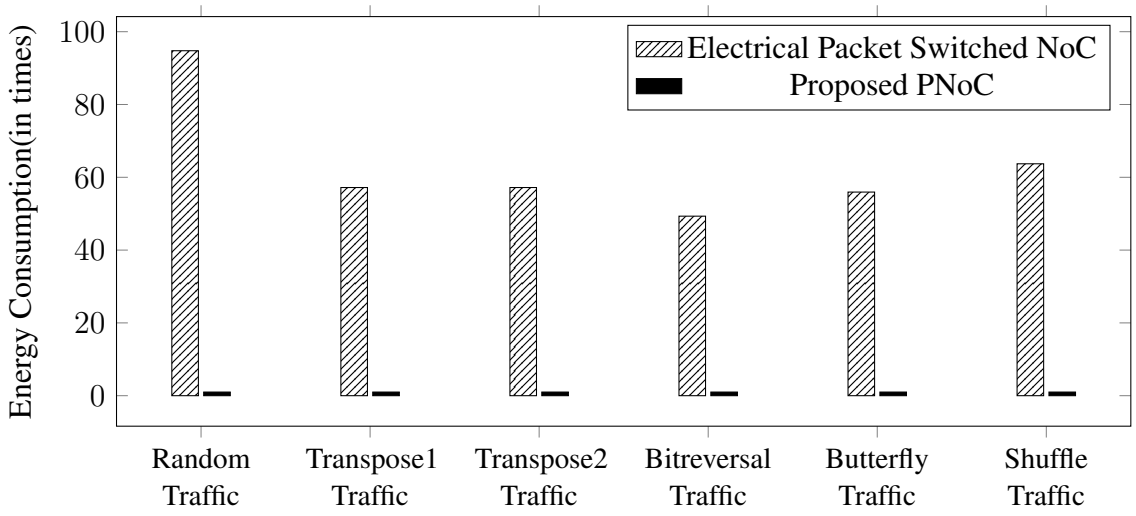


Figure 2.19: Energy Consumption(Synthetic Traffic): Electrical noC vs Proposed PNoC

mit a signal from source to destination comes to 239.2875 μ W.

Enercy Consumption Comparison of Electrical NoC vs Proposed PNoC :

The electrical global control and data wires consume significant power during signal communication and arbitration. In comparison to components used in electrical NoC, optical components consume significantly less power. As we haven't taken into consideration the electrical arbitration power consumption in our proposed PNoC, it was justified not to let consider the arbitration power consumed in an electrical NoC while comparing the simulation results. We only measured the energy consumed during communication of the message packets. With a fixed packet injection rate and simulation cycles, we ran several synthetic as well as PARSEC benchmark application traffic through both electrical NoC and proposed PNoC. We plot the power consumed by the electrical NoC normal to the power consumed in the proposed PNoC for all the traffics. From Fig.2.19 and Fig.2.20 , it is clearly evident that the energy consumed in the proposed PNoC is about 55X-65X less than the electrical NoC under most of the traffic conditions.

Energy Consumption Comparison of proposed PNoC architecture vs other PNoC architectures:-

The other reported works on photonic router [23][33][51][34] lack details on electrical controller

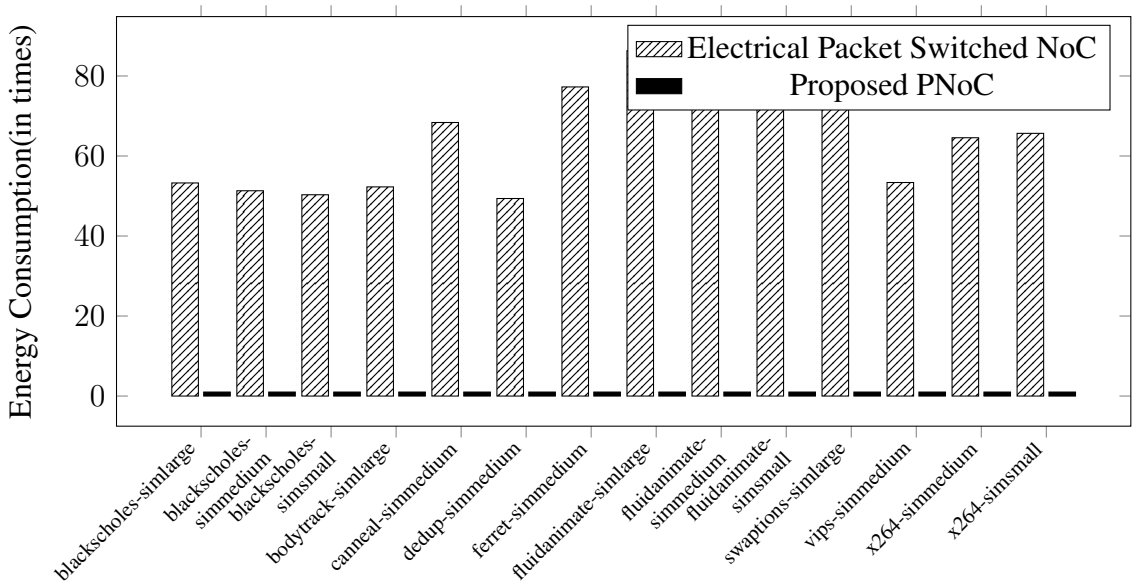


Figure 2.20: Normalized Energy Consumption(PARSEC Benchmark): Electrical noC vs Proposed PNoC

power analysis. Hence we have compared only the power consumption across the photonic switching fabric for a fair analysis. In the following, P_{router} represents power consumed by the photonic switching fabric. We are not considering the λ-router in the power analysis due to unavailablity of data provided in the literature.

We analyzed the three other routers [23][33][51][34] as a part of 2D network and studied their impacts at network level. We analytically determined the energy consumption of the stated three photonic routers in a 8×8 2D mesh NoC using dimension order routing scheme. We evaluated the average energy consumption per optical path in the network [E_{path}] and also the average energy consumption per router [E_{router}]. We calculated E_{path} using equation(1). Here E_j represents the energy consumed on j-th path when the bandwidth is 'B'. 'P' represents the total number of photonic paths in the NoC. E_{router} is calculated using equation(2) where 'R' is the average number of photonic routers in all the optical paths.

$$E_{path} = \frac{\sum_{j=1}^{P} E_j}{P \times B} \tag{2.11}$$

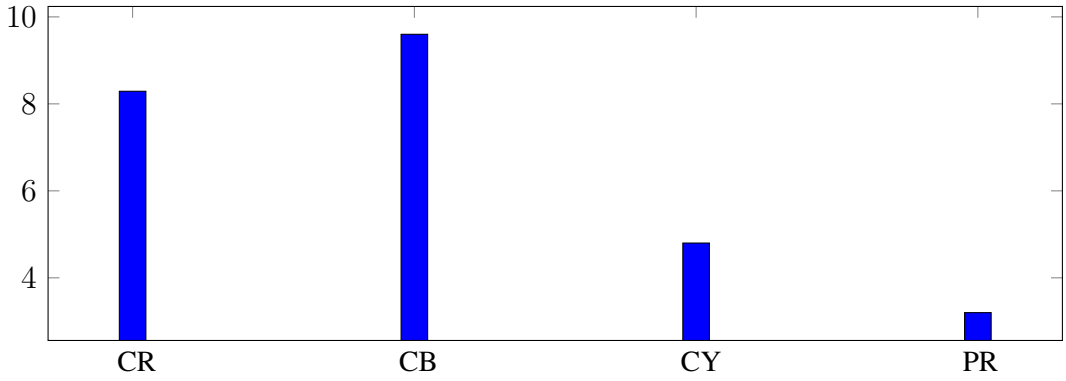


Figure 2.21: Comparison of average energy consumption per optical path, in fJ/bit. *Reprinted with permission from* [2]

$$E_{router} = \frac{E_{path}}{R} \tag{2.12}$$

Network-level analysis shows that the proposed router consumes the lowest average energy per optical path, 3.2fJ/bit. It is 66.67% less than the crossbar router, 61.3% less than Columbian router and 33.3% less than Cygnus router as shown in Fig.2.21. According to network level analysis in[23], in the proposed router based 8×8 2D mesh network, the average no. of routers in an optical path is 6.315. Then from equation Eq.2.12 the average router energy consumption comes to be 0.51fJ/bit, which is 61.1% less than the crossbar router, 66.45% less than columbian router and 32.9% less than the cygnus router as shown in Fig.2.22. A deeper analysis shows that the maximum energy consumption of the proposed router is also 3.2fJ/bit irrespective of network size. As with dimension order routing the maximum no of MRRs to be turned on in order to transmit messsages over a mesh network is closer to one. So the power consumption remains almost constant.

It was difficult to compare the simulation results of proposed PNoC architecture under synthetic and real benchmark traffics with other PNoC architectures such as Meteor and Corona due to lack of required information. As Meteor and Corona utilize electrical NoC both for local tranfers and arbitration and the task distribution percentage is not mentioned in the corresponding papers for different traffics, it was difficult to figure out the normalization criteria and the actual energy consumption in photonic communication in each case.

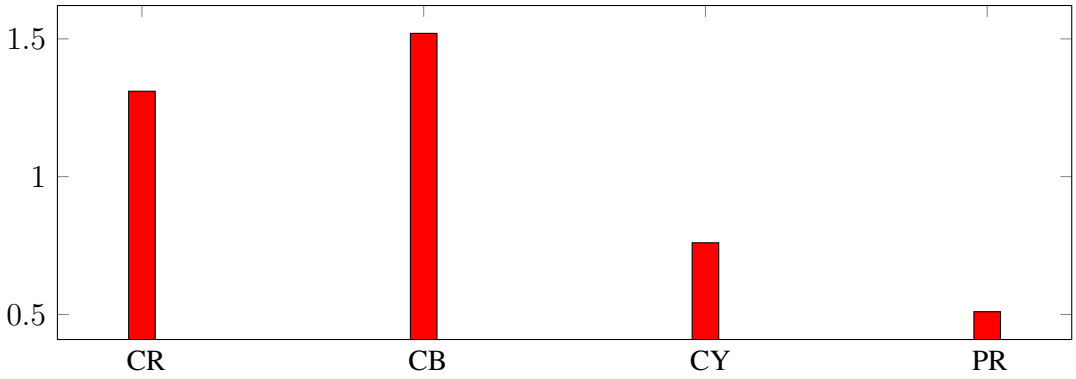


Figure 2.22: Comparison of energy consumption per router in fJ/bit. *Reprinted with permission from* [2]

The maximum power consumption of the proposed router based network on a given optical path is constant while using dimension order routing, regardless of the network size. The placement of the MRRs in the router takes care of the fact that no MRR is turned ON to transmit a message along a straight line. The router needs to switch on one MRR when a message enters the network from the NI, turns from a row to a column (and vice-versa), or exits the network to the NI. In the worst case, three MRRs need to be powered on to route a message in a network irrespective of the network size. This makes the proposed PNoC highly scalable without worrying about the power consumption in additional routers on a longer path. Fig.2.23 presents the average power consumption per optical path in NoC of different sizes.

2.9.3.5 Optical Insertion loss

Insertion loss plays a significant role in the amount of power consumption in a PNoC. Three major factors which contribute towards insertion loss are :-

1- Propagation loss- Signal power loss during propagation through waveguide.

2- Coupling Loss- It occurs during coupling of signals to and from MRRs.

3- Loss occuring due to waveguide crossing and bending.

Insertion loss of a router determines its feasibility and also the power required by the NI to transmit, and receive optical singals. We took MRR coupling loss equals 0.5dB[57]. A single

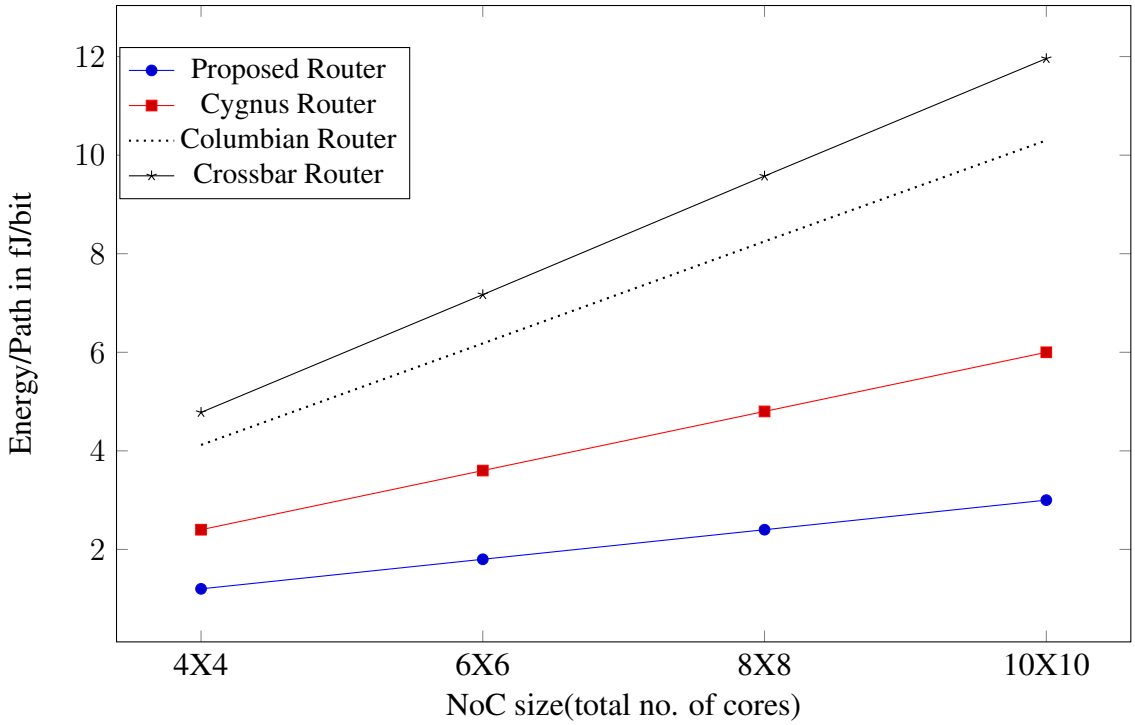


Figure 2.23: Average energy per Path for different PNoC sizes. *Reprinted with permission from* [2]

waveguide crossing introduces an insertion loss of 0.12dB[23]. The waveguide propagation loss is 0.17dB/mm. As the hop length is of the order of μ m, so it is negligible as compared to other losses.

Insertion loss varies across input output pairs in a router. Hence we evaluated the best-case, average-case, and the worst-case insertion losses in all the routers. The results are shown in Fig. 2.24. It is clear from the figure that the proposed router has lowest insertion loss for all other cases except Cygnus router for which the insertion loss is same due to similar router lay-out. Compared to crossbar router, the proposed router has 61% lesser best-case loss, 43% lesser average-case loss, and 28% lesser worst-case loss. As the layout is based on the cygnus router layout, so the insertion loss for both the roters are same. In all the routers mentioned only loss during communication is taken into consideration. Losses are also encountered while modulating and multiplexing too. In mode division multiplexing the main cause of signal degradation is crosstalk.

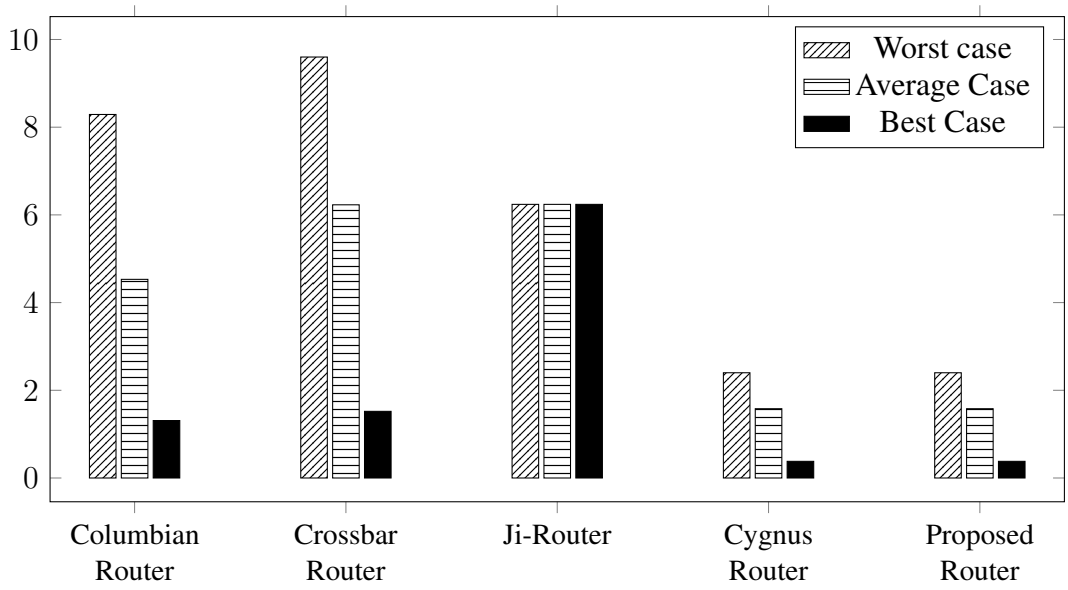


Figure 2.24: Insertion loss per router in dB. *Reprinted with permission from* [2]

To reduce crosstalk we have to increase the coupling gap between multiplexing MRRs and transmission waveguide but it is done on the expense of performance. According to [1], it accounts for a very low(<1.4dB) power penalty.

2.10 Chapter Summary

The rising density of cores in the chip-multiprocessors era has reached a stage where widespread adoption of high performance PNoC is inevitable to facilitate the large demand for communication band-widthwith with low power consumption. The thesis implements a novel scalable, low-power, WDM- compatible mode division multiplexed, 5×5 non-blocking high performance MRR based photonic router.

We proposed a hybrid PNoC architecture which comprises of:-

1- a circuit switching photonic circuit for bulk data transmission

2- a packet switching based electronic control circuit for path set-up, (de)modulation and (de)multiplexing.

We designed a cycle accurate circuit switching simulator PHOTONOXIM, to validate the performance of the proposed PNoC under various traffic patterns. This chapter broadens the field of Silicon Photonic NoCs by introducing MDM along with WDM and TDM for the first time to in-

crease the bandwidth about four times than the previous best reported results. By using minimal resources leading to a smaller footprint, it gives an insight into what the future NoC technologies have in store. The proposed design overpowers the traditional 45nm electrical NoC in almost every respect by providing a 4× boost in terms of bandwidth and a 55×-60× less power consumption depending on the traffic. As compared to other Photonic routers, the proposed router provides almost 3 times higher throughput. The proposed hybrid router consumes 33.3% less energy per optical path and 32.9% less energy per router than the best reported results under uniform network traffic.

Laser is the most power hungry component in a PNoC. We proposed a laser-multiplexing scheme in a 3D PNoC approach to further enhance the energy savings.

As silicon photonic components are susceptible to temperature change, further research is required to study the behavior of the PNoC under various traffics with respect to temperature variations on chip. The next chapter introduces a novel scheme to address this challenge.

3. CROSS-LAYER DYNAMIC THERMAL MANAGEMENT IN PNOC*

3.1 Why Thermal Management?

[1]Microring resonators (MRRs) and waveguides are the basic building blocks of a PNoC. MRRs are used as modulators/demodulators at the source/destination node. MRRs also perform photonic switching operations to route an optical signal in PNoCs. However, photonic components and especially MRRs are extremely susceptible to thermal fluctuations. Fig.3.1 depicts the impact of thermal variation on MRRs. MRRs R_1-R_n have been designed to resonate on wavelengthsn λ_1-λ_n respectively at temperature T1. As the temperature increases, due to the resulting variations in refractive index, each MRR now resonates with a different wavelength towards the red end of the visible spectrum (i.e., red-shift). This red-shift is shown in the figure where, at temperature T2, MRR R_i will now be in resonance with λ_{i-1}. This phenomenon reduces transmission reliability and results in wastage of available bandwidth, e.g., MRRs are unable to read or write to wavelength λ_n at temperature T2.

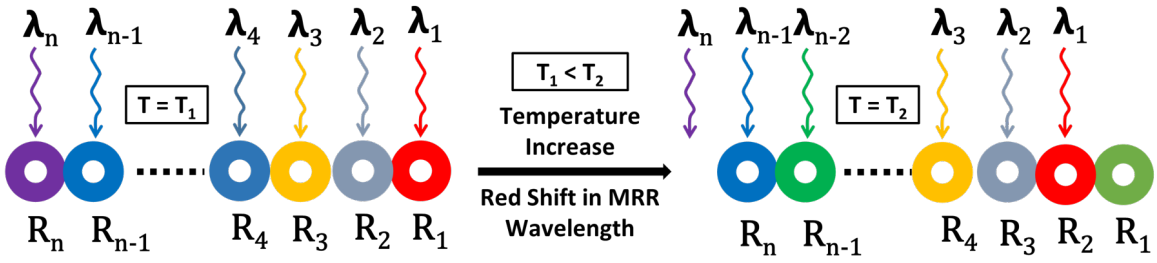


Figure 3.1: Impact of thermal variations on MRRs. *Reprinted with permission from* [4]

Maintaining a uniform temperature across all the MRRs is a must for reliable data transmission in PNoCs. But thermal fluctuations and gradients are common in CMPs. 3D-ICE[58] simulations of PARSEC[5] and SPLASH-2[6] benchmarks indicate a 15-20K peak thermal gradient in a 64-

[1]Adapted with permission from [7] & [4]

core CMP as shown in Fig.3.2. Such a huge gradient causes a mismatch of resonant wavelengths of MRRs, leading to unreliable data transmission and PNoC performance degradation.

3.2 Related Work

Recently, few techniques have been proposed to address thermal issues in PNoCs. At the device-level, a trimming mechanism is proposed in [59] that induces a blue shift (decrease) in the resonance wavelengths of MRRs using carrier injection. A tuning technique was demonstrated in[60] where a redgshift (increase) in the resonance wavelengths is induced by using a localized heater. Further several athermal photonic devices have been presented to reduce the localized tuning/trimming power in MRRs. These design time solutions include using cladding to reduce thermal sensitivity [61] and using heaters as well as temperature sensors for thermal control. While these device-level techniques are promising, they either possess a high power overhead or require costly changes in the manufacturing process (e.g., much larger device areas) that would decrease network bandwidth density and area efficiency. At the system-level, a thread migration framework was presented in[62] to avoid on-chip thermal threshold violations and also reduce trimming/tuning power for MRs. In[63], a ring aware thread scheduling policy was proposed to reduce on-chip thermal gradients in a PNoC. A proportional-integral-derivative (PID) heater mechanism was proposed in[7] that minimizes the effect of thermal variation on PNoCs performance and power. However, all these system-level techniques do not consider the impact of run-time workload variations and also result in considerable power performance overheads.

Our goal in this paper is to minimize thermal variations with reduced localized thermal tuning and trimming in PNoCs, thereby reducing key overheads and ultimately easing the adoption of PNoCs for future CMP systems. We propose a novel low-power thermal management framework that integrates an adaptive heater mechanism at the device-level and a dynamic thread migration scheme at the system-level. This chapter makes the following contributions:

- A novel temperature island framework with adaptive heater based MRR to handle thermal gradients across PNoC;

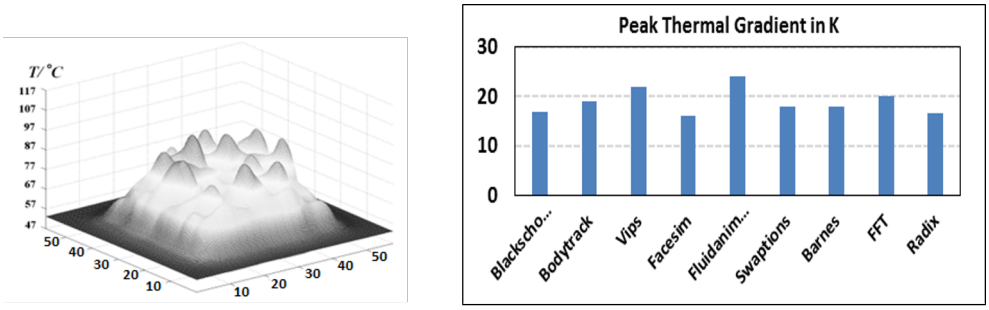


Figure 3.2: Thermal Distribution: (a) a 64-core CMP, (b) Peak thermal gradient across a 64-core chip running 48-threaded PARSEC [5] and SPLASH-2 [6] benchmarks. *Reprinted with permission from* [4]

- An island of heaters based dynamic thread migration (IHDTM) scheme in conjunction with a support vector regression based temperature prediction mechanism. Such a scheme nullifies on-chip thermal threshold violations and also reduces trimming/tuning power for MRRs;
- The evaluation of the proposed framework on a 64-core CMP with a system-level simulator shows: (a) 70% improvement in trimming power dissipation over the most recent prior work, (b) 64.1% improvement in total power dissipation compared to a state-of-the-art thermal management technique, (c) 13.72K improvement in peak temperature, and (d) these improvements are achieved while maintaining full network-bandwidth.

3.3 IHDTM: Islands of Heater-based Dynamic Thermal Management

The proposed IHDTM framework enables variation-aware thermal management by integrating device-level and system-level enhancements. A high-level overview of the framework is shown in Fig.3.3. At the device-level, the entire PNoC layer is divided into k regions or islands, namely: T_{IS1}-island, T_{IS2}-island, T_{IS3}-island, and so on. All MRRs in the T_{ISi}-island (i $\leq$ k) are designed to operate at T_{ISi}; similarly, MRRs in the other islands are designed to operate at their respective temperatures. We use our device-level technique to overcome small deviations ($\pm$10K) in T_{ISi} whereas the system-level technique is used to adapt to larger variations (> $\pm$10K). The device-level technique aims to adapt to the changing on-chip thermal profile, maintaining maximum bandwidth

and correct MRR operation while minimizing trimming and tuning power in the PNoC. At the system-level, the dynamic thread migration scheme maintains acceptable core-temperatures for each island. The following sections explain the proposed (i) device-level island framework and (ii) system-level thread migration scheme in detail.

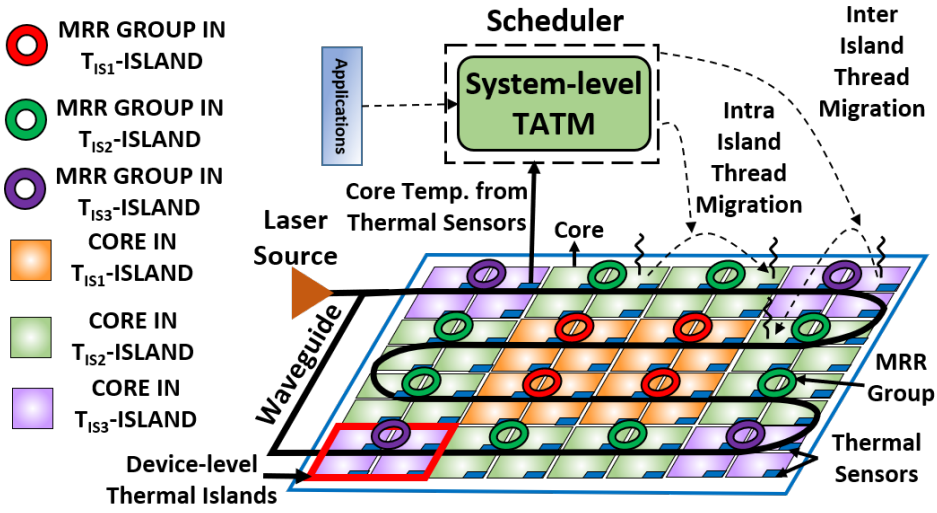
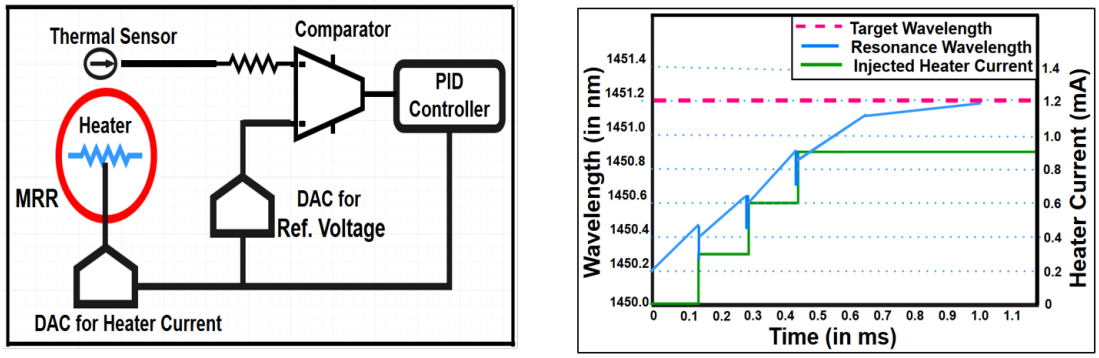


Figure 3.3: IHDTM framework with device-level thermal islands and system-level temperature-aware thread migration mechanism (TATM). *Reprinted with permission from* [4]

3.3.1 Thermal Islands

The thermal distribution across a 64-core PNoC chip running PARSEC and SPLASH-2 benchmarks (using 3D-ICE simulation) shows three major zones of temperature: 363K, 343K, and 323K. Also, the average thermal gradient in the PNoC chip is found out to be approximately 15-20K. To reduce this gradient, the proposed device-level framework adopts three islands (as shown in Fig. 3.3) each of which are maintained at a unique temperature by assigning T_{IS1}, T_{IS2}, and T_{IS3} to 363K, 343K, and 323K respectively. As mentioned in the previous section, MRRs in the 363K-island (T_{IS1}-island) are designed to operate at 363K with a variation range of $\pm$10K. MRRs employ thermal tuning and electrical trimming when they are operated below and above their designed temperatures respectively. Similarly, MRRs in other islands are designed to operate at the respective

temperatures. For PNoCs of other sizes (e.g. 16-core, 25-core, 36-core, 128-core, 256-core), there can be slight variations in the number of islands and their respective temperature zones. Accordingly, the numbers of islands and their temperatures can be fixed at design time.



(a) (b)

Figure 3.4: (a) MRR with adaptive heater (b) Thermal tuning of MRR. *Reprinted with permission from* [7]

To manage localized temperature variation below designed temperature, each MRR is integrated with a PID controller based heater as shown in Fig.3.4(a). The PID controller is tuned with proportional band Kp=50, integral cycle-time Ki=1 millisecond (ms), and derivative coefficient Kd=0. An open source PID tuning software [64] is used to determine optimal values of K_p, K_i, and K_d.

Algorithm.2 depicts the control algorithm for the heater in each MRR to stabilize thermal variations. In the algorithm, T represents the temperature across an MRR as detected by the corresponding thermal sensor, T_{island} is the fixed temperature of the island in which the MRR resides (T_{island} = T_{ISi}), P_{Heat} is the heater power, iHeat represents heater current, and H_{eff} stands for the transfer function of the heater. With any local temperature change dT, there is an equivalent shift in resonance for the MRR. To undo this resonance shift in an MRR, an equivalent amount of heat must be radiated by the heater integrated with that MRR. As per the algorithm, the controller collects temperature data T from the local thermal sensor as input. In step 1, the absolute value

Algorithm 2 Thermal management of MRR

1: **Input:Temperature (T) around the MRR as detected by Thermal sensor** Controller converts T to appropriate heater current as follows:
2: dT = $|T_{island}$ - T|
3: P_{Heat} = $\frac{dT}{\rho} \times H_{eff}$
4: **if** (T $\leq T_{island}$) **then**
5: i_{Heat} =i_{Max}- $\sqrt{\frac{P_{Heat}}{R_{Heat}}}$
6: **else**
7: i_{Heat} =i_{Max} + $\sqrt{\frac{P_{Heat}}{R_{Heat}}}$
8: Delay of 1 milisecond
9: Go to step 1
10: **Output: current (i_{Heat}) to be fed to heater**

of the difference between T and T_{island} is calculated followed by determining the required heater power PHeat in step 2. T is compared with T_{island} in step 3 and accordingly the required heater current iHeat is computed either in steps 3-4. The evaluated value of iHeat is fed to the heater coil. This amount of current is needed by the heater to maintain the fixed temperature T_{island} around the MRR. Our analysis shows that a maximum of 1 ms of time is needed for the heater element to bring the surrounding temperature to the desired value of T_{island}. We account for this time delay in our simulations. Fig.3.4(b) shows the tuning process of an MRR with injected heater current as explained in the algorithm. The control algorithm is invoked after every 1ms for each MRR.

This heater-based technique helps to stabilize thermal fluctuations in each temperature island with reduced tuning power. However, if the power footprint of a workload on a core associated with a 363K-island is very low, its core temperature may fall below the lower thermal limit (i.e. smaller than 353K). This thermal gradient can significantly increase tuning power consumption of an associated MRR. Similarly, if the power footprint of a workload on a core associated with a 323K-island keeps increasing beyond a threshold, then its core temperature might reach beyond the control of the MRR-trimmer (i.e. greater than 333K). This will in turn permanently shift the resonance of the MRR, inducing errors during communication. To address these issues, we propose a system-level temperature-aware thread migration (TATM) technique that performs thread migra-

tion to idle cores to maintain temperatures of corresponding MRRs close to the design temperatures of their respective islands. By intelligently migrating threads, this technique reduces device-level tuning/trimming power in MRRs. TATM also aims to proactively reduce thermal hotspots, which in turn will reduce instances of irrecoverable drift in MRRs.

3.3.2 Temperature-Aware Thread Migration Scheme (*TATM*)

- Objective: The primary goal with TATM is to maintain the temperature of all the cores in an island on a die below a specified thermal threshold (T_t) and above a thermal limit (T_l), i.e., for a core i in the T_{ISj}-island, $T_{lj} \leq T_i \leq T_{tj}$ where T_i is the temperature of core i, T_{tj} is threshold temperature of T_{ISj}-island, and T_{lj} is thermal limit of T_{ISj}-island. TATM maintains the core temperatures such that the temperature of all the MRRs within an island is close to their design temperature, to reduce tuning power consumption in adaptive heaters as explained in the previous section.

  We utilize support vector based regression (SVR) to predict the future temperature of a core. This predicted temperature of a core is compared with the corresponding islands thermal threshold (upper limit) and thermal limit (lower limit) to determine the potential for a thermal emergency. If such a potential exists, then TATM initiates thread migration. Inter-island thread migration (Inter-island cores (IEIC)) is preferred over intra-island thread migration (Intra-island cores (IAIC)). This step has a two-fold benefit. Firstly, by moving the thread away from a core that could suffer a thermal emergency, we avoid instances of irrecoverable drift in the MRR groups of that core. Secondly, by moving the thread to a core in different island, we ensure that the temperature of the island and its corresponding ring blocks remains between the islands thermal threshold (T_{t1}, T_{t2}, and T_{t3}) and thermal limit (T_{l1}, T_{l2}, and T_{l3}) to conserve trimming/tuning power. If a thermal emergency occurs due to exceeding the thermal threshold, then it is preferred that the thread is migrated to a core in an island whose MRR design temperature is higher. If a thermal emergency occurs due to temperature falling below the thermal limit then it is preferred that the thread is migrated to a core in an island whose MRR design temperature is lower. The parameters used to describe TATM in this

section are shown in Table 3.1.

Table 3.1: List of TATM parameters and their definitions. *Reprinted with permission from* [4]

Symbol	Definition
IPC_i	Instruction per cycle of ith core
T_i	Current temperature of ith core
TN_i	Average temperature of immediate neighboring cores of ith core; if this core is on chip periphery and missing neighbors, then we consider virtual neighbor cores at ambient temperature in lieu of the missing cores
PT_i	Predicted temperature of ith core
T_t	Thermal threshold
$IEIC_j$	Inter-island cores for island-j
$IAIC_j$	Intra-island cores for island-j
C	Regularization parameter
W	Weight vector for regression
x_i and y_i	input and output in training and test data
β	slack variable
ϵ	Error function
b	bias for cost function

- Temperature Prediction Model: We designed a support vector regression (SVR) based temperature predictor that accepts input parameters reflecting the workload for a core i, in terms of instructions per cycle (IPC_i), temperature (T_i), and surrounding core temperatures (TN_i), and predicts the future temperature for core i.

  Architecture: A typical SVR[65] relies on defining a prediction model that ignores errors that are situated within the ϵ range of the true value. This type of a prediction model is called an ϵ-insensitive prediction model. The variables (ξ and ϵ) measure the cost of the errors on the training points. These are zero for all points that are inside the ϵ-insensitive

band. SVR is primarily designed to perform linear regression using a cost function (CF) as depicted in equation 3.1.

$$CF = min\frac{1}{2}W^T.W + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{3.1}$$

Subject to:

$$y_i - W^T\phi(x_i) - b \leq \epsilon + \xi_i(\xi_i \geq 0, i = 1, 2, , n) \tag{3.2}$$

$$W^T\phi(x_i) + b - y_i \geq \epsilon + \xi_i^*(\xi_i^* \geq 0, i = 1, 2, , n) \tag{3.3}$$

$$\kappa(x_i, x_j) = \phi(x_i)^T\phi(x_j) \tag{3.4}$$

SVR performs linear regression in this high-dimension space using ϵ-insensitive loss and, at the same time, tries to reduce model complexity by minimizing $W^T.W$. This can be described by introducing (non-negative) slack variables ξ_i and ξ_i^* (i = 1 to n), to measure the deviation of training samples outside the ϵ-insensitive band. Thus SVR is formulated as minimization of the cost function (CF) in equation 3.1 with constraints shown in equations 3.2 and 3.3.

To handle nonlinearity in data, SVR first maps the input x_i onto an m-dimensional space using some fixed (nonlinear) mapping denoted as ϕ, and then a linear model is constructed in this high-dimensional space as shown in equations 3.2 and 3.3. This allows it to overcome drawbacks of linear and logistic regression towards handling nonlinearity in data. This class of SVRs is called kernel based SVRs which use kernel κ as shown in equation 3.4 for implicit mapping of nonlinear training data into a higher dimensional space.

As on-chip temperature variation data is nonlinear in the original space, our SVR model employs a kernel based regression which uses a Radial Basis Function (RBF)[66] as shown

in equation 3.5:

$$\kappa(x_i, x_j) = exp(-\gamma|x_i - x_j|^2) \tag{3.5}$$

The RBF kernel improves the accuracy of SVR when data has nonlinearity in the original space. We performed a sensitivity analysis (SA) to determine regularization parameter (C) and gamma (γ) values of the kernel based SVR (see Section 3.4.1 for chosen values). This SA overcomes the possibility of over fitting of training data and improves accuracy further. The definition of each of the variables used in equations 3.1 to 3.5 are mentioned in Table 3.1. Training and Accuracy: We trained our SVR model using a set of multi-threaded applications from the PARSEC[5] and SPLASH-2[6] benchmark suites, specifically: blackscholes (BS), bodytrack (BT), vips (VI), facesim (FS), fluidanimate (FA), swaptions (SW), barnes (BA), fft (FFT), radix (RX), radiosity (RD), and raytrace (RT) with different thread counts: 2, 4 and 8. We considered different combinations of thread mappings on a 9-core (3×3) floorplan, to train our predictor to determine the temperature of the center (target) core. The threads mapped to a 9-core floorplan represents a generic mapping and can be applied to 64-core, 128-core, and 256-core floorplans. As the future temperature of a target core is dependent on the average temperature of its immediate neighboring cores, we trained our SVR model with temperature inputs from the target core running a single thread, as well as its surrounding cores running a variable number of threads. Simulations with various mappings of these threads allowed us to obtain data to train our SVR model. This data included temperature for the target core and its neighboring core temperatures, as well as instructions per cycle (IPC) for the target core. IPC is very useful to determine if there is a phase change in an application and plays a crucial role in maintaining future temperature prediction accuracy especially when temperatures of a target core and its neighbors are similar at a given time. Our training algorithm involved an iterative process that adjusts the weights and bias values in the SVR (equations 3.1 to 3.3) to fit the training set.

We verified the accuracy of our SVR model for multi-threaded benchmark workloads (we

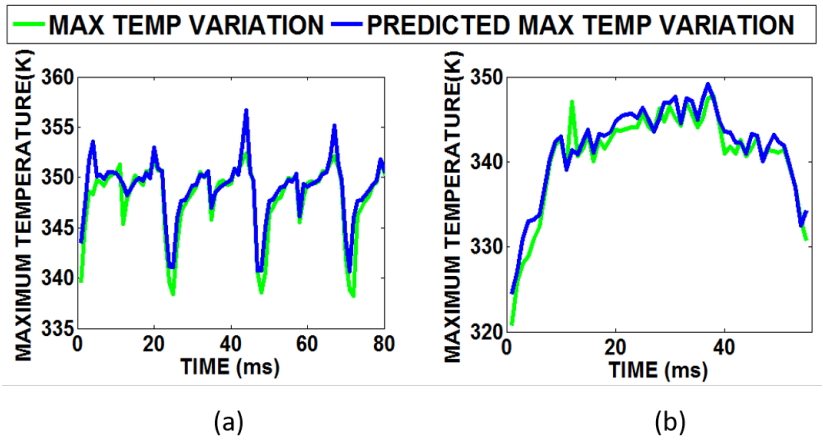


Figure 3.5: Actual and predicted maximum temperature variation with execution time for (a) fluidanimate (FA) and (b) radiosity (RD) benchmarks run on a 64-core platform executing 32-threads. *Reprinted with permission from* [4]

considered 6000 floorplans, with 70% of input data for training and 30% for testing) and found that it has an accuracy of over 95%. Fig.3.5(a) & (b) show actual and predicted on-chip temperature variations for a 64-core platform executing 32 threads of the FA and RD benchmarks. From these figures it can be seen that our temperature predictor tracks temperature quite accurately. When predicted temperature is beneath thermal limit or exceeds the thermal threshold our thread migration mechanism (which is discussed next) migrates threads between cores to reduce tuning/trimming power and keep overall maximum temperature below the threshold.

- Thermal Management Algorithm: Fig.3.6 illustrates the entire TATM technique. For each core, we periodically monitor the IPC value from performance counters and temperature from thermal sensors. If a thermal emergency is predicted for a core by the SVR predictor, then TATM initiates a thread migration procedure, otherwise no action is taken. In this work we have considered the thermal threshold of an island to be equal to maximum allowable temperature in that island i.e. T_{tj} = T_{ISj} + 10K to avoid instances of irrevocable drift in MRs and thermal limit of an island is minimum allowable temperature in that island i.e. T_{lj} = T_{ISj} - $10K$ to reduce tuning power.

Algorithm 3 IHDTM thread migration algorithm

Inputs: Current core temperature (T_i), average neighboring core temperature (TN_i), current core IPC (IPC_i)
for each core i **do** *// Loop that predicts future temperature*
    PT_i = SVR_predict_future_temperature (T_i, TN_i, IPC_i)
for each core i **do** *// Loop that checks for free IAICs*
    j = Find island of Core(i)
    if IPC_i == 0 **then**
        List_$IAIC_i$ = Push i *// add core to $IAIC_i$ list*
for core i **do** // Loop that performs thread migration
    j = Find island of Core(i)
    for each island k **do** *// Loop that generates IEIC list for island-j*
        if k $\neq$ j **then**
            $IEIC_j$ = push $IAIC_k$
    if $PT_i \geq T_{tj}$ **then**
        if List_$IAIC_i \neq \{\}$ **then** *// Do intra-island thread migration*
            Migrated_core = min_temp_core(List_$IAIC_j$)
            Thread_migration(core_i $\rightarrow$ Migrated_core)
            List_$IAIC_j$ = Pop Migrated_core
        else if List_$IEIC_j \neq \{\}$ **then**
            Migrated_core = min_temp_core(List_$IEIC_j$)
            Thread_migration(core_i $\rightarrow$ Migrated_core)
            n = island of Migrated_core
            List_$IAIC_n$ = Pop Migrated_core
Output: Thread migration to IAIC or IEIC cores

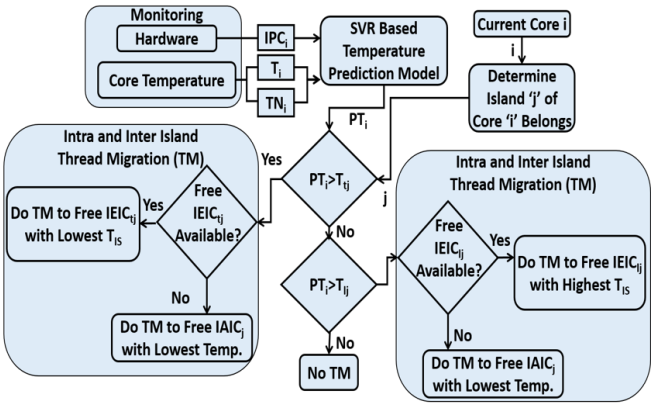


Figure 3.6: Overview of TATM technique with support vector regression (SVR) based temperature prediction model. *Reprinted with permission from* [4]

Algorithm. 3 shows the pseudocode for the TATM thread migration procedure. Firstly, future temperature (PT_i) of the ith core is predicted using the SVR based predictor with inputs:core temperature (T_i), core IPC (IPC_i), and temperature of neighboring cores (TN_i) in steps 1-2. The list of available free cores ($IAIC_j$) in T_{ISj}-island (i.e., those that are not currently executing any thread) is obtained in steps 3-5. In steps 6-9, a loop iterates over islands to generate a list of free cores $IEIC_{tj}$ and $IEIC_{lj}$ in other islands whose T_{IS} is higher and lower than current island respectively. In step 10, a loop iterates over all cores to perform thread migration. Step 12 and 22 checks for possible thread migration conditions (i.e., thermal emergency cases where current core predicted temperature (PT_i) in T_{ISj}-island is greater than thermal threshold (T_{tj}) or smaller than thermal limit (T_{lj})). If a thread migration is required as $PT_i > T_{tj}$, then in steps 13-21, we check for free $IEIC_{tj}$, and if they are available then we migrate the thread from the current core to the $IEIC_{tj}$ core with the lowest T_{IS} (inter-island migration), else we migrate the thread to a free IAICj with the lowest temperature (intra-island migration). On the other hand, if a thread migration is required as $PT_i < T_{lj}$, then in steps 23-31, we check for free $IEIC_{lj}$, and if they are available then we migrate the thread from the current core to the $IEIC_{lj}$ core with the highest T_{IS} (inter-island migration), else we migrate the thread to a free $IAIC_j$ with the lowest temperature (intra-island

migration). This TATM thread migration technique is invoked at every 1ms (epoch) and the sample frequency of SVR is considered as 0.1 ms (10 times lower compared to the epoch for thread migration). This sampling frequency is sufficient to monitor on-chip temperature variations [20].

3.4 Experiments, Results, and Analysis

3.4.1 Experimental Setup

The IPKISS[48] tool was used for the design and simulation of heaters, MRRs, and other silicon photonic components. This tool allows photonic component layout design, virtual fabrication of components in different technologies, physical simulation of components, and optical circuit design and simulation. The circuit-level results obtained from IPKISS were used for system-level simulation.

We target a 64-core CMP system for evaluation of our IHDTM framework. Each core has a Nehalem x86[67] microarchitecture with 32 KB L1 instruction and data caches and a 256 KB L2 cache, at 32nm and running at 5GHz. We evaluate our framework on two well-known PNoC architectures: Corona[68] and Flexishare[69]. Corona uses a 64×64 multiple write single read (MWSR) crossbar with token slot arbitration. Flexishare uses 32 multiple write multiple read (MWMR) waveguide groups with a 2-pass token stream arbitration. Each MWSR waveguide in Corona and each MWMR waveguide in Flexishare is capable of transferring 512 bits of data from a source node to a destination node.

Table 3.2: Properties of materials used by 3D-ICE tool. *Reprinted with permission from* [4]

Material	Thermal Conductivity	Volumetric Heat Capacity
Silicon	1.30e-4 W/μm K	1.628e-12 J/μm^3 K
Silicon Dioxide	1.46e-6 W/μm K	1.628e-12 J/μm^3 K
BEOL	2.25e-6 W/μm K	2.175e-12 J/μm^3 K
Copper	5.85e-4 W/μm K	3.45e-12 J/μm^3 K

We modeled and simulated these architectures with the IHDTM framework for multi-threaded applications from the PARSEC [5] and SPLASH-2[6] benchmark suites (Section 3.3.2). Simulations were performed with an execution period of one billion cycles. Power and instruction traces for the benchmark applications were generated using the Sniper 6.0[67] simulator and McPAT[70]. We used the 3D-ICE tool[58] for thermal analysis. We considered a three layered 3D-stacked CMP system as advocated in existing PNoC architectures [68], [69] with a planar die area footprint of $400mm^2$, where the top layer is the core-cache layer, the middle layer is the analog electronic layer [68] which contains control circuits for modulator and photodetector and also the trans-impedance amplifiers of detectors, and the bottom layer is the photonic layer with MRRs, waveguides, ring heaters, and ring trimmers for carrier injection. Some of the key materials used in the construction of the 3D-stack in the 3D-ICE tool and their properties are shown in Table 3.2. We used a heat sink adjacent to the core-cache layer for heat dissipation to the ambient environment.

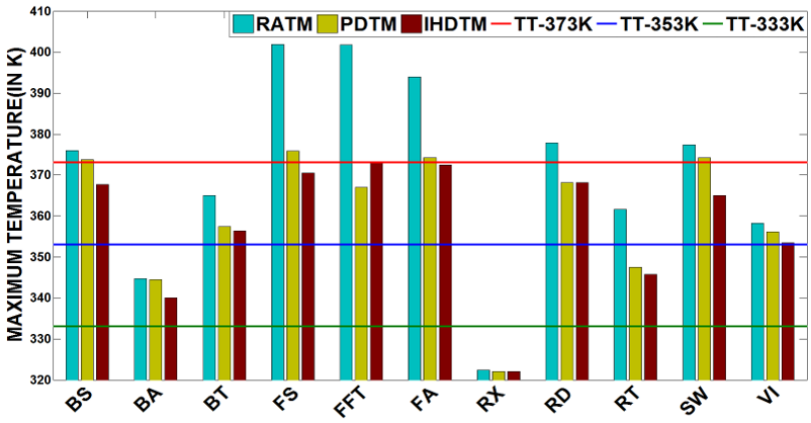
The MRR thermal sensitivity was assumed to be 0.11nm/K[60]. For PNoCs, we considered 64 dense-wavelength-division-multiplexing (DWDM) waveguides sharing the working band 1530-1625 nm. The MRR trimming power is set to 130ţW/nm[59] for current injection (blue shift) and tuning power is set to 240μW/nm[60] for heating (red shift). To compute laser power, we considered detector responsivity as 0.8 A/W[71], MRR through loss as 0.02 dB, waveguide propagation loss as 1 dB/cm, waveguide bending loss as 0.005 dB/90^0, and waveguide coupler/splitter loss as 0.5 dB[71]. We calculated photonic loss in components using these values, which sets the photonic laser power budget and correspondingly the electrical laser power. For energy consumption of photonic devices, we adapt parameters from[65], with 0.42pJ/bit for every modulation and detection event, and 0.18pJ/bit for modulator/detector driver circuits. The ambient temperature was set to 303K for our analysis and the for T_{IS1}-island, T_{IS2}-island, and T_{IS3}-island thermal thresholds were set to 373K, 353K, and 333K respectively and the thermal limits were set to 353K, 333K, and 313K respectively. Based on our sensitivity analysis we get the best accuracy for our SVR-based temperature predictor when parameters C and γ are set to 1000 and 0.1 respectively. We also considered thread migration overhead in our simulations that ranged from 500-1000 cy-

cles to account for startup latency (extra cache misses, branch miss predictions) in the migrated core. Further, in the simulation we considered a 250-500 cycles overhead towards migration of threads for writing dirty cache lines from the write back caches, flushing the pipeline in the source core, and also PNoC latency to transfer data from architectural registers from the source core to the migrated core.

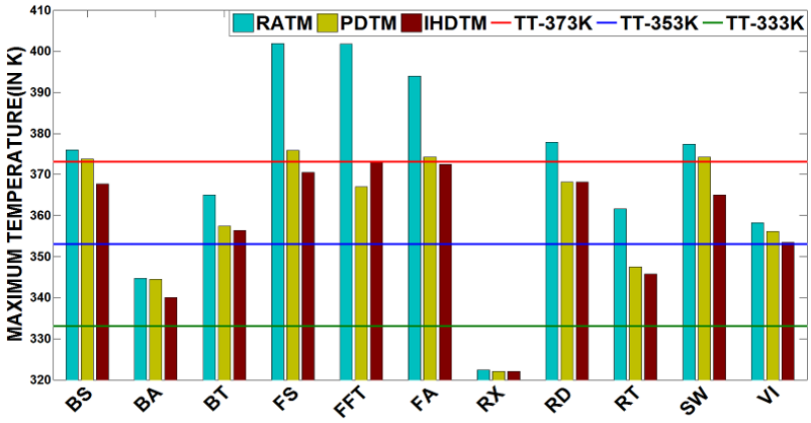
3.4.2 Experimental Results

We compared the performance of our IHDTM framework with two prior works on multicore thermal management: a ring aware policy (RATM)[63] and a predictive dynamic thermal management (PDTM) framework[72]. To compare these frameworks, we consider Corona and Flexishare PNoC architectures. RATM distributes threads uniformly across cores that are closer to PNoC nodes first and then distributes the remaining threads in a regular pattern from outer cores to inner cores. PDTM uses a recursive least square based temperature predictor to determine if the predicted temperature of a core exceeds a thermal threshold, and if so then thread migration is performed from that core to the coolest free core.

Fig.3.7 shows the maximum temperature obtained with the three frameworks across eleven applications from the PARSEC and SPLASH-2 benchmarks suites with 48 and 32 thread counts executed on a 64-core system with the Corona PNoC architecture. From Fig. 3.7(a) it can be observed that for the IHDTM framework the FFT application with 48 threads exceeds the threshold (363K) by 0.4K as there are insufficient number of free cores in the 363K-island on the chip whose temperature is below the thermal threshold to migrate threads. However, in Fig. 3.7(b) our IHDTM framework avoids violating thermal thresholds for all the benchmark applications with 32 threads. On average, IHDTM has 13.27K and 13.72K lower maximum temperature compared to the RATM policy for 48 and 32 threads, respectively. Along with local thermal stabilization by PID controlled heaters, IHDTM migrates threads from hotter cores to cooler cores to control maximum temperature, whereas RATM does a simple thread allocation that is unable to appropriately control maximum temperature. For most of the cases, maximum temperatures with PDTM and IHDTM are below the thermal threshold. On average, IHDTM has 2.37K and 1.56K lower maximum



(a)

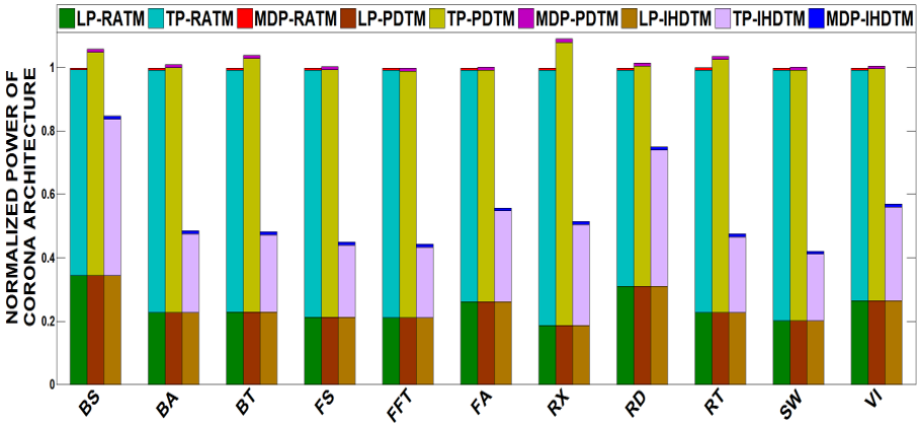


(b)

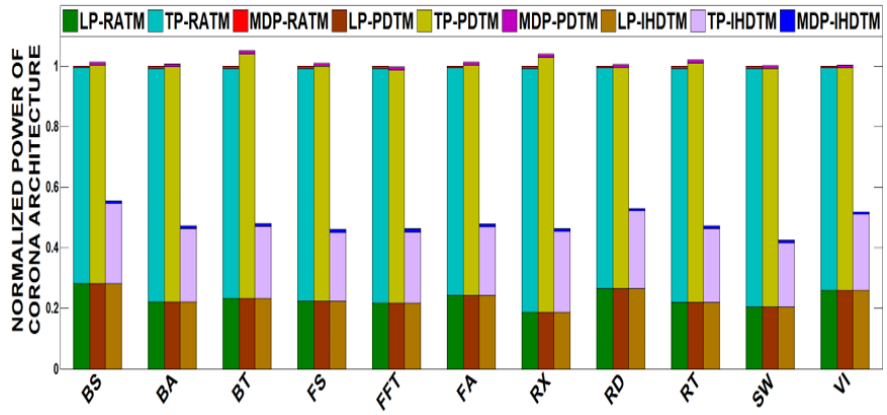
Figure 3.7: Maximum temperature comparison of IHDTM with RATM and PDTM for (a) 48 (b) 32 threaded PARSEC and SPLASH-2 benchmarks executed on 64-core multicore system with Corona PNoC. *Reprinted with permission from* [4]

temperature compared to the PDTM policy for 48 and 32 threads, respectively. IHDTM prefers to migrate threads within islands (inter-island) of cores based on the power consumption of running thread, which facilitates reduction in its peak temperature compared to PDTM.

Fig.3.8 and Fig.3.9 show the power consumption comparison for the three thermal-management techniques across multiple 48-threaded and 32-threaded applications for the Corona and Flexishare PNoC architectures, respectively. One of the main reasons why IHDTM has lower power consumption than RATM and PDTM is that it more aggressively reduces thermal tuning and trimming power in both Corona and Flexishare PNoCs. It is evident from Fig.3.8(a)-(b) that IHDTM run-



(a)



(b)

Figure 3.8: Normalized power (Laser Power (LP), Trimming and tuning power (TP) and modulating and detecting Power (MDP)) comparison of IHDTM with RATM and PDTM for (a) 48 and (b) 32 threaded applications of PARSEC and SPLASH-2 suites executed on Corona PNoC architectures for a 64-core multicore system. Results shown are normalized w.r.t RATM. *Reprinted with permission from* [4]

ning 48 threads has 61.6% and 62.5%; and IHDTM running 32 threads has 67.3% and 68.5% lower thermal tuning and trimming power on average compared to RATM and PDTM for Corona PNoC architecture respectively. Similarly, from Fig.3.9(a) and (b), it can be seen that IHDTM running 48 threads has 62.8% and 63.9%; and IHDTM running 32 threads has 68.5% and 70% lower tuning and trimming power on average compared to RATM and PDTM for Flexishare PNoC.respectively. The IHDTM framework intelligently conserves tuning/trimming power compared to RATM and PDTM by performing intelligent intra-island and inter-island thread migration.

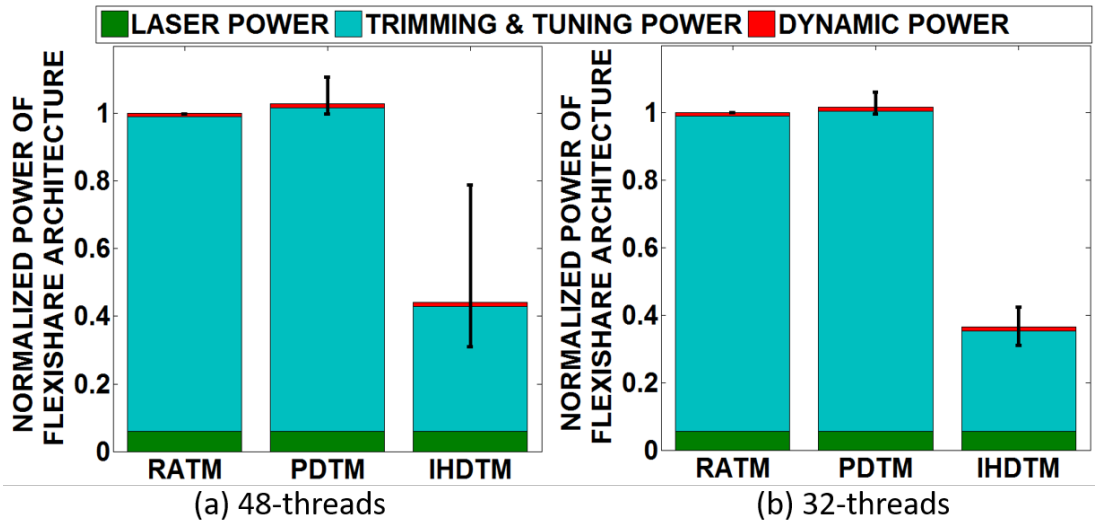
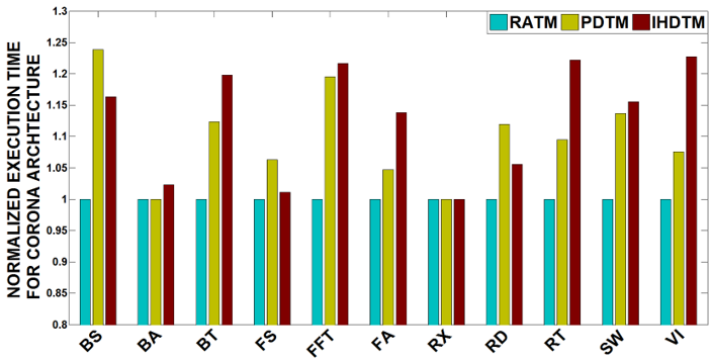


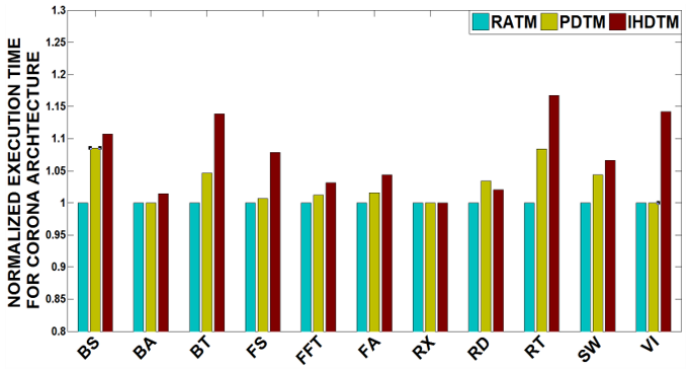
Figure 3.9: Normalized average power (Laser Power (LP), Trimming and tuning power (TP) and modulating and detecting Power (MDP)) comparison of IHDTM with RATM and PDTM for (a) 48 and (b) 32 threaded applications of PARSEC and SPLASH-2 suites executed on Flexishare PNoC architectures for a 64-core multicore system. Power results are normalized wrt RATM results. Bars represent mean values of power dissipation; confidence intervals show variation in power dissipation across PARSEC and SPLASH-2 benchmarks. *Reprinted with permission from* [4]

IHDTM saves considerable thermal tuning and trimming power to ultimately reduce total power. From the power analysis in Fig.3.8 and Fig.3.9, it can be observed that IHDTM with Corona running 48 threads has 45.5% and 46.8%; and IHDTM with Corona running 32 threads has 51.6% and 52.3% lower total power consumption compared to Corona with RATM and PDTM respectively. Further, Flexishare with IHDTM running 48 threads has 55.9% and 57.2%; and 32 threads has 63.5% and 64.1% lower power consumption compared to Flexishare with RATM and PDTM respectively.

Fig.3.10 shows the average execution time comparison between the three frameworks across the 11 48-threaded and 32-threaded applications from the PARSEC and SPLASH-2 suites, for the Corona PNoC architectures respectively. From Fig.3.10(a) and (b) it can be seen that Corona with IHDTM running 48 and 32 threads has 12.8% and 7.4% higher execution time respectively compared to Corona with RATM. Corona with IHDTM needs extra execution time to migrate threads between cores whereas the RATM policy simply schedules threads without any migration,



(a)



(b)

Figure 3.10: Normalized execution time comparison of IHDTM with RATM and PDTM for Corona PNoC running (a) 48; and (b) 32 threaded applications from PARSEC and SPLASH-2 suites executed on 64-core system. Results shown are normalized wrt RATM. *Reprinted with permission from* [4]

and thus does not possess such overheads. The execution time overhead of Corona with IHDTM running 32 threads is lower compared to 48-threaded version, as it lowers traffic congestion in the Corona PNoC which in turn reduces overall latency. Further, Corona with IHDTM running 48 and 32 threads has 2.6% and 4.3% higher execution time respectively compared to PDTM. IHDTM has more number of thread migrations compared to the number of thread migrations in PDTM, as IHDTM performs intra-island and inter-island thread migrations when the thermal emergencies are predicted by the SVR predictor. Similarly, from Fig.3.11(a) and (b), the Flexishare with IHDTM running 48 and 32 threads has 9% and 5.9% higher execution time compared to RATM and 3.4% and 4.4% higher execution time compared to Flexishare with PDTM. From the execution time results it can be seen that Flexishare has lower execution time overhead compared to Corona as it

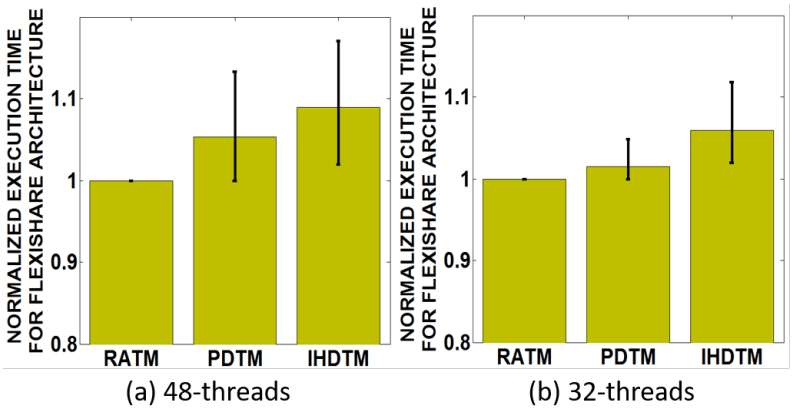


Figure 3.11: Normalized average execution time comparison of IHDTM with RATM and PDTM for Flexishare PNoC running (a) 48; and (b) 32 threaded applications from PARSEC and SPLASH-2 suites executed on 64-core system. Power results are normalized wrt RATM results. Bars represent mean values of power dissipation; confidence intervals show variation in power dissipation across PARSEC and SPLASH-2 benchmarks. *Reprinted with permission from* [4]

uses a faster MWMR crossbar instead of slower MWSR crossbar in Corona.

Lastly, from the power consumption and execution time results, we can obtain energy consumption results for the three frameworks. On an average, for Corona, energy consumption of IHDTM running 48 threads is 38.5% and 45.4% lower compared to RATM and PDTM, respectively. Further energy consumption of Corona with IHDTM running 32 threads is 48.1% and 50.3% lower compared to RATM and PDTM, respectively. On the Flexishare architecture, IHDTM running 48 threads has 52.2% and 56% lower energy consumption compared to RATM and PDTM respectively; and IHDTM running 32 threads has 61.4% and 62.6% lower energy consumption compared to RATM and PDTM, respectively. From the energy consumption results IHDTM has better energy savings for the optimized Flexishare compared to the Corona.

3.5 Chapter Summary

We have presented the IHDTM framework that exploits device-level on-chip thermal islands and system-level dynamic thread migration scheme TATM for the reduction of maximum on-chip temperature and also conserves trimming and tuning power of MRRs in DWDM-based PNoC architectures. The proactive thermal management scheme used in IHDTM results in interesting

trade-offs between performance and power/energy across two different state-of-the-art crossbar-based PNoC architectures. Our experimental analysis on the well-known Corona and Flexishare PNoC architectures has shown that IHDTM can notably conserve total power by up to 64.1% and thermal tuning power by up to 70%.

4. APPLICATION SPECIFIC PNOC FOR BIG DATA COMPUTING*

4.1 Introduction

[1]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 [73] implemented either in the cloud or on commodity clusters in datacenters.

Recently, Hadoop[74] and Spark[75] based distributed frameworks are being increasingly used for MapReduce im-plementations on cloud services. However, wide-spread secu-rity exploits and higher off-loading time with cloud compu-ting have driven several organizations to build their own dat-acenters for big data processing[74][75]. Such datacenters are safer from intrusion with lower off-loading time, but according to the Hamiltons cost model[76] 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[77] (Fig.4.1). In the initial multicast phase (Fig.4.1(a)) 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 ex-

[1]Adapted with permission from [8]

change or shuffle data with other servant nodes. This phase is called the shuffle phase (Fig. refmapreduce(b)). Lastly, in the aggregation phase (Fig.4.1(c)), 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.

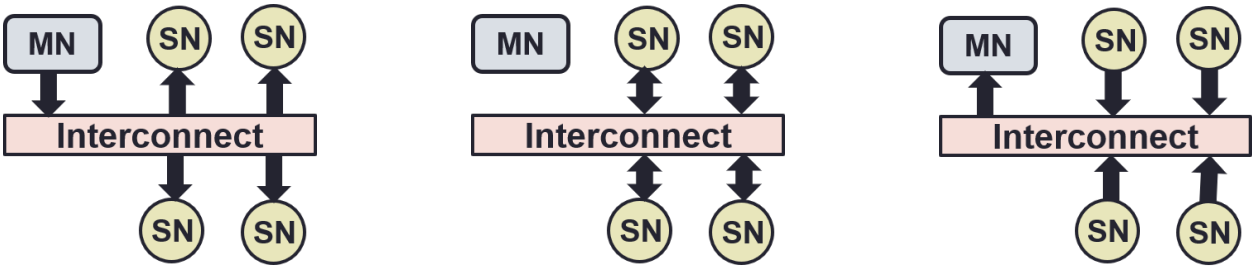


Figure 4.1: MapReduce (a) multicast phase, (b) shuffle phase, and (c) aggregation phase of communication while executing iterative machine learning algorithms for large-scale data analytics applications. MN: Master Node; SN: Servant Node

Multicasting is a performance bottleneck in executing large scale data analytics applications that have large fanout, 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[78] requires multicasting of approximately 200 MB of sampled cancer genomic features in each iteration, from 100 image samples, each of size 2MB. 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 385MB of data being distributed to servant nodes per iteration, over hundreds of iterations[11]. 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 advan-

tages 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[79]. 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[79]-[80] 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.

We present a novel application-specific PNoC architecture for manycore chips, called BiGNoC, to execute large-scale data analytics applications with high throughput and ultralow 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: (1) We devise a master-servant cluster based communication fabric (MSNoC) with dedicated channels for master-to-servant and servant-to-master communication; (2) We design a hierarchical manycore BiGNoC architecture with multiple MSNoCs to execute any combination of high performance large-scale data analytics applications; and (3) 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].

4.2 Related Work

Photonic interconnects utilize several photonic devices such as microring resonators (MRs) as modulators, detectors, and switches; photonic waveguides; splitters, and transimpedance 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.,[81]-[82], [83]-[84]) 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.,[81]-[82], [83] 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 detune 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[81] and MWMR[80] 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 [85] that uses photonic-distributed arbitration and concurrent channel reservation mechanisms. This multistage NoC uses an electrical router to interconnect multiple photonic subnetworks and achieves lower latencies. A high-throughput hybrid photonic mesh-diagonal links topology was proposed in [86] 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 [87], which can work as a coprocessor 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 [88] 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 [89]. The cores in this architecture are connected by a modified version of the torus topology, whereas the inter-chip topology is a 2D triangular mesh with 6-port routers. A neural network architecture called EMBRACE is proposed in [90] 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 regionbased 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 [91], [92]. A theoretical analysis for determining a preferred interconnect architecture for general purpose configurable emulation of spiking neural networks is presented in [91] 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 [92], which employs a reconfigurable communication fabric that efficiently handles multicast communication. In [93], 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 [94] for large-scale data analytics applications. The authors in [95] propose Melia, which is an FPGA-based MapReduce architecture. None of the above mentioned 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.

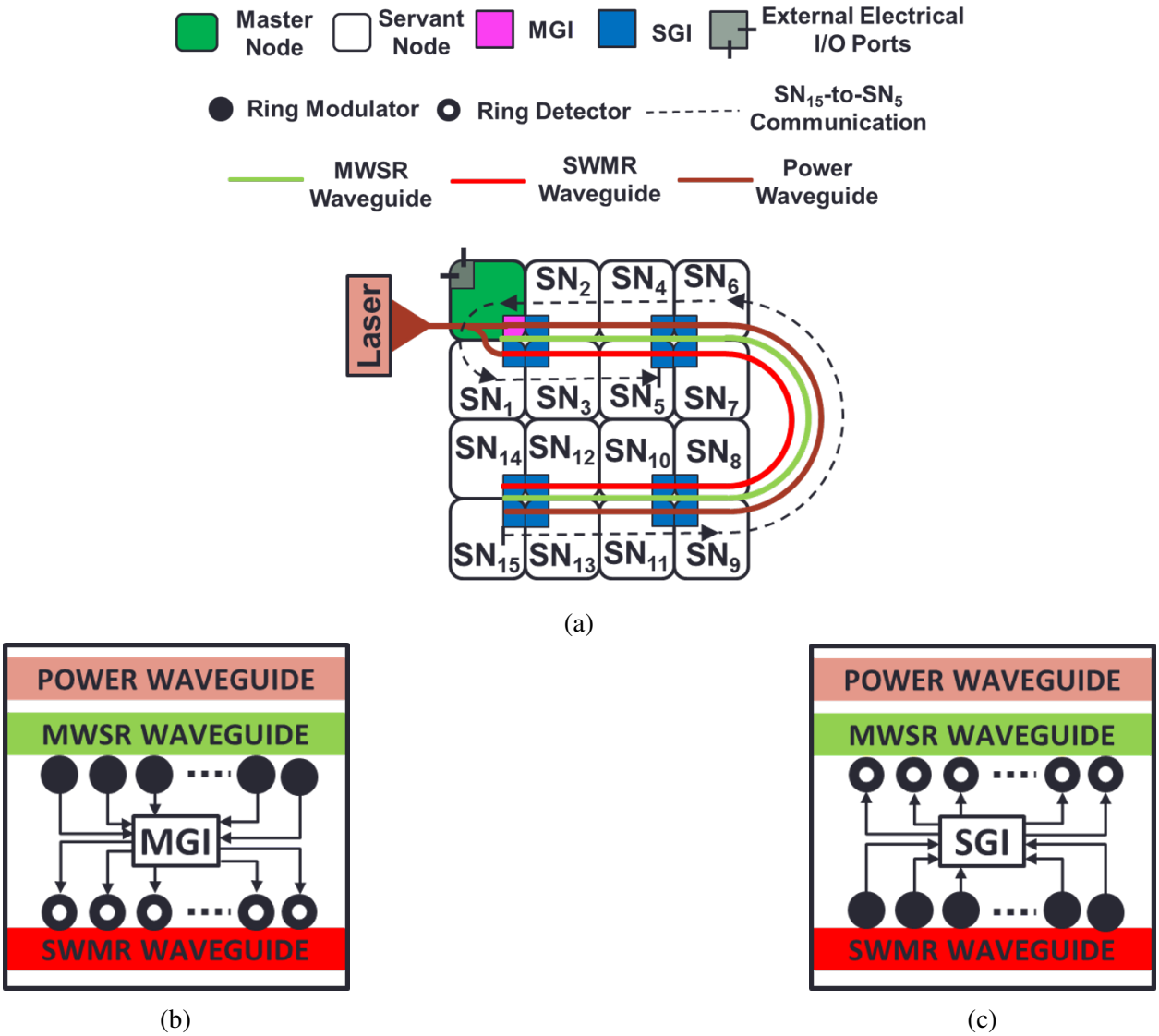


Figure 4.2: (a) MSNoC layout with SWMR, MWSR, and power waveguides (b) master gateway interface (MGI) (c) servant gateway interface (SGI). *Reprinted with permission from* [8]

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 ex-

change 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.

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 4.1. 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 4.1). As master cores access main memory more frequently compared to servant cores, therefore, the larger L1 size of a master core boosts the MSNoCs 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.4.2(a), 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., SN1 to SN15). The master GI (MGI) and servant GI (SGI) are shown in figures 4.2(b) and 4.2(c), respectively, and discussed further in Sections 3.1-3.3. 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 (mas-

ter 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 3.1-3.3). 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. 4.2(a). 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 4.6). 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 4.1. 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.

4.3 Master-Servant Cluster Architecture

4.3.1 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 photonic waveguides in our MSNoC cluster, to enable faster master-servant communication. As shown in Fig.4.2(a), in an MSNoC cluster we use a multicast enabled Single-Write-Multiple-Read (SWMR) waveguide group to enable

Table 4.1: Micro-architectural parameters for MSNoC cluster. *Reprinted with permission from* [8]

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

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., SN1-SN15) in the cluster (Fig.4.2(a)) to enable MN-to-SN communication. An MN has the ability to write on the SWMR waveguide group using its ring modulators (see Fig.4.2(b), 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. 4.2(c), 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 100mm2 chip area for a 64 core CMP at 22nm technology node, traversal of light through an SWMR waveguide group takes 2 cycles (i.e., 0.4 ns) in an MSNoC cluster at 5GHz 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.4.3. These time slots are further classified into two types: reservation cycle slots (RCS), and data cycle slots (DCS).

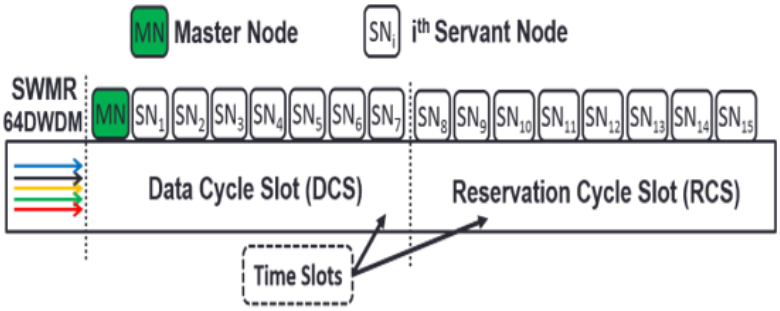


Figure 4.3: Distribution of reservation cycle and data cycle slots within SWMR waveguide to enable MN-to-SN communication. *Reprinted with permission from* [8]

In our reservation assisted MN-to-SN communication pro-cess, MNs send data to SNs in two cycles (Fig.4.3). 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.4.3). The remaining three SWMR waveguides in the SWMR waveguide group are used only in the data cycle to transfer data. Each SNi is assigned a receiver selection wavelength λ_i, that is available in the first SWMR waveguide of the SWMR waveguide group. 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 (λ_j λ_j+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 SNi, 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 SNi 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.4.4(a), suppose an MN needs to send data to SN8 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 SN8) is made available by removing all of the wavelengths except λ8 (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 SN8 to SN15 switch-on detectors with resonance wavelengths λ_8 to λ_{15}, respectively) in the first SWMR waveguide of the SWMR waveguide group. Therefore, at SN8 only the detector for wavelength λ_8 is switched on in the RCS. Once λ_8 is detected, SN8 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

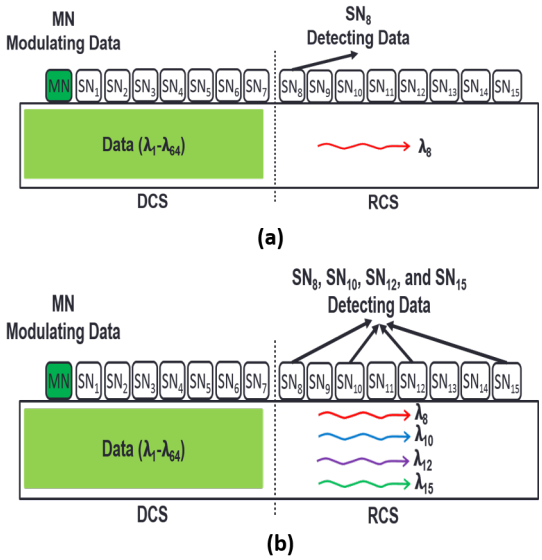


Figure 4.4: (a) Transmission of unicast data from an MN to SN8 in MSNoC, which shows receiver selection wavelength λ_8 in RCS of the SWMR waveguide; (b) Multicast of data from an MN to multiple SNs SN8, SN10, SN12, and SN15 in MSNoC, which shows respective receiver selection wavelengths λ_8, λ_{10}, λ_{12}, and λ_{15} in RCS of the SWMR waveguide. *Reprinted with permission from* [8]

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 [96]. Therefore, for MN to multiple SN communication in an MSNoC, we avoid such repeated unicast messages by providing multicasting support in the MSNoCs SWMR waveguides.

Unlike Corona [80] and Firefly [81] 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.4.4(b)) 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 detune the ring detectors from their resonating wavelengths [79], 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 [97], [98], as these works are orthogonal to our work.

Fig. 4.4(a) & (b) illustrate the difference between transmission of unicast and multicast messages in our MSNoC cluster. Suppose an MN needs to multicast data to SN8, SN10, SN12, and SN15 whose corresponding receiver selection wavelengths are λ_8, λ_{10}, λ_{12}, and λ_{15}, respectively. The MN modulates in the next RCS, such that only λ_8, λ_{10}, λ_{12}, and λ_{15} are made available by removing all the wavelengths except λ_8, λ_{10}, λ_{12}, and λ_{15} (using the MNs modulators; Fig. 4.4(b)) from the first SWMR waveguide of SWMR waveguide group. At the receiver end at SN8, SN10, SN12, and SN15, the detectors for wavelengths λ8, λ10, λ12, and λ15 respectively on the first SWMR waveguide of the SWMR waveguide group are switched on when these SNs are in the RCS. At SN8, 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 SN8) from their corresponding resonating wavelengths in that node. The partial detuning of ring detectors of SN8 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 λ_{10}, λ_{12}, and λ_{15}, nodes SN10, SN12, and SN15 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.

4.3.2 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. 4.2(a), this MWSR waveguide group starts from the last SN (i.e., SN15) and traverses all of the remaining SNs (i.e., SN1-SN14) 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. 4.2(c) 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. 4.2(b) 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 4.5. 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 4.3, a power waveguide (see Fig. 4.2(a)) 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 4.6.

4.3.3 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 commu-nication via the MN. We illustrate this SN-to-SN communication with a simple example. When SN15 wants to send data to SN5, first SN15 sends data to the MN using an MWSR waveguide group, and then the MN sends the received data to SN5 using an SWMR waveguide group. We show the SN15-to-SN5 communication path in Fig. 4.2(a) 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 4.5, we describe how multiple MSNoC clusters are combined to form the BiGNoC architecture.

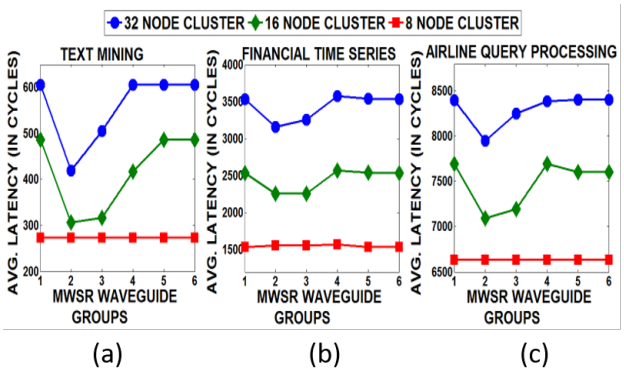


Figure 4.5: Variation of average packet latency in MSNoC cluster with (a) 32 nodes (b) 16 nodes, and (c) 8 nodes having different MWSR waveguide groups (each group has 4 waveguides) across three big data applications. *Reprinted with permission from* [8]

4.4 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 [99], and Airline Query Processing [100]. 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.

Fig. 4.5(a)g(c) 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 figures 5 (a)-(c), 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 wait-

ing 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 figures 5 (a)-(c) 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 multicluster architecture (BiGNoC) which we describe in detail in the next section.

4.5 BiGNoC Architecture

4.5.1 Homogeneous BiGNoC Architecture

In Section 4.3, 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 4.2 later in the paper), (ii) increase in average packet latency (see Fig. 4.5), 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 multicluster architecture (BiGNoC-HOM) with four uniform clusters represented as C0, C1, C2, and C3, as

shown in Fig. 4.6(a), where each cluster has 16 nodes (i.e., 64 cores). Each 16-node cluster in the

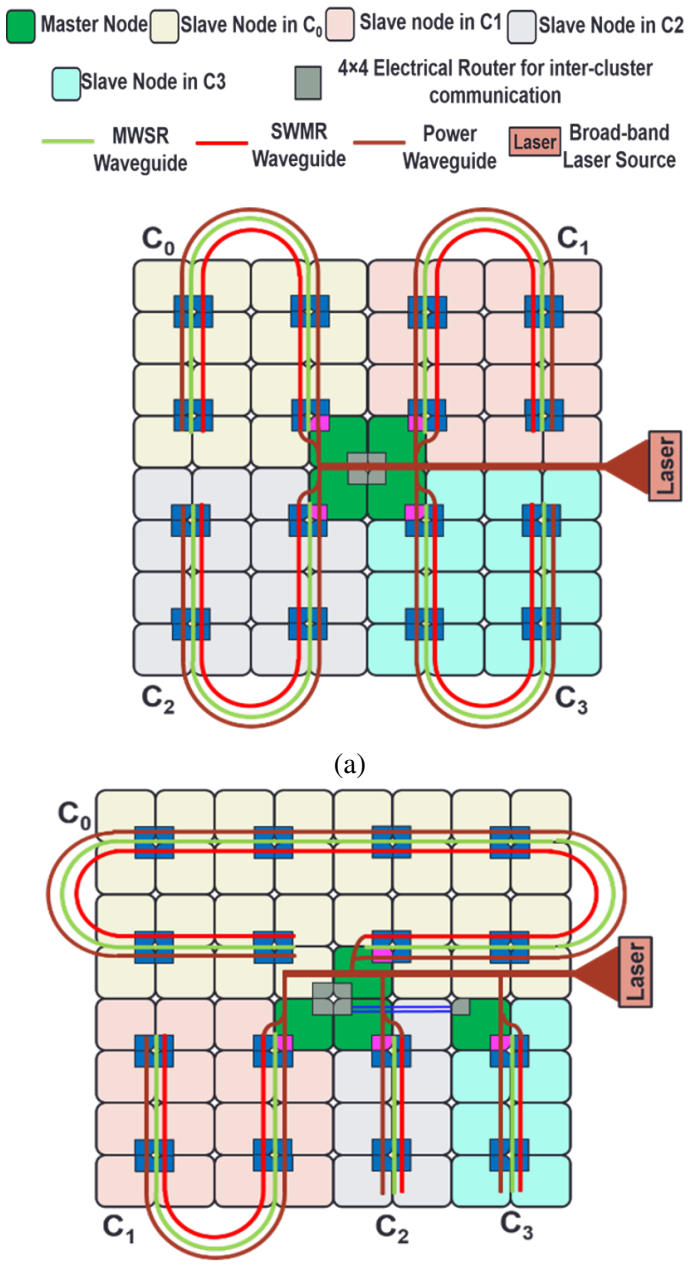


Figure 4.6: (a) Homogeneous BiGNoC with four uniform clusters C0, C1, C2, C3, with each cluster having 16 nodes, (b) Heterogeneous BiGNoC with four clusters C0, C1, C2, and C3 having 32, 16, 8, and 8 nodes, respectively. *Reprinted with permission from* [8]

BiGNoC-HOM architecture uses one SWMR waveguide group for MN-to-SN communication. As explained in Section 4.3.1, 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. 4.6, 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. 4.2(a)). This electrical router is used for intercluster 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, intercluster MN-to-MN communication occurs in one hop through the electrical router. Intercluster 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 intercluster SN-to-MN communication requires SN-to-MN (intra-cluster) and MN-to-MN (intercluster) hops. Further, intercluster 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 SNgto-SN communication across different clusters with a simple example. If node N2 (i.e., SN) of C0 needs to send a packet to node N10 (i.e., SN) of cluster C1, then N2 of C0 first sends data to N0

(i.e., MN) of C0 using an MWSR waveguide group. Then from this node the packet is sent to N0 (i.e., MN) of C1 through the electrical router that enables intergcluster communication. Lastly, the packet is sent to N10 of C1 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.

4.5.2 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 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 [79]. 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. 4.6(b).To enable receiver selection in SWMR waveguide groups of these clusters, we divided the waveguides in clusters C0, C1, C2, and C3 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 4.4, we use 2, 2, 1, and 1 MWSR waveguide groups for clusters C0, C1, C2, and C3 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 4.6.3.

4.5.3 Application scheduling in BiGNoC

Algorithm 1 shows the pseudocode 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 is more than or equal to the required number of master and servant cores for these applications. Applications (AP_i) are assumed to have master core (MA_i) and servant core (SA_i) requirements. The target BiGNoC platform is characterized by its clusters (C_j), master cores (MC_j), and servant cores (SC_j). First, the applications and BiGNoC platform clusters are sorted in the descending order of their SA_i and SC_j counts, respectively (steps 1-2). In steps 3-4, the algorithm initializes the required number of master cores (NMA_i) and servant cores (NSA_i) that are to be scheduled for each application, and also initializes the number of available free master cores (FMC_j) and free servant cores (FSC_j) in each cluster of BiGNoC, respectively. A nested loop iterates over all applications (AP_i) and clusters (C_j) in steps 5-6. If FSC_j are available in cluster 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 coreswithin 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.

4.6 Experiments

4.6.1 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) [101] as well as with two state-of-the-art photonic crossbar NoCs: Flexishare with token stream arbitration [82] and Firefly with a reservation assisted SWMR (R-SWMR) waveguide groups [81]. 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 [11], [12]-[13] (Table 4.2) 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 4.2 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

Algorithm 4 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)
Sort AP_i (highest SA to lowest SA)
Sort BigNoC clusters (highest SC to lowest SC)
for all i **do**
    NSA_i = SA_i
    NMA_i = MA_i
for all j **do**
    FSC_j = SC_j
    FMC_j = MC_j
for AP_i **do**
    for C_j **do**
        if FSC_j > 0 **then** *// Checks for free cores in clusters*
            if F **then**SCj NSAi 0
                Do-Scheduling ($AP_i \rightarrow NSA_i$ servant cores of C_j) *// Map servants*
                FSC_j = FSC_j - NSA_i
                NSA_i = 0
            if FMC_j > 0 and FMC_j $NMA_i \geq 0$ **then**
                Do-Scheduling ($AP_i \rightarrow NMA_i$ master cores of C_j) *// Map masters*
                FMC_j = FMC_j NMA_i
                NMA_i = 0
            else if FMC_j > 0 and FMC_j NMA_i < 0 **then**
                Do-Scheduling ($AP_i \rightarrow (NMA_i$ $FMC_j)$ master cores of C_j)
                NMA_i = NMA_i FMC_j
                FMC_j = 0
            else
                Do-Scheduling ($AP_i \rightarrow (NSA_i$ $FSC_j)$ servant cores of C_j)
                NSA_i = NSA_i FSC_j
                FSC_j = 0
            if FMC_j > 0 and FMC_j $NMA_i \geq 0$ **then**
                Do-Scheduling ($AP_i \rightarrow NMA_i$ master cores of C_j)
                FMC_j = FMC_j NMA_i
                NMA_i = 0
            else if FMC_j > 0 and FMC_j NMA_i < 0 **then**
                Do-Scheduling ($AP_i \rightarrow (NMA_i$ $FMC_j)$ master cores of C_j)
                NMA_i = NMA_i FMC_j
                FMC_j = 0
Output: Scheduled master-servant cores of app onto clusters of BiGNoC

(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 C0, C1, C2, and C3 of BiGNoCgHOM and BiGNoC-HET using the application scheduling algorithm presented in Section 4.5.3. We analyzed the actual execution characteristics of the big data applications presented in Table 4.2 (such as the master processing time, servant processing time, etc.) that are measured using an Amazons Elastic Compute Cloud (EC2) instance [102], 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.

Table 4.2: Big data application benchmarks, with three variations each, based on their Master-Servant requirements [11], [12]-[13]. *Reprinted with permission from* [8]

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)

We targeted a 22nm process technology for the 256-core system. Based on geometric calculations of the waveguides for a 20mm$\times$20mm 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., 1.6ns) 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, BiG-NoC-HET, Flexishare, and Firefly PNoCs. We considered an on-off switching time of 3.1 ps

for a ring modulator and ring detector [82], which is less than one clock cycle (i.e., 200ps) at 5GHz frequency.

The static and dynamic energy consumption of the electrical routers is based on results obtained from the DSENT tool [103]. 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 [14], [15]-[16] and shown in Table 4.3. Here $E_{dynamic}$ is the energy per bit for modulators and photodetectors and $E_{logicdyn}$ is the energy per bit for the driver circuits of modulators and photodetectors. PSWMRggFY and PMWMR-FX are the static power dissipation of SWMR and MWMR waveguide groups in Firefly and Flexishare architectures, respectively. Further, the PMWSR and PSWMR rows in Table 4.3 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 75mW, 35mW, and 15mW 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 [79]. All the static power dissipation values for waveguides presented in Table 4.3 include the power overhead of MR thermal tuning. We consider an MR heating power of 15 ţW per MR and detector responsivity of 0.8 A/W [15]. 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 0.0065mm2 and 0.008mm2, and power overheads of 0.12W and 0.16W in the electrical arbiters for the MWSR waveguide groups of BiGNoC-HOM and BiG-NoC-HET, respectively.

4.6.2 BigNoC: Sensitivity Analysis

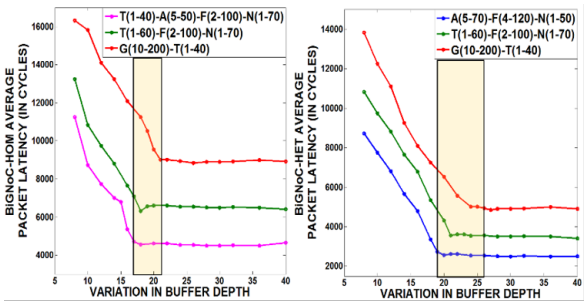
Our first set of experiments presents a sensitivity analysis to explore the optimal buffer size of the electrical router that is used for inter-cluster communication in two variants of our BiGNoC architecture with 256 cores: BiGNoC-HOM and BiG-NoC-HET. BiGNoC-HOM has four homogeneous clusters with each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

Table 4.3: Energy and losses for photonic devices [14], [15], [16]. *Reprinted with permission from* [8]

Cluster-wise static power per waveguide group of BiGNoC			
Waveguide Type	**32-Node Power**	**16-Node Power**	**8-Node Power**
PMWSR	1.54W	0.62 W	0.21W
PSWMR	5.72 W	2.69 W	1.26 W
Static power per waveguide group		**Power**	
PSWMR-FY		1.15 W	
PMWMRgFX		3.73 W	
Energy consumption type		**Energy**	
$E_{dynamic}$		0.42 pJ/bit	
$E_{logicdyn}$		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 90^0	

Fig.4.7(a) & (b) 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 [81], [82]. 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 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.4.7 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 pack-ets occurs



(a) (b)

Figure 4.7: Average packet latency comparison for (a) BiGNoC-HOM and (b) BiG-NoC-HET in a 256-core CMP with different buffer depths (8-40). *Reprinted with permission from* [8]

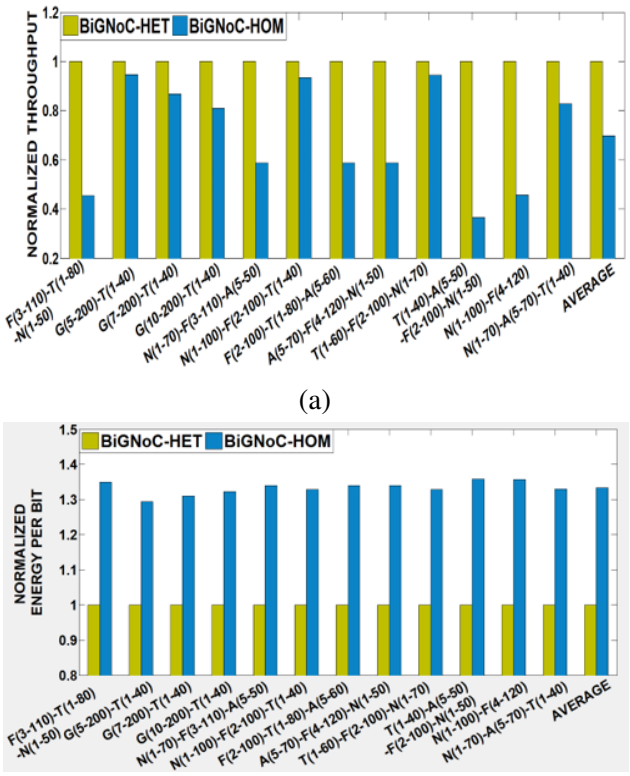
at the master nodes for workloads with higher inter-cluster traffic, which increases their queueing delay and average packet latency. Also, for all workloads executing on both BiGNoC-HOM and BiGNoC-HET, a smaller buffer size should intuitively result in higher average packet latency, as the buffer in 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 4.3.2) 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.4.7 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 Fig. 7 (a) and (b), 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, respectively, which are the highest buffer depths of the optimal performance regions shown in Fig. 4.7(a) & (b). We use these optimal buffer depths for BiGNoC-HOM and BiGNoC-HET in the rest of our analysis.

4.6.3 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. Fig. 4.8(a) & (b) 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.4.8(a) it can be seen that on an average BiGNoC-HET has 30.4% higher average throughput compared to BiG-NoC-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.4.8(b) it can be observed that on an average BiGNoC-HET has 33.3% 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 BiG-NoC-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.4.8, 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. 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 [82], and Firefly with R-SWMR

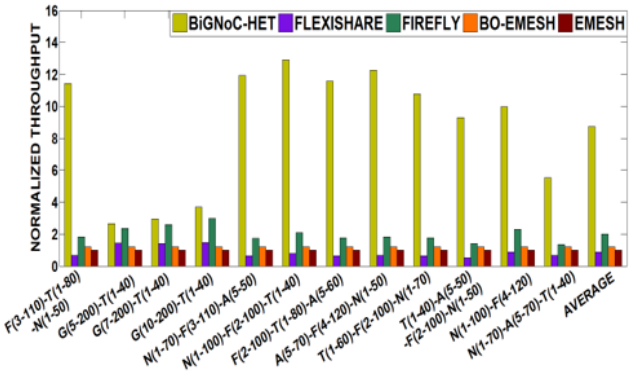


(a)

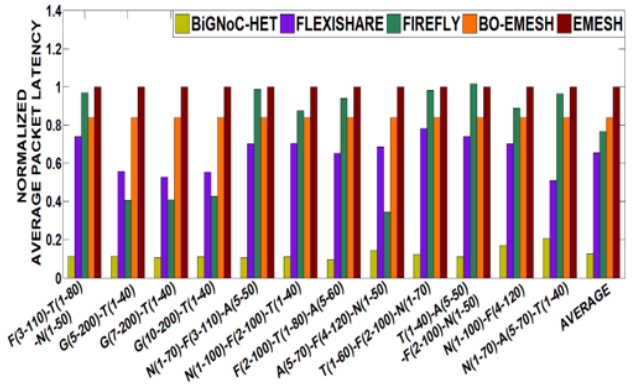
(b)

Figure 4.8: (a) Normalized throughput, (b) normalized EPB comparison of BiG-NoC-HOM with BiGNoC-HET for 256-core CMP. Results are shown for multi-application workloads and normalized w.r.t. BiGNoC-HET. *Reprinted with permission from* [8]

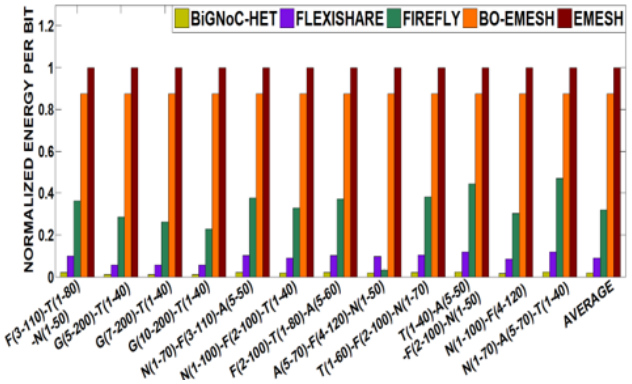
waveguide [81] architectures. Fig. 4.9 (a)-(c) show the results of this comparative analysis, where all the results are normalized with respect to the EMesh results. From the throughput com-parison in Fig. 4.9(a), it can be observed that, not surprisingly, BiGNoC-HET provides 8.7× and 7.2× higher throughput than EMesh and BO-EMesh, respectively, due to the presence of higher bandwidth photonic links for data communication. Furthermore, as shownin Fig.4.9 (a), 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 communi-cation), the bottlenecks at these routers limit BiGNoC-HET performance.



(a)



(b)



(c)

Figure 4.9: Normalized (a) throughput (b) latency (c) EPB comparison of BiG-NoC-HET with other architectures for a 256-core CMP. Results are for multi-application workloads and normalized w.r.t. EMesh. *Reprinted with permission from* [8]

BiGNoC-HET has nearly 9.9× 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 Flexishares 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.4.9 (b), by reducing the time spent waiting for access to the photonic waveguides. On average BiGNoC-HET has 81%, 84%, 85%, and 88% lower average packet delay over Flexishare, Firefly, BO-EMesh, and EMesh, respectively for the different multiapplication workloads.

Table 4.4: Phototnic hardware comparison. *Reprinted with permission from* [8]

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.4.9(c) shows the EPB comparison between the architectures. It can be observed that on average BiGNoC-HET has 88%, 90%, 96%, and 98% 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 4.4 shows the photonic hardware comparison between the PNoC architectures. It can be seen that BiGNoC-HET has 82% 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.

4.7 Chapter Summary

In this chapter, 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×, packet latency by up to 88%, and energy-per-bit by up to 98% 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. BigNoc addresses the communication aspect of large-scale computing. To address the computational challenges at a core-level, we envision to propose a photonic computing machine design. The next chapter demonstrates our invention of a large-scale deep learning photonic computer design.

5. NEUROMORPHIC COMPUTING USING SILICON PHOTONICS*

5.1 Introduction

[1]In today's era of big data, the volume of data that computing systems process has been increasing exponentially. Deep neural networks have become the state-of-the-art across a broad range of big data applications such as speech processing, image recognition, financial predictions, etc. Convolutional neural networks (CNNs) are a popular deep learning framework with superior accuracy on applications that deal with videos and images. However, CNNs are highly compute and memory intensive, requiring enormous computational resources. With Moores law coming to an end, traditional Von Neuman computing systems such as heterogeneous CPU/GPU platforms cannot address this high computational demand, within reasonable power and processing time limitations. Therefore, several FPGA [104] and ASIC [105] approaches have been proposed to accomplish large-scale deep learning acceleration.

A CNN comprises of two stages: training and inference (i.e., testing). Most hardware accelerators for CNNs in prior literature focus only on the inference stage. However, training a CNN is several hundred times more compute intensive and power intensive than its inference [106]. Moreover, for many applications, training is not just a one-time activity, especially under changing environmental and system conditions, where re-training of the CNN at regular intervals is essential to maintaining prediction accuracy for the application over time.

Training a CNN in general, incorporates a backpropagation algorithm which involves notable memory locality and compute parallelism. Recently, a few resistive memory (memristor) based training accelerators have been demonstrated for CNNs, e.g. ISAAC [105], PipeLayer [10]. ISAAC uses highly parallel memristor crossbar arrays to address the need for parallel computations in CNNs. In addition, ISAAC uses a very deep pipeline to improve system throughput. However, this is only beneficial when a large number of consecutive images can be fed into the architecture. Unfortunately, during training, in many cases, a limited number of consecutive images

[1]Part of this chapter is adapted with permission from [9]

need to be processed before weight updates. The deep pipeline in ISAAC also introduces frequent pipeline bubbles. Compared to ISAAC, PipeLayer demonstrates an improved pipeline approach to enhance throughput. However, both ISAAC and PipeLayer involve several analog-to-digital (AD) and digital-to-analog (DA) conversions which become a performance bottleneck, in addition to their large power consumption. Also, training in these accelerators involves sequential weight updates from one layer to another. This incurs inter-layer waiting time for synchronization, which reduces overall performance. This motivates an analog accelerator that can drastically reduce the number of AD/DA conversions, and inter-layer waiting time.

It has been recently demonstrated that a completely analog matrix-vector multiplication is 100× more efficient than its digital counterpart implemented with an ASIC, FPGA, or GPU [107]. HP labs have showcased a memristor dot product engine that can achieve a speed-efficiency product of 1000× compared to a digital ASIC [107]. Vandroome et al. in [108] have demonstrated a small-scale efficient recurrent neural network using analog photonic computing. A few efficient on-chip photonic inference accelerators have also been proposed in [109], [110]. However, a full-fledged analog CNN accelerator that is capable of both training and inference has yet to be demonstrated.

In this chapter, we propose a novel silicon photonics-based backpropagation accelerator for training CNNs. We present the design of this novel CNN accelerator (*BPLight-CNN*) that integrates the photonics-based backpropagation accelerator. *BPLight-CNN* is a first-of-its-kind memristor-integrated silicon photonic CNN accelerator for end-to-end training and inference. It is intended to perform highly energy efficient and ultrafast training for deep learning applications with state-of-the-art prediction accuracy. The main contributions of this chapter are summarized as follows:

- We propose BPLight-CNN, a fully analog and scalable silicon photonics-based backpropagation accelerator in conjunction with a configurable memristor-integrated photonic CNN accelerator design;
- We demonstrate a pipelined data distribution approach for high throughput training with BPLight-CNN;

- We synthesize the *BPLight-CNN* architecture using a photonic CAD framework (IPKISS [48]). The synthesized *BPLight-CNN* is used to execute four variants of VGG-Net and two variants of LeNet, demonstrating at least 35×, 31×, and 45× improvements in throughput, computation efficiency, and energy efficiency, respectively, compared to the state-of-the-art CNN accelerators.

5.2 Convolutional Neural Network: Overview

5.2.1 Basics of Convolutional Neural Network

Convolutional neural networks (CNNs) are a class of feedforward neural networks commonly used for analyzing visual imagery for image classification and object detection/prediction tasks. CNNs comprise of a sequence of hidden layers where each layer is composed of neurons arranged in three dimensions: width, height and number of channels. The neurons in a layer are connected to a small region of the layer before it. This ensures that weights are shared among the neural connections across adjacent layers, thereby reducing the number of parameters (weights) to be learnt in the network. The final output layer in a CNN is a fully connected neural network that transforms the full input image into a single vector of class scores arranged along the channel dimension.

Fig.5.1 illustrates an overview of CNN architecture. In principle, three types of layers are used to build a CNN: convolution layer (CONV), pooling layer (POOL) and a fully connected layer (FC). Generally, CONV is accompanied with a nonlinear activation function, such as ReLU. Depending on the sequence in which these layers are arranged, there are different CNN models, such as AlexNet [111], VGG [112], LeNet [113], GoogLeNet [114] etc. Fig. 5.1 illustrates an example of a CNN with [CONV-POOL]-[CONV-POOL]-[FC] i.e., 2 hidden layers each of which comprises of [CONV-POOL]. LeNet has the configuration [CONV-POOL]-[CONV-POOL]-[2FC] and VGG16 is built with [2CONV-POOL]-[2CONV-POOL]-[3CONV-POOL]-[3CONV-POOL]-[3CONV-POOL]-[3FC].

The functional details of the various layers are as follows.

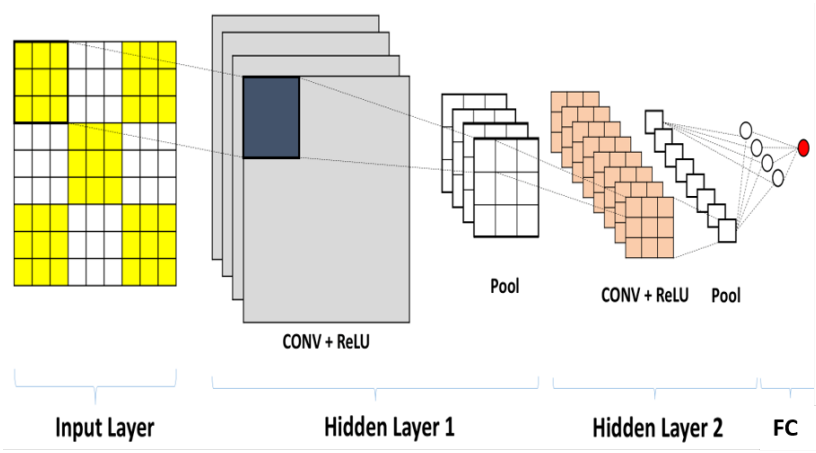


Figure 5.1: Overview of CNN with two hidden layers and an FC layer. Each hidden layer comprises of [CONV-POOL]

- **Convolution layer (CONV)** is used to extract features from the image using multiple filters. An M $\times$ N CONV receives M features as input and produces N features as output. It uses a set of M filters (or kernels), each of size $F_1 \times F_2$. Each of these filters slides across a corresponding feature with a stride of $S_1 \times S_2$ to perform element-wise vector matrix multiplication. The resulting N output features can be represented using the following equation.

$$Out[n][p][q] = \sum_{m=0}^{M} \sum_{i=0}^{F_1} W(n,p,q) \tag{5.1}$$

where,

$$W(n,p,q) = \sum_{j=0}^{F_2} [W[n][m][i][j] \times In[S_1 \times p + i][S_2 \times q + j]] \tag{5.2}$$

Here, n and m are kernel index, i and j are (x, y) values of a kernel, and p and q are (x, y) values of input In.

- **Neural activation layer** performs a biological activation function such as a sigmoid, rectified linear unit (ReLU), or tanh, on each feature of its previous layer. We utilize ReLU which is a widely used nonlinear activation function with state-of-the-art performance [104], [107],

[111], which can be described as follows.

$$ReLU(x) = max(0, x) \tag{5.3}$$

- **Pooling layer (POOL)** is used to obtain spatial invariance while scaling features from preceding layers. A "maximum/average of many features" approach is considered to scale down extracted features. POOL maintains translational invariance, or in other words, it results in a scaled-down feature map identical to its original version.

- **Fully Connected Network layer (FC)** performs the final classification or prediction in a CNN. An FC takes the feature maps generated from previous layers and multiplies a weight matrix following a dense matrix vector multiplication pattern. A few cascaded FC layers carry out the same procedure to produce the final classification or prediction output. The computation of an FC layer can be described by the following equation.

$$Out[p][q] = \sum_{m=0}^{N} [W[p][n] \times In[n][q] \tag{5.4}$$

5.2.2 Backpropagation Algorithm

A deep neural network such as a CNN has two stages: training and inference (testing). In the training phase, the filter weights (and biases) in CONV and FC layers are learnt by using a backpropagation (BP) algorithm. The BP algorithm involves a forward and a backward pass in the deep network. Given a training sample x in the forward pass, the weighted input sum (convolution) z is computed for neurons in each layer l with some initial filter weights w (and bias b) followed by neural activation $\sigma(z)$ ($ReLU(z)$ in our work), and POOL. The final layer L computes the output label of the overall network for every forward pass. This can be summarized as follows.

Forward pass: For each layer l,

$$z^{x,l} \leftarrow w^l a^{x,l-1} + b^l \tag{5.5}$$

$$a^{x,l} \leftarrow \sigma(z^{x,l}) \tag{5.6}$$

A cost function C is defined to quantitatively evaluate how well the output of a neural network at the final layer L compares to the target class label. The optimization goal in training is to minimize this cost function. The output error in the final prediction $\sigma x, L$ is a result of errors induced by the neurons in each hidden layer during the forward pass. To compute the error contribution of a neuron in the previous layer i.e., $\sigma(x, l)$, the final error is back propagated through the network starting from the output layer. This can be summarized as follows.

Output error: At the final layer L,

$$\sigma^{x,L} \leftarrow \nabla_a C_x \odot \sigma^{'}(z^{x,L}) \tag{5.7}$$

Backward pass: For each layer l,

$$\sigma^{x,l} \leftarrow ((w^{l+1})^T \times \sigma^{x,l+1}) \odot \sigma^{'}(z^{x,l}) \tag{5.8}$$

Here, ∇_a is gradient of $a^{x,l}$, and $\sigma^{'}(z^{x,L})$ is derivative of $\sigma(z^{x,L})$. These error contributions are necessary to update the filter weights w and biases b in the respective layers using a gradient descent method. In gradient descent, the forward and backward pass happen iteratively until the cost function is minimized and the network is trained. This can be summarized as follows.

Gradient Descent: For each layer l and m training samples with learning rate η,

$$w_n e w^l \leftarrow w^l - \frac{\eta}{m} \sum_x \sigma^{x,l} \times (a^{x,l-1})^T \tag{5.9}$$

$$b_new^l \leftarrow b^l - \frac{\eta}{m}\sum_x \sigma^{x,l} \tag{5.10}$$

Once the parameters of the model are learnt with the aid of the BP algorithm, recognizing an object in an image involves a simple forward propagation of a test image through a sequence of [CONV-POOL] hidden layers to extract the relevant features. Lastly, the feature maps flow through the FC layer which activates certain neurons in this dense network, to recognize the object it is trained for. Before discussing our proposed *BPLight-CNN* accelerator, the next section presents an overview of on-chip photonic components, which are the building blocks of this accelerator.

5.3 Overview: On-chip Photonic Components

On-chip photonics components such as photonic waveguides, microring modulators (MRMs), semiconductor-optical-amplifiers (SOAs), photodetectors and multi-wavelength laser sources are used for on-chip photonic signaling [115]. An MRM is a circular shaped photonic structure with a radius of 5 ţm which is used to modulate electronic signals onto a photonic signal at the transmission source in a waveguide. MRMs are also used to couple/filter out light from the waveguide at the destination. Each MRM modulates/couple light of a specific wavelength. The geometry of the MRM determines its wavelength selectivity. We can also inject (remove) charge carriers to (from) an MRM to alter its operating wavelength. An SOA is an optoelectronic device that under suitable operating conditions can amplify photonic signals. A detailed description of the structure, functionality, and modeling of SOAs is given in [116].

In a typical high bandwidth photonic link, an off-chip laser source (either on the board or on a 2.5D interposer) generates multiple wavelengths, which are coupled by an optical grating coupler to an on-chip photonic waveguide. The use of multiple wavelengths (e.g., 32) to transmit multiple streams of bits simultaneously is referred to as dense-wavelength-division-multiplexing (DWDM). To enable processing of these photonic signals, the on-chip photonic waveguide guides the input optical power of these DWDM photonic signals via a series of MRMs (where each MRM operates on a photonic signal with specific wavelength) and SOAs. Finally, the photonic signals arrive at

the destination where they are coupled out of the waveguide by MRMs, which drop the photonic signals onto photodetectors, to convert them back to electronic signals.

An important characteristic of photonic signal transmission in an on-chip photonic link is that it is inherently lossy, i.e., the photonic signal is subject to losses such as insertion losses in MRMs, active region losses in SOAs, detection losses in photo-detectors, and propagation and bending losses in waveguides. In addition, there are splitting and coupling losses in grating couplers, splitters, and multiplexers. Higher laser power is needed to compensate for the losses, for reliable photonic signal transmission. These photonic links are used to construct parts of our *BPLight-CNN* architecture, as discussed next.

5.4 BPLight-CNN Architecture

In this section, we discuss the architecture of the proposed *BPLight-CNN* accelerator for end-to-end training and testing. A high-level overview of proposed *BPLight-CNN* is shown in Fig.5.2. As shown in this figure, *BPLight-CNN* comprises of three parts: feedforward CNN accelerator architecture, backpropagation accelerator architecture, and weight update and peripheral circuitry. The rest of this section describes these three parts of *BPLight-CNN* in detail.

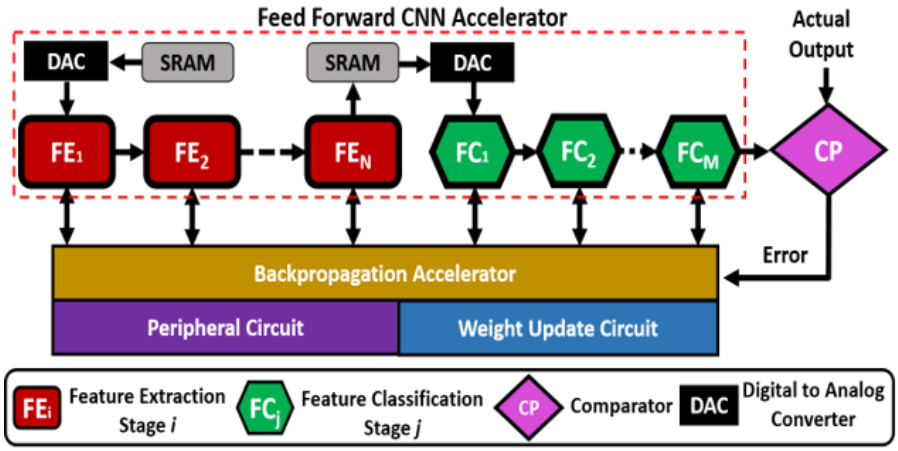


Figure 5.2: Overview of *BPLight-CNN* Architecture

5.4.1 Feedforward CNN Architecture

We consider an image dataset as input data and its classification as the application to be executed with BPLight-CNN. The CNN accelerator in the proposed *BPLight-CNN* architecture (see Fig.5.2) is used for feedforward feature extraction (FE) and feature classification of input images. The FE in the CNN architecture is carried out using multiple FE stages (FE_i). After all of the features are extracted, feature classification is performed using one or more fully-connected layers (FC). Fig.5.3 illustrates the microarchitecture of an FE stage. Each FE stage comprises of multiple memristor-based convolution layers (CONV), a semiconductor-optical amplifier (SOA)-based ReLU layer, an optical comparator based max-pooling (POOL) layer, and an interface layer. The detailed design is discussed in the following subsections.

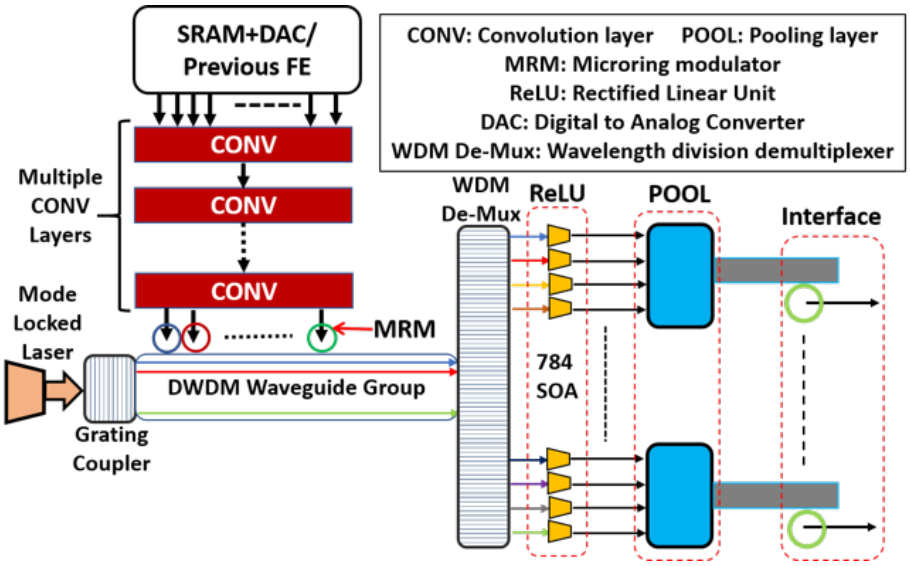


Figure 5.3: Microarchitecture of Feature Extractor (FE) in *BPLight-CNN*

5.4.1.1 CONV Microarchitecture

CONV is the first step for FE. As shown in Fig.5.3, there are multiple CONV layers in each FE. Each CONV layer has multiple weight memristor arrays (WMAs). The basic building block

of a WMA is a memristor. A memristor is a metal-oxide-based two-terminal electronic component [117], whose conductance G can be varied by external current flux. A detailed discussion on the operation of a WMA is presented in the following section. The first CONV layer receives analog data from an SRAM register through a DAC array or from the previous FE stage. An efficient pipelined approach is used to store input data (e.g., image pixels) in the SRAM, and this approach is discussed in Section 5.5. Intermediate CONV layers receive data from previous CONV layer and transfer data to the next CONV layer. Finally, an array of MRMs receives convolved data from the last CONV layer and modulates that information on carrier wavelengths to transfer this data to the ReLU layer (see Section 5.4.1.2) for further processing. In addition, FE process employs array of mode-locked lasers to produce lightwave carriers with different wavelengths.

The FE process in the *BPLight-CNN* architecture can convolve 56×56 image pixels in a complete cycle. The 56×56 input data is divided into 4 chunks of 28×28 input pixels. As explained earlier, before performing convolution, 28×28 input pixels stored in SRAM are converted to analog data using a DAC array. Moreover, an SRAM is connected to the DAC array using eight 128-bit memory buses. To enable conversion of 784 pixels (or 28×28), 13 64-channel DACs are employed. Four WMAs are used are in each CONV layer to process the analog data, where each WMA comprises of 38416 memristors. More information about performing convolution using WMAs is presented in subsections5.4.1.1(i.e., WMA reconfiguration) and 5.4.1.1(i.e., memristive convolution). Finally, FE utilizes 49 dense wavelength-division-multiplexing (DWDM) waveguides each carrying 64 wavelengths. This ensures simultaneous traversal of 4 28×28 pixels in a single cycle. In the following sections, the term waveguide group refers to the set of 49 DWDM waveguides.

- WMA Reconfiguration: A CNN model can use filters of different sizes such as 1×1, 2×2, 3×3, 4×4, 5×5, and 7×7 etc. The WMA in CONV can be configured to support filters of different sizes. Fig.5.4 demonstrates the WMA reconfiguration process to deploy 3×3 filters as an example. To configure a 3×3 memristor filter, memristors in a WMA are divided into multiple 9-memristor based memristor banks. The input of a memristor

is either connected to an analog output of a DAC from the DAC-array or is grounded for zero-padding which is required in the convolution. The output terminal of each memristor is connected to two electronic switches p and q. Similarly, to configure a 7×7 memristor bank, all the memristors in a WMA are divided into 784 memristor banks, each comprising of 49 memristors. This novel reconfiguration approach makes the proposed *BPLight-CNN* architecture flexible enough to emulate any CNN model.

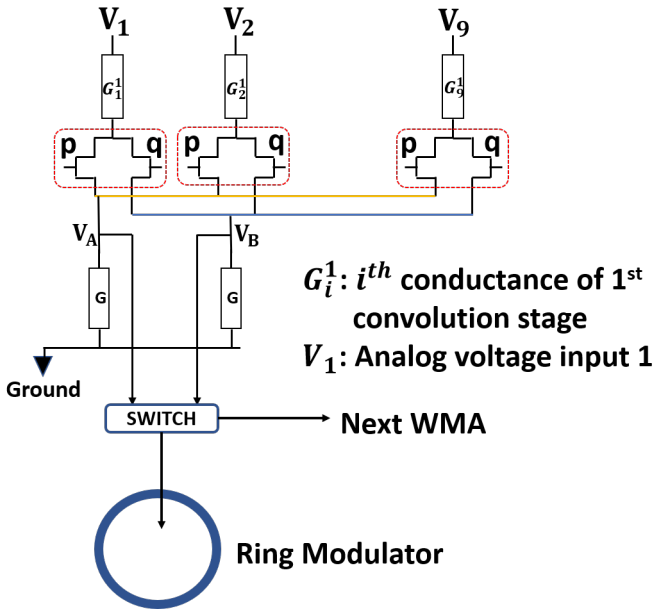


Figure 5.4: Memristive convolution in a CONV layer. *Reprinted with permission from* [9]

- Convolution in CONV: Fig.5.4 demonstrates memristive convolution using a 3×3 memristor bank as an example. Each memristor bank has 9 memristors with conductance G_1^C, G_2^C, G_9^C. A memristor can be programmed to carry up to 1000 states or conductance values [118]. We chose the value of each conductance G_i^C such that ($W_i^C = G_i^C$, $i = 1, 2, 9$), where W_1^C, W_2^C,W_9^C are elements of a 3×3 kernel. The kernel elements are chosen randomly in the beginning and then are updated by backpropagation during the training mechanism. The weight update mechanism was explained in Section 5.2.2.

Convolution with a memristor bank works as follows. Each weight value W_i^C can be a positive value or a negative value. As conductance of a memristor cannot be negative, a CMOS switch is used in our design to store negative weight in a memristor. If W_i^C is positive, switch p of the corresponding memristor (i.e., G_i^C) is set. If W_i^C is negative, switch q is set as shown in Fig. 5.4. Let the analog voltage inputs (i.e., input image pixels converted to analog data) to memristors G_1^C, G_2^C, G_9^C of the first memristor bank be V_1, V_2,V_9 respectively. In Fig. 5.4, currents from memristors carrying positive weights are accumulated in terminal A and currents from memristors carrying negative weights are accumulated in terminal *B* (Kirchhoffs law). The convolved output can be written as $(V_A - V_B)$ where V_A is voltage at A and V_B is voltage at B:

$$C_k = V_A - V_B \tag{5.11}$$

where C_k is the resulting voltage or the convolved output from the k_th memristor bank, $(V_A = I_p \times R)$ and $(V_B = I_q \times R)$, I_p is current accumulated from memristors through all switches marked as p and I_q is the current accumulated from memristors through all switches marked as q. The current values are:

$$I_p = \sum_{i=1}^{9} [I_{9*k+i} \times G_i^C] for W_i^C > 0 \tag{5.12}$$

Each convolved output C_k is fed to the peripheral circuit as it will be used by the backpropagation architecture later. Details of the peripheral circuit are discussed in the next section. Apart from the peripheral circuit, each C_k is input to a microring modulator (MRM) for data modulation. There are 784 MRMs each of which can modulate a lightwave in the DWDM waveguide. A modulated photonic signal with wavelength λ_k in a photonic waveguide can be expressed as:

$$L_k = C_k * A \sin(\frac{2\pi}{\lambda_k} t + \phi) \tag{5.13}$$

where L_k is modulated lightwave with wavelength λ_k, carrying convoluted data C_k, A is the

amplitude of the k_th lightwave before the data modulation phase. By setting $A = 1$,

$$L_k = C_k * A\sin(\frac{2\pi}{\lambda_k}t + \phi) \tag{5.14}$$

After data modulation, all the lightwaves L_k ($k = 1, 2, , 784$) are decoupled from the DWDM waveguide by a DWDM decoupler. After decoupling, each individual lightwave is fed to a semiconductor-optical-amplifier (SOA) in the ReLU layer.

5.4.1.2 ReLU Microarchitecture

As discussed in Section 5.3, an SOA is a silicon photonic component used to amplify a photonic signal. It has been demonstrated in [108] that an SOA can be used as a neural activation unit. An SOA uses an electronic pumping mechanism to provide gain to an input photonic signal. The electronic pump current to an SOA can be varied to set its total gain. The SOA characteristics is almost linear when an SOAs gain is close to 1. This linear behavior is identical to ReLU (see Eq.5.15) which is a widely used deep learning neural activation function. Therefore, we set the gain of all the SOAs in our design to 1. There are 784 SOAs in a ReLU layer of ***BPLight-CNN*** as shown in Fig.5.3. The k_{th} SOA takes lightwave L_k as input and produces the following output.

$$ReLU_k = \begin{cases} 0 & \text{if } C_k \leq 0 \\ L_k & \text{if } C_k > 0 \end{cases} \tag{5.15}$$

The outputs $ReLU_k$ of all SOAs are subsequently fed to the max-pooling layer, which is discussed next.

5.4.1.3 POOL Microarchitecture

The VGG [112] and LeNet [113] benchmarks that are used in this work operate on a 2×2 max-pooling with a stride of 2. Therefore, for max-pooling we consider a 2×2 window with a stride of 2. To facilitate 2×2 max-pooling for 784 outputs from the ReLU layer, the photonic max-pooling layer (POOL) uses 196 4-input max-pooling units such as P_1, P_2, P_3, P_196. Each max-pooling

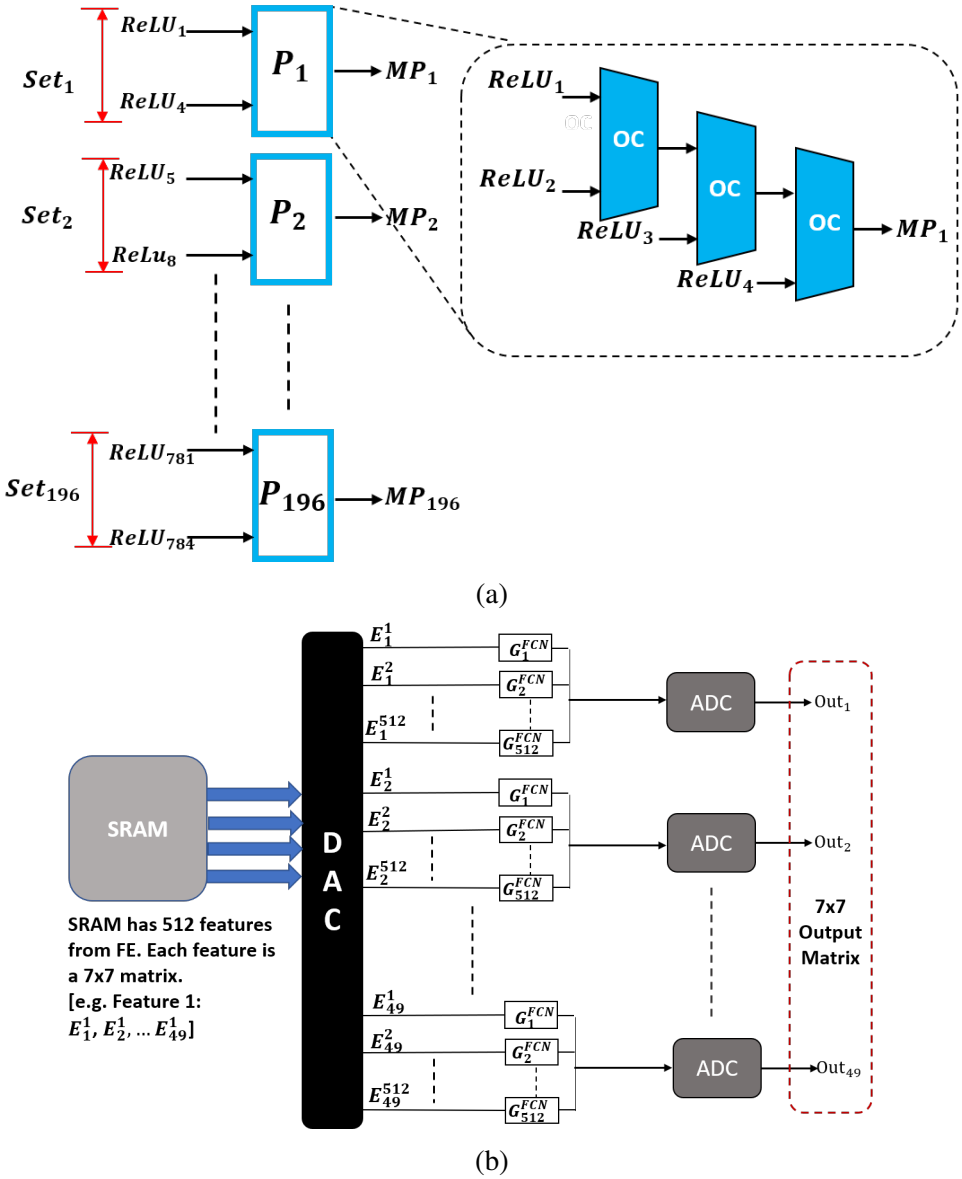


Figure 5.5: (a) Cascaded optical comparator in POOL, (b) Fully Connected Layer (FC). *Reprinted with permission from* [9]

unit consists of a cascaded optical comparator arrangement to perform max-pooling. As shown in Fig.5.5(a), three 2-channel optical comparators are cascaded to form a 4-input max-pooling unit. For benchmarks operating on higher order max-pool (e.g. 3×3), proportional number of optical comparators can be integrated to design the desired max-pooling unit. We consider a high-speed two-channel optical comparator identical to [119]. The 784 outputs from the ReLU layer

are bundled into 196 sets and each set j is fed to max-pooling unit P_j. Assuming $ReLU_1$, $ReLU_2$, $ReLU_3$, and $ReLU_4$ belong to set 1 and are input to P_1, the max-pooling output of unit P_j can be written as:

$$MP_j = max(ReLU_{4(j-1)+1}, ReLU_{4(j-1)+2}, ReLU_{4(j-1)+3}, ReLU_{4(j-1)+4}) \tag{5.16}$$

5.4.1.4 FC Microarchitecture

After feature extraction is performed using the FE stages (by using the CONV, ReLU, and POOL microarchitectures discussed in the previous subsections), features are sent to a feature classification phase. In CNN, the feature classification segment can be viewed as a special case of convolution, where each extracted feature map uses the largest possible kernel. In other words, feature classification comprises of one or more fully-connected (FC) layers. All the layers discussed in the previous subsections are used for feature extraction, whereas the FC layer performs feature classification.

BPLight-CNN employs a photonic matrix vector multiplication (P-MVM) based FC layer. The working principle of P-MVM based FC is similar to that of memristive convolution. Fig.5.5(b) illustrates a logical layout of a P-MVM based FC layer. When all the features from the feature extraction stage are stored and available in the SRAM buffer, the features are fed to the DAC array of the FC layer as depicted in the figure. As an example, we consider 512 features coming from the feature extraction (FE) stages. VGG and LeNet operate on a 7×7 kernel in FC. Therefore, each feature is a 7×7 matrix, e.g. the ith feature is E_1^i, E_2^i, E_49^i. FC has 49 identical memristor banks each of which has 512 memristors: G_1^{FC}, G_2^{FC}, G_{512}^{FC}. Each memristor G_i^{FC} represents an FC weight W_i^{FC} which is obtained through offline training. After analog conversion, each analog value of a feature is applied as voltage across a memristor of the FCs memristor bank. For example, a voltage of E_1^i is applied across G_1^{FC}, voltage of E_2^i across G_2^{FC}, etc. Depending on whether the weight W_i^{FC} corresponding to memristor G_i^{FC} is positive or negative, the p or q switch is set respectively, (ref: Fig.5.4). The accumulated current from each memristor bank is

fed to the next FC stage until the final FC stage is reached. Also, outputs from each FC stage are fed to the peripheral circuit to be used by the backpropagation architecture for weight update. The outputs from the final FC stage are the classified outputs for a feedforward CNN. During training, the classified outputs and target outputs are input to an analog subtraction unit, the result of which is fed to the backpropagation architecture, as discussed next.

5.4.2 Backpropagation Architecture

The backpropagation (BP) architecture mainly involves computing matrix-vector multiplication in the backward pass. A photonic modulator can be used for analog amplitude modulation of a light carrier. In its simplest term, analog amplitude modulation is the multiplication of a scalar input with an analog signal. The authors in [115], [120] have demonstrated photonic modulator based analog multipliers. Fig. 5.6 illustrates the microarchitecture of the proposed BP accelerator design. It is based on photonic matrix-vector multiplication using MRMs (which were discussed in Section 5.3). We use MRMs for their high accuracy and quality factor.

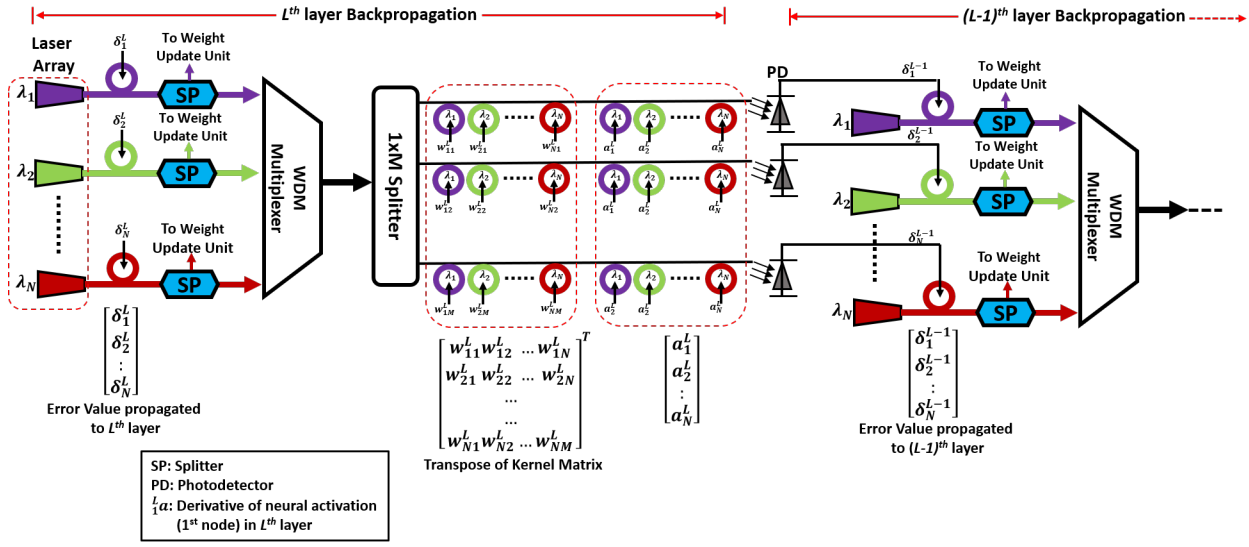


Figure 5.6: Backpropagation architecture in *BPLight-CNN* which presents the backpropagation between the final layer $l = L$ and penultimate layer $l = L - 1$

We now describe the operation of the proposed BP architecture. As discussed in Eq.5.7, the error at the final layer ($l = L$) of BP is $\delta^{x,L} \leftarrow \nabla_a C_x \odot \sigma'(z^{x,L})$. Here, $\nabla_a C_x$ is rate of change of output w.r.t the output activation (i.e., difference of actual classified output from FC of CNN architecture and the target output). $\sigma'(z^{x,L})$ is the derivative of the ReLU function in the final FC stage of the CNN architecture. Outputs from the final FC stage of the CNN architecture are fed to an analog subtraction and multiplication unit to determine $\delta^{x,L}$. Using Eq.5.8 and the computed $\delta^{x,L}$, we calculate error for the $(L-1)^{th}$ layer using the following equation:

$$\delta^{x,L-1} \leftarrow ((w^L)^T \times \delta^{x,L}) \odot \sigma'(z^{x,L-1}) \tag{5.17}$$

where, w^L is weight matrix obtained from L^{th} layer of CNN architecture through the peripheral circuit. The details of how the peripheral circuit is explained in the next subsection. Fig.5.6 shows the backpropagation between the final layer $l = L$ and its penultimate layer $l = L-1$. As illustrated in the figure, there are N number of wavelength carriers coming from a mode-locked laser array. The value of N for a layer equals to the output feature size for the corresponding layer in the CNN architecture, e.g. N equals 49 (7×7) for the last layer. Each wavelength in layer l is modulated with error $\delta^{x,L}$ by an MRM tuned to that wavelength. In Fig.5.6, the violet MRM is tuned to modulate λ_1. Now the jth MRMs output is $MRM_j = \delta_j^{x,L} * A \sin(\frac{2\pi}{\lambda_j}t + \phi)$. Each MRM_j is split into two parts. The first part is sent to the weight-update circuitry (see Section 5.4.3) to update the corresponding weights in the CNN architecture. The other part is fed to a WDM multiplexer. A WDM multiplexer is used to combine multiple light wavelengths into a single multi-wavelength carrier. After multiplexing, the combined optical signal is split into M parts where M equals the number of neurons in layer $L-1$. Each part is fed to a multi-wavelength waveguide. As a result, in each waveguide there are N wavelengths each carrying data $\delta_{j,n}^{x,L} * B \sin(\frac{2\pi}{\lambda_j}t + \phi)$, where $1 \leq n \leq N$, $B = \frac{A}{2N}$. Each weight w_{ij}^L of the transpose of w^L obtained from the peripheral circuit is modulated to a light carrier. This results in:

$$M_{i,n} = w_{ij}^{L} * \delta_{j,n}^{x,L} * A\sin(\frac{2\pi}{\lambda_j}t + \phi) \tag{5.18}$$

Now, each $M_{i,n}$ is modulated with a_n^L which is a derivative of the ReLU functions of layer $L-1$ (equal to $\sigma'(z^{x,L-1})$ in Eq.5.17). Then, $M_{i,n}$ becomes,

$$M_{i,n} = w_{ij}^{L} * \delta_{j,n}^{x,L} * a_n^L * A\sin(\frac{2\pi}{\lambda_j}t + \phi) \tag{5.19}$$

Next, a photodiode is used to demodulate photonic data from each waveguide. The photodiode demodulates the combined output $M_(i,n)$ for all wavelengths in a waveguide which is nothing but the matrix-vector multiplication identical to Eq.5.17. The output of each photodiode is passed through a signal conditioning and filtering circuit to remove unwanted noises. The output from the signal conditioning circuit is:

$$\delta^{x,L-1} = ((w^L)^T \times \delta^{x,L}) \odot a^L \tag{5.20}$$

where, $\delta^{x,L-1}$ is the error to be propagated from layer $(L-1)$ to $(L-2)$. This procedure is continued until the 1^{st} layer is reached. While doing the backpropagation, the error value in each layer is also fed to the corresponding weight-update circuit, which is discussed in more details in the next section.

5.4.3 Weight Update and Peripheral Circuitry

5.4.3.1 Weight-update circuitry

For weight-update, each element of a weight kernel in any layer l of CNN architecture can be written as $w_{k,j}^l$. Each $w_{k,j}^l$ is stored in a memristor of a memristor bank in layer l as $G_{k,j}^l$ (as explained in Section 5.4.1.1). The weight-update equation for $w_{k,j}^l$ (or, $G_{k,j}^l$) can be written as per Eq.5.9, as follows:

$$^{new}G_{k,j}^l \leftarrow {}^{old}G_{k,j}^l - \frac{\eta}{m} \times \delta_k^l \times O_j^{l-1} \tag{5.21}$$

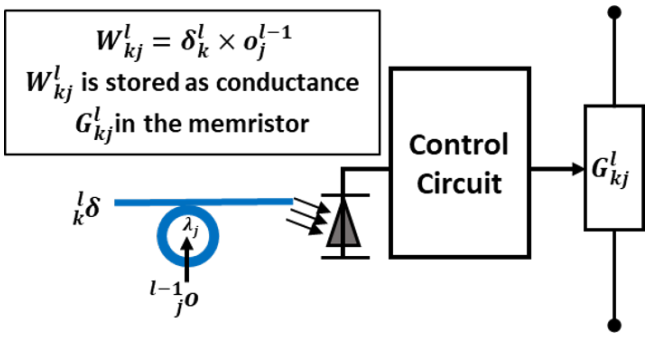


Figure 5.7: Weight Update Circuitry for any layer l

where, O_j^{l-1} is the j_{th} output from the POOL of the $(l-1)$ layer of the CNN architecture. Fig.5.7 illustrates the weight update circuitry for a layer l of *BPLight-CNN*. As shown in the figure, δ_k^l is obtained from the BP architecture as a photonic signal. O_j^{l-1}, which is collected from the peripheral circuit, is used to modulate the light carrier carrying the error value δ_k^l. The modulated output is demodulated using a photodiode and then sent to a signal conditioning circuit. In the signal conditioning circuit, first the analog signal is filtered (from noises) and passed through a subtractor to obtain new $G_{k,j}^l$ as depicted in Eq. (20). The previous conductance or weight value ${}^{old}G_{k,j}^l$ is fed to the subtractor from the lth layer memristor bank. The new conductance value ${}^{new}G_{k,j}^l$ is now fed to the equivalent memristor control circuit to update its weight value. The conditioning circuit as well as the memristor control circuit are inspired from [106].

5.4.3.2 Peripheral Circuitry

The output MP_j from the POOL of a layer l can be written as MP_j^l. During the feedforward training phase, each MP_j^l is stored as conductance in a memristor in the peripheral circuitry. This is used in backpropagation as O_j^l, an output of the l^{th} layer (as per Eq.5.21). Each MP_j^l is sent to a signal conditioning circuit and then a memristor control circuit. The resulting electronic signal is used to update the conductance (or weight value) of the memristor.

5.5 BPLight-CNN Case Study

In this section, we demonstrate the working principle of a pipelined *BPLight-CNN* architecture for a CNN benchmark VGG [112] on the ImageNet dataset [121]. We select a particular configuration, namely, VGG-A for the case study. However, we also experiment with all variants of the VGG [112] and LeNet [113] benchmarks as shown in Fig. 5.9 and discussed in Section 5.6. Using microarchitectures of the convolution layer, ReLU layer, POOL layer, interface layer, and FC layer as explained in Section , we configured *BPLight-CNN* as illustrated in Fig.5.8(a) for VGG-A application with four FE stages. The details of it are as follows.

VGG for the ImageNet dataset operates on a 224×224 image input. As explained in Section 5.4.1.1, *BPLight-CNN* can convolve 56×56 pixels at a time, i.e., one *BPLight-CNN* cycle. Therefore, it requires 16 *BPLight-CNN* cycles to execute a 224×224 image. Please note that a *BPLight-CNN* cycle is different from its clock cycle. Here, one *BPLight-CNN* cycle refers to the complete feature extraction and feature classification of a 56×56 image. The SRAM register array in *BPLight-CNN* is of size 2 KB to store the 56×56 input data. CONV performs feature extraction on a 28×28 input data at a time in a pipelined manner. FE in *BPLight-CNN* is performed as explained in the CONV architecture (Section 5.4.1.1).

Fig.5.8(b) demonstrates the pipelined dataflow of the feedforward operation in BPLight-CNN. We consider a 2.5 GHz clock. Therefore, the clock cycle period T_{sm} = 400 ps. As shown in Fig.5.8(b), at $t = T_{sm}$, the first set of 28×28 pixels from SRAM (i.e., A) are convolved (64 filters/features) and are stored in memristors in the peripheral circuit. The other three set of 28×28 pixels are namely, B, C, and D. Note that CONV convolves a 28×28 input in one clock cycle. As FE-1 for VGG-A consists of one convolution layer (Fig. 5.9), convolved outputs of CONV-1 of FE-1 is directly sent to the modulation phase. In the modulation phase, each convolved output is modulated by an MRR of a particular tuning wavelength to a light carrier of that wavelength in the DWDM waveguide group. This means that it can accept 4 28×28 features. The time required for convolved data of one FE to arrive at the next FE, T_{FE} = modulation time + ReLU time + POOL time + interface time = 20 ps + 10 ps + 10 ps + 10 ps = 50 ps. From $t = T_{sm}$ to $t = 2T_{sm}$,

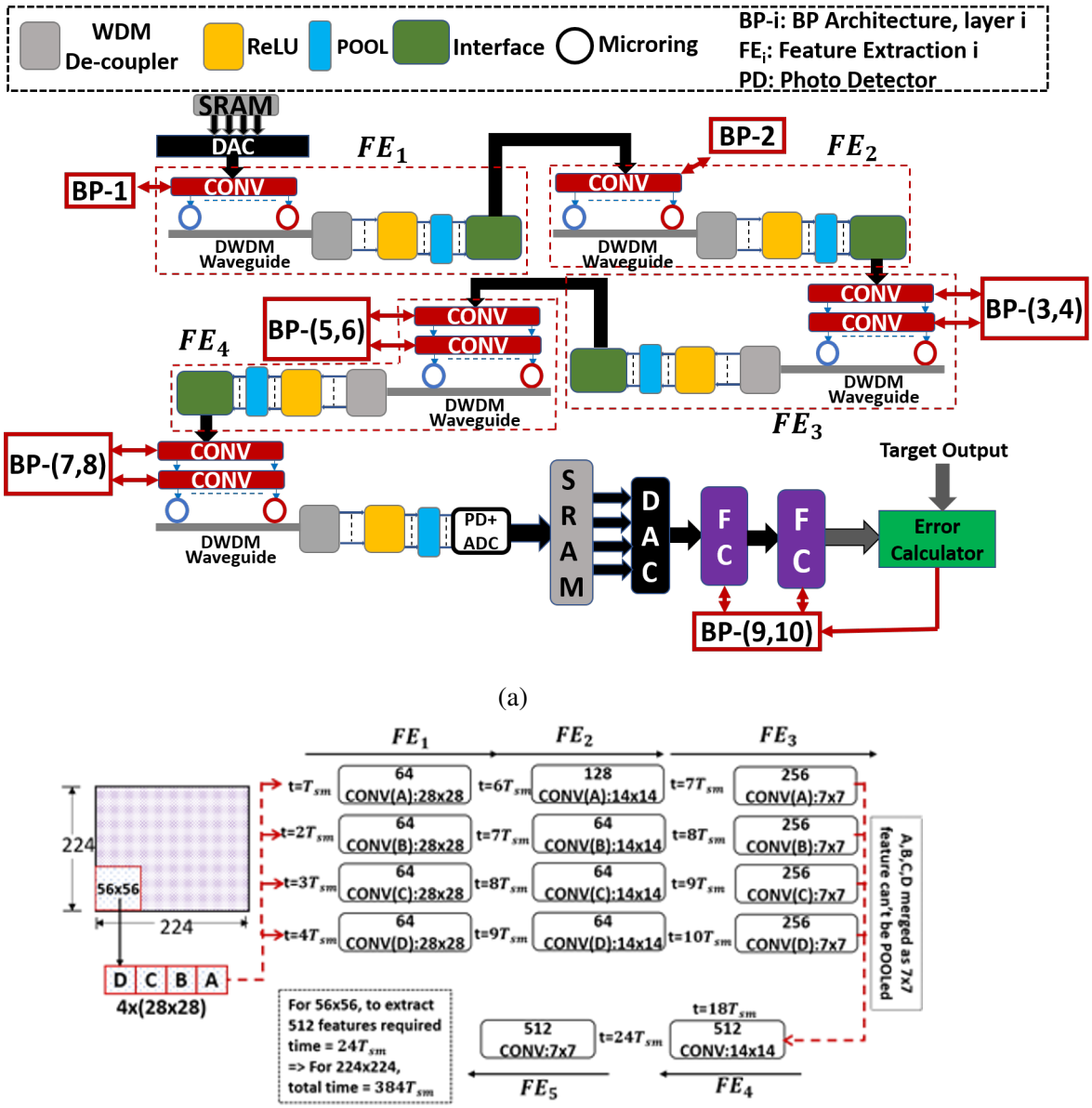


Figure 5.8: (a) VGG-A implemented on *BPLight-CNN* (b) Pipelined dataflow in feedforward operation in *BPLight-CNN*

CONV(A) outputs from the peripheral circuit of FE_1 are modulated, ReLU and POOL'ed, and then fed to FE_2. There can be 8 such data movements as $\frac{T_{sm}}{T_{FE}} = 8$. In one data movement, 4 28×28 features can be processed. Therefore, at $t = 2T_{sm}$, 32 CONV(A) features arrive at FE_2. Also, at $t = 2T_{sm}$, B from SRAM is convolved and stored in the peripheral circuit of FE_2. Similar

to CONV(A), from $t = 2T_{sm}$ to $t = 3T_{sm}$, 32 CONV(B) features are convolved and stored in the peripheral circuit of FE_2. In this way, at $t = 6T_{sm}$, all the 64 CONV(A) features in FE_1 are convolved in FE_2 (128 features) and stored in the memristors of its peripheral circuit. Similarly, B, C, and D are convolved and stored (Fig.5.8(b)). Note that FE_1 has 64 features, FE_2 has 128 features, FE_3 has 256 features, etc, as per the VGG-A configuration.

A, B, C, and D are convolved separately until $t = 10T_{sm}$ when all of them arrive at FE_3 as 256 7×7 features each. Now, all of these features are merged together to form 256 28×28 features. Therefore, it will require another $8T_{sm}$ time (i.e., $t = 10T_{sm}$ to $t = 18T_{sm}$) to send 256 28×28 features from FE_3 and convolve them as 512 14×14 features at FE_4. Similarly, convolution, ReLU, and POOL are performed in FE_4 and FE_5. As illustrated in Fig.5.8(b), at $t = 24T_{sm}$, 512 features are obtained from FE_5 for 56×56 pixels. As shown in Fig.5.8(a), features from FE_5 are stored in SRAM until all the 224×224 pixels are extracted. For 224×224 pixels, it will take $16 \times 24T_{sm} = 384T_{sm} = 153.6$ns. After this, all the features are retrieved from SRAM and fed to FC for feature classification. The first FC operation requires $(T_{sm} + T)$ time as it is identical to FE. The second FC operation requires T time as no more SRAM read is needed. This means that *BPLight-CNN* requires 153.6ns (for FE) $+T_{sm} + 2T = 154$ns, for one forward pass.

After a forward pass, the FC output is sent to the BP architecture for backpropagation. Each layer in BP requires T_b units of time where T_b = (error modulation to light carrier) + (split time) + (WDM multiplexing time) + (split time) + (weight modulation time) + (ReLu function derivative modulation time) + (photodiode time) = 10 ps + 10 ps + 10 ps + 10 ps + 10 ps + 10 ps +20 ps = 80 ps. It takes $6T_b$ units of time to complete one backward pass. In summary, *BPLight-CNN* requires 154 ns for one forward pass and 80 ps for a backward pass. The ultra-fast nature of photonic interconnects allows for high-speed backpropagation in *BPLight-CNN*.

5.6 Experimental Analysis

5.6.1 CAD for *BPLight-CNN*

We use IPKISS [48], a commercial optoelectronic CAD tool, to design and synthesize all of the photonic components of *BPLight-CNN*. All of the synthesized components are integrated together to design BPLight-CNN. For all of the photonics components, we consider a 32nm IPKISS library. The parametric details for *BPLight-CNN* are shown in Table 5.1. We use Caphe [48], a python-based photonic system validation tool, to estimate power, area, and performance of *BPLight-CNN* accelerator for several benchmarks.

5.6.1.1 Power and Area Models

The power and area of all *BPLight-CNN* components are summarized in Table 5.1. We use Caphe [48] for modeling power and area of all photonic elements such as modulators, demodulators, waveguides, lasers, etc. The energy and area parameters for memristors are adapted from [118]. We use integration and fire mechanism-based DAC identical to PipeLayer [10] in our design. The power and area models are adapted accordingly from PipeLayer. We also use power and area parameters from [106] for the ADC array used in the FC layer of *BPLight-CNN*.

5.6.1.2 Performance Models

We use Caffe [122], a deep learning framework, to train the datasets in conjunction with photonic component results from IPKISS. We manually map each of our benchmarks in waveguides, max-pool, buffers, and FC of BPLight-CNN. This ensures zero pipeline hazards between any two layers in BPLight-CNN. We determine computational efficiency, energy efficiency, throughput, and prediction error rate to compare the performance of *BPLight-CNN* with a state-of-the-art CNN accelerator, namely PipeLayer [10]. We also use GPU results (from [10]) as the baseline for comparison. We evaluate for the following metrics: *Computational efficiency* represents the total number of fixed point operations performed per unit area in one second (GOPS/s/mm2); *Energy efficiency* refers to the number of fixed point operations performed per watt (Giga operations per watt or GOPS/W); *Throughput* is the total number of operations per unit time (GOPS/s); and lastly,

Table 5.1: *BPLight-CNN* parameter details

Component	Parameters	Values	Power (mW)	Area (mm^2)
SRAM	Size	2KB	10	0.2
	Count	16		
DAC	Resolution	8-bit		
	Frequency	1.2 Gbps		
	Channel	64		
	Count	208	4.374	0.000208
ADC	Resolution	8-bit		
	Frequency	1.2 Gbps		
	Count	245	490	0.294
WMA in CONV	Count	48	24.5	0.000514
WMA in FC	Count	49	0.14	0.000003
Modulator	Switch Time	20ps		
	Count	62720	140	0.0009
De-modulator	Switch Time	20ps		
	Count	62720	140	0.0009
WDM Coupler	Count	16	0	0.00028
WDM Decoupler	Count	16	0	0.00028
Optical Comparator	Response Time	60ps	0	0.0045
Mode-locked laser	Wavelengths	64		
	Count	196	35000	0.384
Waveguide	DWDM	64 channels		
	Width	450nm		
	Count	32	0	20

Prediction error rate is the percentage of error in inferring any datasets.

5.6.1.3 Benchmarks and Datasets

We use two widely used CNN benchmarks: VGG-Net [108] and LeNet [121]. We consider four variants of the VGG benchmark: VGG-A, VGG-B, VGG-C, and VGG-D and two variations of LeNet (LeNet-A and LeNet-B). The configuration of all stages of VGG and LeNet benchmarks for these variants are depicted in Fig.5.9. In the table, CONV-I represents convolution stage I for a benchmark model. M×M, K, N for a convolution stage means that the convolution stage comprises of M×M filters, and N number of back-to-back convolution layers, with each convolution layer having convolutional width K. The convolutional width is the number of convolutional filters in

a convolution layer. Furthermore, we do consider a unit size window stride for the benchmark variants. For VGG, we use ImageNet dataset [121] having 224×224 images. For LeNet, we use 60,000 224×224 images of MNIST datasets [123] for training and 10,000 224×224 images for testing.

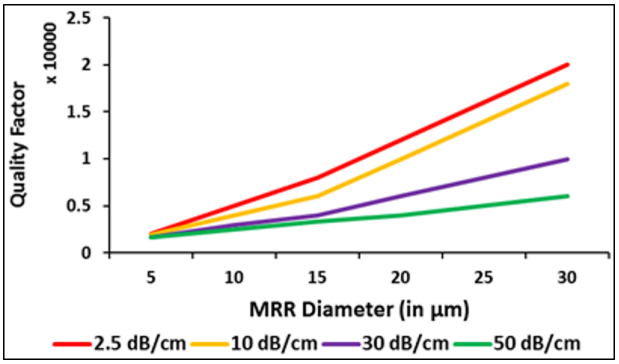
	FE_1	FE_2	FE_3	FE_4	FE_5	
VGG-A	3×3, 64, 1	3×3, 128, 1	3×3, 256, 2	3×3, 512, 2	3×3, 512, 2	FC-4096,2 FC-1000, 1
VGG-B	3×3, 64, 2	3×3, 128, 2	3×3, 256, 2 1×1, 256, 1	3×3, 512, 2 1×1, 256, 1	3×3, 512, 2 1×1, 256, 1	
VGG-C	3×3, 64, 2	3×3, 128, 2	3×3, 256, 3	3×3, 512, 3	3×3, 512, 3	
VGG-D	3×3, 64, 2	3×3, 128, 2	3×3, 256, 4	3×3, 512, 4	3×3, 512, 4	
LeNET-A	3×3, 6,1	3×3, 6,1	3×3, 16,2	3×3, 16, 4	3×3, 120, 1	FC84,1
LeNET-B	3×3, 6,1	3×3, 6,1	3×3, 256, 1	3×3, 16,6	3×3, 120, 1	

Figure 5.9: CNN Benchmark Configuration for VGG & LeNeT

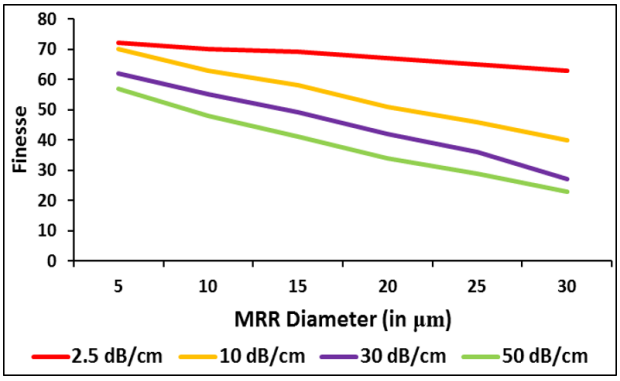
5.6.2 Sensitivity Analysis with Prediction Accuracy

MRMs are used extensively in the *BPLight-CNN* design, both in the feedforward and BP architectures. The prediction accuracy of *BPLight-CNN* depends on losses encountered by the photonic signal when it traverses through the photonic waveguide, MRMs, and other photonic components. These losses degrade photonic signal intensity before it reaches SOA (which acts as ReLu in *BPLight-CNN*), and causes the SOA to operate in a non-linear region, reducing the overall prediction accuracy. Among all of the losses, the MRMs insertion loss and waveguide propagation loss are the major contributors to prediction error in BPLight-CNN. The MRMs insertion loss depends on its Quality factor (Q-factor) and finesse. Q-factor is the number of photonic cycles taken by a photonic signal before its intensity goes to zero in an MRM. Finesse is the number of photonic cycles before a photonic signals intensity becomes 70.7% of its initial value. As both Q-factor and

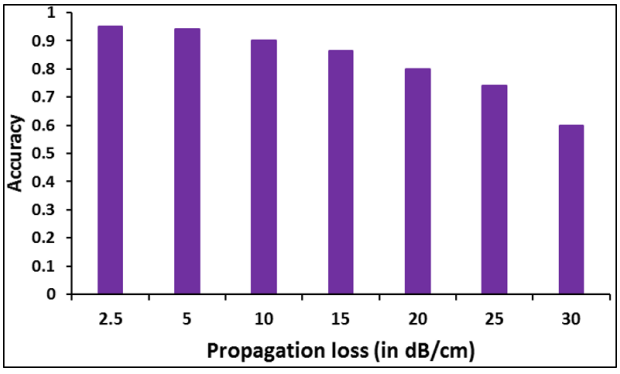
finesse are determined by the MRM diameter, therefore, in this section we present a sensitivity analysis to determine the optimal MRM diameter.



(a)



(b)



(c)

Figure 5.10: (a) MRM Q-factor (b) MRM Finesse (c) average prediction accuracy w.r.t propagation loss in photonic components diameter (assuming a 32-bit weight resolution).

Fig.5.10(a) and (b) illustrate the MRMs Q-factor and finesse w.r.t. its diameter (in μm), respec-

tively. From Fig.5.10(a), it can be seen that increase in MRM diameter leads to higher Q-factor, which ultimately leads to lower insertion loss. On the other hand, from Fig.5.10(b) it can be observed that increase in MRM diameter decreases its finesse, and increases insertion loss. Therefore, we select an optimal MRM of diameter of 10μm to minimize overall insertion loss. Considering this MRM diameter, Fig.5.10(c) presents average prediction accuracy variation with increase in waveguide propagation loss (in dB/cm) for all the applications discussed in Section 5.6.1.3. In this analysis, we have considered photonic waveguide groups of fixed lengths in different parts of the *BPLight-CNN* architecture, where each waveguide in a waveguide group is coupled with a fixed number of MRMs with 10μm diameter. From this plot it can be seen that increase in photonic waveguide propagation loss decreases prediction accuracy. An increase in waveguide propagation loss decreases photonic signal integrity and decreases predication accuracy. In addition, increase waveguide propagation loss also increases insertion losses of MRMs which increases overall losses and worsens prediction accuracy further. Therefore, we have considered the lowest propagation loss of 2.5 dB/cm [124] for the rest of our analysis.

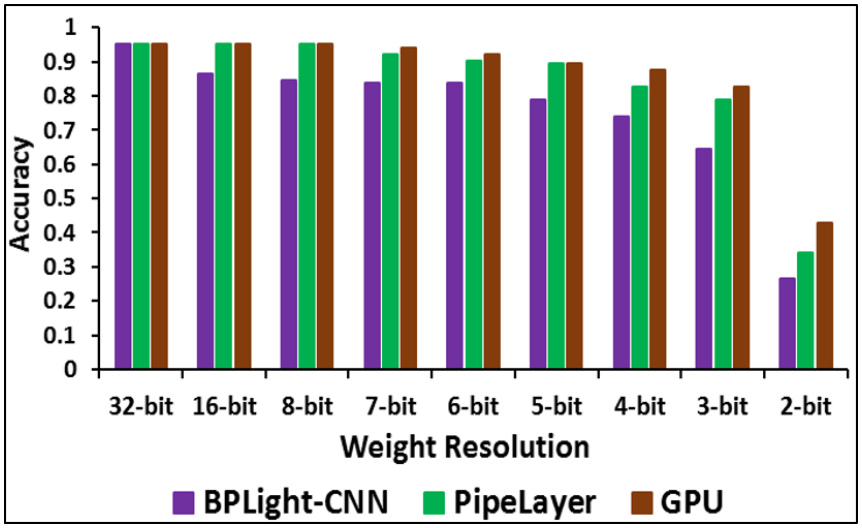


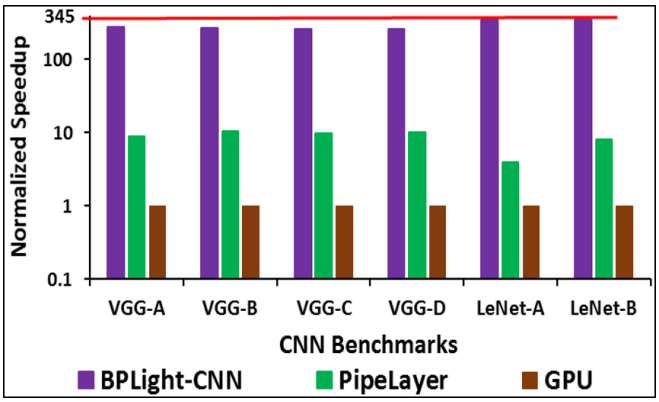
Figure 5.11: *BPLight-CNN* average prediction accuracy comparison with PipeLayer [10] and GPU-based execution across different weight resolutions varying from 2-bit to 32-bit.

There are other minor factors which affect the prediction accuracy of *BPLight-CNN*: (1) Each memristor can have 1000 quantized states. The quantization error encountered due to limited number of memristor states contributes up to 1.2% of Prediction Error (PER); (2) The signal-to-noise ratio of SOA used in *BPLight-CNN* is 50 dB, which is adapted from [108]. The SOAs contribution to the overall PER is 2.35%; (3) Each optical comparator in *BPLight-CNN* has an SNR of 40 dB [112]. This accounts for a PER of 1%; and (4) the memristor-photonic interface is noisy. The signals from memristors going to modulators encounter a noise with an SNR of 25 dB which leads to a PER of 1.45%. We obtained these numbers through detailed optoelectronic synthesis using the IPKISS tool.

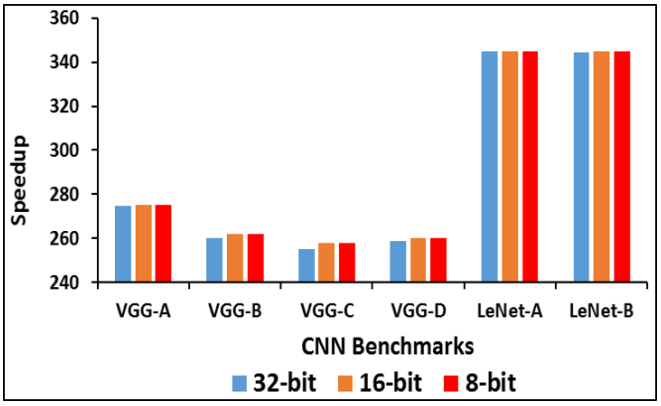
We conducted another sensitivity analysis to explore the impact of weight resolution on average prediction accuracy. Fig.5.11 compares the accuracy of the proposed *BPLight-CNN* with PipeLayer [10] and GPU-based execution across different weight resolutions from 2-bit to 32-bit. From this plot it can be seen that the accuracy of *BPLight-CNN* increases with increase in weight resolution, due to the resulting reduction in quantization error across *BPLight-CNN*. Interestingly, *BPLight-CNN* achieves a prediction accuracy of 95% (i.e., equal to state-of-the-art GPU and PipeLayer design) when its weight resolution is 32-bit. Therefore, we use a 32-bit weight resolution in our performance and energy analysis.

5.6.3 Performance Analysis

Fig.5.12(a) demonstrates normalized speedup (throughput) of *BPLight-CNN* and PipeLayer [10] compared to the baseline GPU implementation results, also from [10], for four variations of the VGG and two variants of the LeNet benchmarks. The GPU-based accelerator performs with an average throughput of 310 GOPS/s. PipeLayer shows an average throughput of 87000 GOPS/s. The proposed *BPLight-CNN* shows an average throughput of 2784000 GOPS/s. The superior performance of *BPLight-CNN* is due to the intelligent integration of ultra-fast memristors and high speed photonic components such as MRAs, SOAs, and comparators. The overall throughput of PipeLayer is affected by inter-layer data conversion with relatively slow ADCs. Also, PipeLayer spends most of its time in sequential weight updates during training. However, *BPLight-CNN*



(a)



(b)

Figure 5.12: (a) Normalized speedup (throughput) comparison across accelerators, (b) Speedup of *BPLight-CNN* w.r.t. weight resolution.

has an inherent advantage due to its photonic parallel weight update mechanism. On average, *BPLight-CNN* outperforms PipeLayer and GPU by 35× and 345× in terms of speedup, respectively. Fig.5.12(b) illustrates the effects of weight resolution on overall speedup of *BPLight-CNN*. With the rise in weight resolution, there is a very little degradation in speedup (5% lower for 32-bit compared to 16-bit). This is due to the additional delay in storing 32-bit data in SRAM compared to 16 or 8-bit data. However, data conversion is done either at the beginning or at the end of the forward pass in *BPLight-CNN*. Therefore, the effect is very minimal.

Fig.5.13 illustrates the normalized computational efficiency (CE) (i.e., the total number of fixed point operations performed per unit area in one second (GOPS/s/mm2)) comparison of the proposed *BPLight-CNN* and memristor crossbar based PipeLayer [10] with respect to a baseline

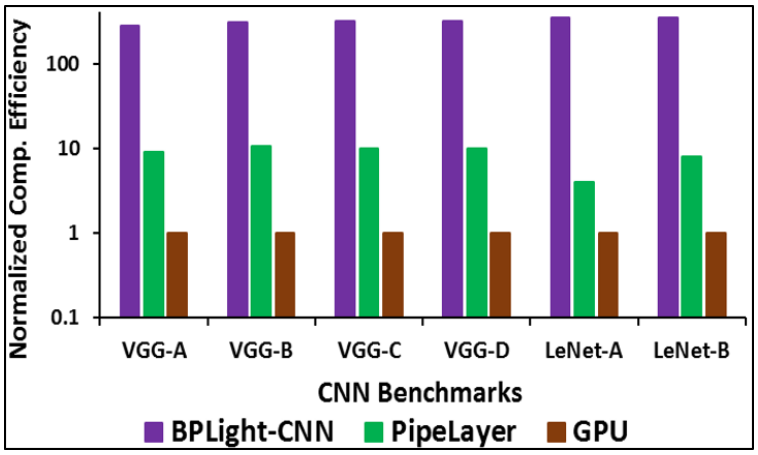
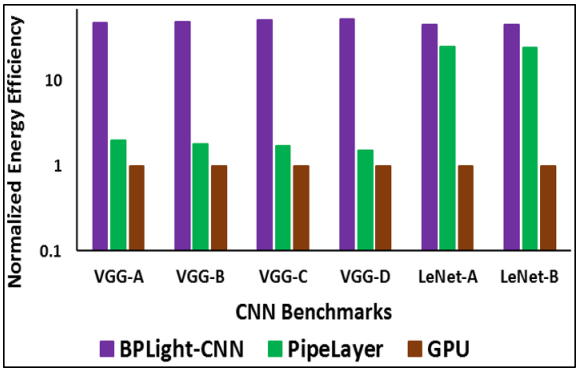


Figure 5.13: Normalized computational efficiency of *BPLight-CNN* compared to state-of-the-art.

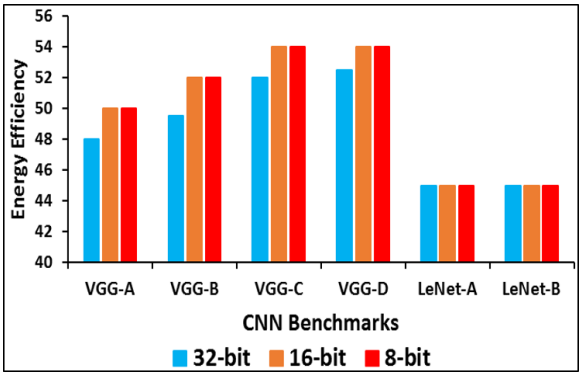
GPU based design. PipeLayer uses memristor crossbars for the bulk of its arithmetic operations. Each memristor crossbar has a CE of 1707 GOPS. However, the overall CE of PipeLayer comes down to 1485 GOPS due to its extensive usage of data conversions. Also, ReLU and POOL are performed by a digital ALU in PipeLayer. This requires more memory to store intra-layer data for synchronizing with its pipeline mechanism. The superiority of *BPLight-CNN* comes from the fact that it is a completely analog accelerator. Therefore, *BPLight-CNN* does not involve inter-layer data conversions or storage for synchronization. AD and DA conversions are done either at the beginning or at the end of feature extraction in *BPLight-CNN*. In addition to the compute efficient memristor, *BPLight-CNN* also uses high speed SOA as ReLU which has a CE in the order of 50000 GOPS/s/mm2 [113]. As shown in Fig.5.13, *BPLight-CNN* has $31\times$ and $320\times$ higher computational efficiency compared to PipeLayer and GPU, respectively. Weight resolution has a negligible effect on computational efficiency of *BPLight-CNN*, therefore, we do not present that result.

5.6.4 Energy Savings

We compare the energy efficiency of *BPLight-CNN* with PipeLayer and GPU as shown in Fig.5.14(a). The average energy efficiency for PipeLayer is 142.9 GOPS/W/s which is $7.17\times$ higher than GPU based accelerator. *BPLight-CNN* works with an average energy efficiency of



(a)



(b)

Figure 5.14: (a) Normalized energy efficiency across accelerators, (b) Energy efficiency of *BPLight-CNN* w.r.t. weight resolution

6432 GOPS/W/s. PipeLayer replicates its early feature extraction layers several times (close to 50K times) to maintain a balanced pipeline. This involves excessive use of high power consuming data conversions. *BPLight-CNN* uses passive optical components such as waveguides and comparators, in addition to energy efficient components such as ring modulators/demodulators, SOAs, and memristor. Also, *BPLight-CNN* uses very few ADCs/DACs compared to PipeLayer. As shown in Fig.5.14(a), we obtain $45\times$ and $360\times$ improvements in energy efficiency for *BPLight-CNN* compared to PipeLayer and GPU, respectively. Fig.5.14(b) shows the effects of weight resolution on overall energy efficiency for all the benchmarks in *BPLight-CNN*. Due to its analog operation, *BPLight-CNN* encounters very minimal effect (0.012%) of weight resolution on energy efficiency.

In summary, from the results presented in this section, it is apparent that our novel *BPLight-CNN* accelerator outperforms previously proposed CNN accelerators by combining photonics-

based backpropagation accelerator with a configurable memristor-integrated photonic CNN accelerator design. The excellent performance and energy gains compared to previous approaches strongly motivate the use of *BPLight-CNN* to execute future CNN based workloads.

5.7 Chapter Summary

This work demonstrates a fully analog CNN accelerator called *BPLight-CNN* that integrates compute-efficient memristors and ultra-fast photonic components. We introduce a reconfigurable convolution design in each CNN layer to enable *BPLight-CNN* to emulate a range of sample CNN models. We also use a novel approach to handle analog signed-weight arithmetic in the memristive convolution layers. Compared to PipeLayer [10] and GPU implementations, the proposed *BPLight-CNN* architecture shows higher computational and energy efficiency due to the use of energy efficient SOAs, optical comparators, and also due to its use of a fully analog feature extraction method. We demonstrated that the proposed design has the potential to achieve up to $35\times$ acceleration in training in addition to $31\times$ improvement in computational efficiency and $45\times$ energy saving compared to the state-of-the-art PipeLayer accelerator with similar accuracy. Photonic components have insertion losses which may slightly affect the overall accuracy when the number of deep learning stages increases. Our future work will address that issue for a highly accurate photonic-based deep learning accelerator.

6. CONCLUSIONS & FUTURE DIRECTIONS

6.1 Conclusions

This dissertation focuses the efficient use of silicon photonics interconnects and devices to design high-performance computing architectures. It begins with the design of a novel ultrafast on-chip photonic router. Such a router is used to design efficient 2D and 3D photonic on-chip network architecture. Further, the dissertation studies the impact of thermal variations in photonic architectures to take necessary measures. In addition to high-throguhput photonic on-chip network architectures, the dissertation also investigates in designing deep learning architectures using photonic-based neuromorphic computing.

In Chapter 2 we demonstrate a novel adaptive multiplexing scheme based energy-efficient photonic network-on-chip. In this work, we propose a non-blocking, low power, low-cost, and high performance 5×5 photonic router design using silicon microring resonators(MRR). In this router, we introduce wavelength-division-multiplexing (WDM) compatible mode-division-multiplexing (MDM) scheme for maximizing the aggregate bandwidth. The proposed photonic router is utilized to design low-cost and energy-efficient 2D and 3D photonic network-on-chip (PNoC). Laser is found to be the most power hungry element in a photonic system. The proposed PNoCs adopt a novel laser-multiplexing scheme to enhance their energy-efficiency.

In Chapter 3, We present the IHDTM framework that exploits device-level on-chip thermal islands and system-level dynamic thread migration scheme TATM for the reduction of maximum on-chip temperature and also conserves trimming and tuning power of MRRs in DWDM-based PNoC architectures. The proactive thermal management scheme used in IHDTM results in interesting trade-offs between performance and power/energy across two different state-of-the-art crossbar-based PNoC architectures. Our experimental analysis on the well-known Corona and Flexishare PNoC architectures has shown that IHDTM can notably conserve

total power by up to 64.1% and thermal tuning power by up to 70%.

In Chapter 4, We demonstrate 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×, packet latency by up to 88%, and energy-per-bit by up to 98% 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.

In Chapter 5, we introduce a fully analog CNN accelerator called *BPLight-CNN* that integrates compute-efficient memristors and ultrafast photonic components. We introduce a reconfigurable convolution design in each CNN layer to enable *BPLight-CNN* to emulate a range of sample CNN models. We also use a novel approach to handle analog signed-weight arithmetic in the memristive convolution layers. Compared to PipeLayer [4] and GPU implementations, the proposed *BPLight-CNN* architecture shows higher computational and energy efficiency due to the use of energy efficient SOAs, optical comparators, and also due to its use of a fully analog feature extraction method. We demonstrated that the proposed design has the potential to achieve up to 35× acceleration in training in addition to 31× improvement in computational efficiency and 45× energy saving compared to the state-of-the-art PipeLayer accelerator with similar accuracy Photonic components have insertion losses which may slightly affect the overall accuracy when the number of deep learning stages increases. Our future work will address that issue for a highly accurate photonic-based deep

learning accelerator.

6.2 Future Directions

Exascale distributed training in the future would require massively parallel and ultrafast architecture. Photonic deep learning is an ideal candidate for such a system design. However, integrating multiple photonic deep learning chip together would require further investigations both in terms of signal integrity and scalability.

Photonic components bring reliability issues, e.g. thermal and process variations based faults. This calls for a reliability aware design for future photonic computing systems.

In the future, the volume of data that computing systems process would grow astronomically. Exascale deep neural networks will become the state-of-the-art for a broad range of terascale data applications such as speech processing, image recognition, financial predictions, etc. Convolutional neural networks (CNNs) are a popular deep learning framework with superior accuracy on applications that deal with videos and images. However, CNNs are highly compute and memory intensive, requiring enormous computational resources. With Moores law coming to an end, traditional Von Neuman computing systems such as heterogeneous CPU/GPU platforms cannot address this high computational demand, within reasonable power and processing time limitations. In addition to that, FPGA and GPU also bring energy pitfalls to tackle these massive tasks. A fully photonic neuromorphic computing system may be an ideal candidate due to its exascale parallelism, ultra-low power nature, and lightspeed characteristics. To establish a photonic neuromorphic system at a commercial level would require investigatios from device to system. This also calls for specific CAD tools design. An efficient CAD framework in this area would accelerate research in this domain.

REFERENCES

[1] L.-W. Luo, N. Ophir, C. P. Chen, L. H. Gabrielli, C. B. Poitras, K. Bergmen, and M. Lipson, "Wdm-compatible mode-division multiplexing on a silicon chip," *Nature Communications*, vol. 5, pp. 3069 EP –, Jan 2014. Article.

[2] D. Dang, B. Patra, R. Mahapatra, and M. Fiers, "Mode-division-multiplexed photonic router for high performance network-on-chip," in *2015 28th International Conference on VLSI Design*, pp. 111–116, Jan 2015.

[3] D. Dang, B. Patra, and R. Mahapatra, "A 2-layer laser multiplexed photonic network-on-chip," in *Sixteenth International Symposium on Quality Electronic Design*, pp. 397–401, March 2015.

[4] D. Dang, S. V. R. Chittamuru, R. Mahapatra, and S. Pasricha, "Islands of heaters: A novel thermal management framework for photonic nocs," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 306–311, Jan 2017.

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72–81, Oct 2008.

[6] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," in *Proceedings 22nd Annual International Symposium on Computer Architecture*, pp. 24–36, June 1995.

[7] D. Dang, R. Mahapatra, and E. J. Kim, "Pid controlled thermal management in photonic network-on-chip," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pp. 17–23, Oct 2015.

[8] S. V. R. Chittamuru, D. Dang, S. Pasricha, and R. Mahapatra, "Bignoc: Accelerating big data computing with application-specific photonic network-on-chip architectures," *IEEE*

Transactions on Parallel and Distributed Systems, vol. 29, pp. 2402–2415, Nov 2018.

[9] D. Dang, J. Dass, and R. Mahapatra, "Convlight: A convolutional accelerator with memristor integrated photonic computing," in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, pp. 114–123, Dec 2017.

[10] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 541–552, Feb 2017.

[11] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 98–109, Aug. 2011.

[12] "Text mining datasets." `http://www.cs.umb.edu/~smimarog/textmining/datasets/`. Accessed: 2017-07-01.

[13] "Grey sorting." `http://sortbenchmark.org/`. Accessed: 2017-07-03.

[14] L. H. K. Duong, M. Nikdast, S. L. 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 Design Test*, vol. 31, pp. 55–65, Oct 2014.

[15] 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, pp. 5172–5186, Mar 2011.

[16] P. Grani and S. Bartolini, "Design options for optical ring interconnect in future client devices," *J. Emerg. Technol. Comput. Syst.*, vol. 10, pp. 30:1–30:25, June 2014.

[17] "The international technology roadmap for semiconductors (itrs) 2011." `http://www.itrs.net/Interconnections`.

[18] C.-H. Liu, Y.-C. Chang, T. B. Norris, and Z. Zhong, "Graphene photodetectors with ultra-broadband and high responsivity at room temperature," *Nature Nanotechnology*, vol. 9, pp. 273 EP –, Mar 2014.

[19] G. Reed and A. Knights, "Silicon photonics: An introduction, wiley," pp. 97–103, 2004.

[20] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of cmos," in *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.*, pp. 7 pp.–15, Dec 2005.

[21] T. Mudge, "Power: a first-class architectural design constraint," *Computer*, vol. 34, pp. 52–58, April 2001.

[22] A. Shacham, K. Bergman, and L. P. Carloni, "*Photonic networks-on-chip for future generations of chip multiprocessors,*," p. 12461260, 2008.

[23] H. Gu, K. H. Mo, J. Xu, and W. Zhang, "A low-power low-cost optical router for optical networks-on-chip in multiprocessor systems-on-chip," in *2009 IEEE Computer Society Annual Symposium on VLSI*, pp. 19–24, May 2009.

[24] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*. IEEE Computer Society Annual Symposium on VlSI, 2002.

[25] A. Shacham, K. Bergman, and L. P. Carloni, "On the design of a photonic network-on-chip," in *First International Symposium on Networks-on-Chip (NOCS'07)*, pp. 53–64, May 2007.

[26] C. Batten, A. Joshi, J. Orcutt, C. Holzwarth, M. Popovic, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic, and K. Asanovic, "Building manycore processor-to-dram networks with monolithic cmos silicon photonics," *IEEE Micro*, pp. 1–1, 2018.

[27] A. Joshi, C. Batten, Y. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic, "Silicon-photonic clos networks for global on-chip communication," in *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pp. 124–133, May 2009.

[28] A. Kamierczak, E. Drouard, M. Ere, P. Rojo-Romeo, X. Letartre, I. OConnor, F. Gaffiot, and Z. Lisik, "*Optimization of an integrated optical crossbar in SOI technology for optical networks on chip*, journal of telecommunications & information technology, vol. 2007 issue 3, pp 109-114," pp. 109–114, 2007.

[29] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System implications of emerging nanophotonic technology," in *2008 International Symposium on Computer Architecture*, pp. 153–164, June 2008.

[30] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating future network-on-chip with nanophotonics," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, (New York, NY, USA), pp. 429–440, ACM, 2009.

[31] A. V. Krishnamoorthy, R. Ho, X. Zheng, H. Schwetman, J. Lexau, P. Koka, G. Li, I. Shubin, and J. E. Cunningham, "Computer systems based on silicon photonic interconnects," *Proceedings of the IEEE*, vol. 97, pp. 1337–1361, July 2009.

[32] G. Hendry, E. Robinson, V. Gleyzer, J. Chan, L. P. Carloni, N. Bliss, and K. Bergman, "Time-division-multiplexed arbitration in silicon nanophotonic networks-on-chip for high-performance chip multiprocessors," *Journal of Parallel and Distributed Computing*, vol. 71, no. 5, pp. 641 – 650, 2011. Networks-on-Chip.

[33] R. Ji, L. Yang, L. Zhang, Y. Tian, J. Ding, H. Chen, Y. Lu, P. Zhou, and W. Zhu, "Five-port optical router for photonic networks-on-chip," *Opt. Express*, vol. 19, pp. 20258–20268, Oct 2011.

[34] G. Fan, R. Orobtchouk, and J.-M. Fedeli, "Highly integrated optical 8x8 lambda-router in silicon-on-insulator technology: comparison between the ring and racetrack configuration," *Proceedings of SPIE - The International Society for Optical Engineering*, pp. 10–, 04 2010.

[35] M. Briere, B. Girodias, Y. Bouchebaba, G. Nicolescu, F. Mieyeville, F. Gaffiot, and I. O'Connor, "System level assessment of an optical noc in an mpsoc platform," in *2007 Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, April 2007.

[36] A. W. Poon, F. Xu, and X. Luo, "Cascaded active silicon microresonator array cross-connect circuits for wdm networks-on-chip," 2008.

[37] W. Bogaerts and a. et., "Silicon microring resonators, laser & photonics reviews 6, no. 1, pp 47-73," pp. 47–73, 2012.

[38] D. Dai, "Silicon nanophotonic integrated devices for on-chip multiplexing and switching," *J. Lightwave Technol.*, vol. 35, pp. 572–587, Feb 2017.

[39] A. Alduino, "Demonstration of a high speed 4-channel integrated silicon photonics wdm link with hybrid silicon lasers," in *2010 IEEE Hot Chips 22 Symposium (HCS)*, pp. 1–29, Aug 2010.

[40] S. Bagheri and W. M. J. Green, "Silicon-on-insulator mode-selective add-drop unit for on-chip mode-division multiplexing," in *2009 6th IEEE International Conference on Group IV Photonics*, pp. 166–168, Sep. 2009.

[41] Y. Huang, G. Xu, and S. Ho, "An ultracompact optical mode order converter," *IEEE Photonics Technology Letters*, vol. 18, pp. 2281–2283, Nov 2006.

[42] Y. Kawaguchi and K. Tsutsumi, "Mode multiplexing and demultiplexing devices using multimode interference couplers," *Electronics Letters*, vol. 38, pp. 1701–1702, Dec 2002.

[43] J. Leuthold, R. Hess, J. Eckner, P. A. Besse, and H. Melchior, "Spatial mode filters realized with multimode interference couplers," *Opt. Lett.*, vol. 21, pp. 836–838, Jun 1996.

[44] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, pp. 684–689, June 2001.

[45] A. Bianco, D. Cuda, R. Gaudino, G. Gavilanes, F. Neri, and M. Petracca, "Scalability of optical interconnects based on microring resonators," *IEEE Photonics Technology Letters*, vol. 22, pp. 1081–1083, Aug 2010.

[46] Y. Xie, M. Nikdast, J. Xu, W. Zhang, Q. Li, X. Wu, Y. Ye, X. Wang, and W. Liu, "Crosstalk noise and bit error rate analysis for optical network-on-chip," in *Design Automation Conference*, pp. 657–660, June 2010.

[47] D. Dang, B. Patra, and R. Mahapatra, "A multilayered design approach for efficient hybrid 3d photonics network-on-chip," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, (New York, NY, USA), pp. 121–126, ACM, 2015.

[48] "Ipkiss - a generic and modular software framework for parametric design." `https://www.lucedaphotonics.com/en`.

[49] G. Z. Mashanovich, M. Milosevic, P. Matavulj, S. Stankovic, B. Timotijevic, P. Y. Yang, E. J. Teo, M. B. H. Breese, A. A. Bettiol, and G. T. Reed, "Silicon photonic waveguides for different wavelength regions," *Semiconductor Science and Technology*, vol. 23, no. 6, p. 064002, 2008.

[50] "A network simulator." `http://noxim.sourceforge.net/`. Accessed: 2014-07-02.

[51] A. Shacham, B. G. Lee, A. Biberman, K. Bergman, and L. P. Carloni, "Photonic noc for dma communications in chip multiprocessors," in *15th Annual IEEE Symposium on High-Performance Interconnects (HOTI 2007)*, pp. 29–38, Aug 2007.

[52] S. Abadal, A. Cabellos-Aparicio, J. A. Lázaro, M. Nemirovsky, E. Alarcón, and J. Solé-Pareta, "Area and laser power scalability analysis in photonic networks-on-chip," in *2013 17th International Conference on Optical Networking Design and Modeling (ONDM)*, pp. 131–136, April 2013.

[53] G. Kurian, J. Miller, J. Psota, J. Eastep, J. Liu, J. Michel, L. Kimerling, and A. Agarwal, "*ATAC: A 1000-Core Cache-Coherent Processor with On-Chip Optical Network*, 19th inter-

national conference on parallel architectures and compilation techniques, acm, pp. 477488," p. 477488, 2010.

[54] C. Nitta, M. Farrens, and V. Akella, "*DCOFAn Arbitration Free Directly Connected Optical Fabric*, ieee journal on emerging and selected topics in circuits and systems, vol.2, no.2, pp.169,182," pp. 169–182, 2012.

[55] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "*Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures*, ieee transaction on computers, vol. 54, no. 8," 2005.

[56] S. bahirat & Sudeep Pasricha, "*METEOR: Hybrid photonic ring-mesh network-on-chip for multicore architectures*,acm transactions on embedded computing systems (tecs) - special issue on design challenges for many-core processors, vol 13 issue 3s, march 2014 article no. 116," 2014.

[57] S. Xiao, M. Khan, H. Shen, and Q. M., "*Multiple channel silicon micro-resonator based filters for WDM application*,optics express vol 15 pp. 7489-7498," pp. 7489–7498, 2007.

[58] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 463–470, Nov 2010.

[59] J. Ahn *et al.*, "Devices and architectures for photonic chip-scale integration, in Applied Physics A: MSP, 95:989-997," June 2009.

[60] C. Nitta, M. Farrens, and V. Akella, "Addressing system-level trimming issues in on-chip nanophotonic networks," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pp. 122–131, Feb 2011.

[61] S. S. Djordjevic, K. Shang, B. Guan, S. T. S. Cheung, L. Liao, J. Basak, H.-F. Liu, and S. J. B. Yoo, "Cmos-compatible, athermal silicon ring modulators clad with titanium dioxide," *Opt. Express*, vol. 21, pp. 13958–13968, Jun 2013.

[62] S. V. R. Chittamuru and S. Pasricha, "Spectra: A framework for thermal reliability management in silicon-photonic networks-on-chip," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pp. 86–91, Jan 2016.

[63] T. Zhang, J. L. Abellán, A. Joshi, and A. K. Coskun, "Thermal management of manycore systems with silicon-photonic networks," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[64] "Pid lab." `http://www.pidlab.com/en/`.

[65] S. V. R. Chittamuru, I. G. Thakkar, and S. Pasricha, "Pico: Mitigating heterodyne crosstalk due to process variations and intermodulation effects in photonic nocs," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2016.

[66] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), pp. 144–152, ACM, 1992.

[67] T. E. Carlson, W. Heirmant, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, Nov 2011.

[68] D. Vantrease, N. Binkert, R. Schreiber, and M. H. Lipasti, "Light speed arbitration and flow control for nanophotonic interconnects," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 304–315, Dec 2009.

[69] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pp. 1–12, Jan 2010.

[70] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore archi-

tectures," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 469–480, Dec 2009.

[71] S. V. R. Chittamuru, S. Desai, and S. Pasricha, "Reconfigurable silicon-photonic network with improved channel sharing for multicore architectures," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, (New York, NY, USA), pp. 63–68, ACM, 2015.

[72] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *2008 45th ACM/IEEE Design Automation Conference*, pp. 734–739, June 2008.

[73] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[74] "Hadoop." `https://hadoop.apache.org`. Accessed: 2017-07-15.

[75] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2010.

[76] J. Hamilton, "Cooperative expendable micro-slice servers (cems): Low cost, low power servers for internet-scale services," 2009.

[77] Y. Xia, T. S. E. Ng, and X. S. Sun, "Blast: Accelerating high-performance data analytics applications by optical multicast," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1930–1938, April 2015.

[78] "Refining knn." `https://www3.nd.edu/~steve/computing_with_data/17_Refining_kNN/refining_knn.html`. Accessed: 2017-08-05.

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

[80] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System implications of emerging nanophotonic technology," in *2008 International Symposium on Computer Architecture*, pp. 153–164, June 2008.

[81] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating future network-on-chip with nanophotonics," *SIGARCH Comput. Archit. News*, vol. 37, pp. 429–440, June 2009.

[82] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pp. 1–12, Jan 2010.

[83] R. W. M. Jr. and A. K. Kodi, "Power-efficient and high-performance multi-level hybrid nanophotonic interconnect for multicores," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pp. 207–214, May 2010.

[84] E. Fusella, J. Flich, and A. Cilardo, "Path setup for hybrid noc architectures exploiting flooding and standby," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 1403–1416, May 2017.

[85] C. Li, M. Browning, P. V. Gratz, and S. Palermo, "Luminoc: A power-efficient, high-performance, photonic network-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 826–838, June 2014.

[86] E. Kakoulli, V. Soteriou, C. Koutsides, and K. Kalli, "Design of high-performance, power-efficient optical nocs using silica-embedded silicon nanophotonics," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pp. 1–8, Oct 2015.

[87] A. Kulkarni, T. Abtahi, E. Smith, and T. Mohsenin, "Low energy sketching engines on many-core platform for big data acceleration," in *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '16, (New York, NY, USA), pp. 57–62, ACM, 2016.

[88] K. Kanoun, M. Ruggiero, D. Atienza, and M. v. d. Schaar, "Low power and scalable many-core architecture for big-data stream computing," in *2014 IEEE Computer Society Annual Symposium on VLSI*, pp. 468–473, July 2014.

[89] 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 Journal of Solid-State Circuits*, vol. 48, pp. 1943–1953, Aug 2013.

[90] 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 Transactions on Parallel and Distributed Systems*, vol. 24, pp. 2451–2461, Dec 2013.

[91] D. Vainbrand and R. Ginosar, "Network-on-chip architectures for neural networks," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pp. 135–144, May 2010.

[92] A. Firuzan, M. Modarressi, and M. Daneshtalab, "Reconfigurable communication fabric for efficient implementation of neural networks," in *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp. 1–8, June 2015.

[93] 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, pp. 1525–1538, Dec. 2013.

[94] 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 *2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*, pp. 1–10, Oct 2016.

[95] Z. Wang, S. Zhang, B. He, and W. Zhang, "Melia: A mapreduce framework on opencl-based fpgas," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 3547–3560, Dec

2016.

[96] N. E. Jerger, L. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *2008 International Symposium on Computer Architecture*, pp. 229–240, June 2008.

[97] I. G. Thakkar, S. V. R. Chittamuru, and S. Pasricha, "Run-time laser power management in photonic nocs with on-chip semiconductor optical amplifiers," in *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–4, Aug 2016.

[98] C. Chen and A. Joshi, "Runtime management of laser power in silicon-photonic multi-bus noc architecture," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, pp. 3700713–3700713, March 2013.

[99] R. S. Tsay, *Analysis of Financial Time Series*. John Wiley & Sons, Inc., ISBN 0-471-41544-8, 2002.

[100] "Airline query processing." `http://www.stat.purdue.edu/~sguha/rhipe/doc/html/airline.html`. Accessed: 2017-07-05.

[101] T. Krishna, L.-S. Peh, B. M. Beckmann, and S. K. Reinhardt, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, (New York, NY, USA), pp. 71–82, ACM, 2011.

[102] "Amazon elastic cloud computer." `http://aws.amazon.com/ec2`. Accessed: 2017-07-06.

[103] 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 *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pp. 201–210, May 2012.

[104] C. Zhang, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, Nov 2016.

[105] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 14–26, June 2016.

[106] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," *Frontiers in Neuroscience*, vol. 10, p. 333, 2016.

[107] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2016.

[108] K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, and P. Bienstman, "Parallel reservoir computing using optical amplifiers," *IEEE Transactions on Neural Networks*, vol. 22, pp. 1469–1481, Sep. 2011.

[109] Y. Shen, N. C. Harris, D. Englund, and M. SoljaCiC, "Deep learning with coherent nanophotonic circuits," in *2017 Fifth Berkeley Symposium on Energy Efficient Electronic Systems Steep Transistors Workshop (E3S)*, pp. 1–2, Oct 2017.

[110] D. Dang, J. Dass, and R. Mahapatra, "Convlight: A convolutional accelerator with memristor integrated photonic computing," in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, pp. 114–123, Dec 2017.

[111] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[112] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[113] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[114] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

[115] Y. Long, L. Zhou, and J. Wang, "Photonic-assisted microwave signal multiplication and modulation using a silicon mach-zehnder modulator," *Scientific Reports*, vol. 6, pp. 20215 EP –, Feb 2016. Article.

[116] M. J. Connelly, "Reflective semiconductor optical amplifier pulse propagation model," *IEEE Photonics Technology Letters*, vol. 24, pp. 95–97, Jan 2012.

[117] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80 EP –, May 2008.

[118] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, Dec 2012. Review Article.

[119] P. Li, X. Yi, X. Liu, D. Zhao, Y. Zhao, and Y. Wang, "All-optical analog comparator," *Scientific Reports*, vol. 6, pp. 31903 EP –, Aug 2016. Article.

[120] T. Fujita, Y. Toba, Y. Miyoshi, and M. Ohashi, "Optical analog multiplier based on phase sensitive amplification," in *2013 18th OptoElectronics and Communications Conference held jointly with 2013 International Conference on Photonics in Switching (OECC/PS)*, pp. 1–2, June 2013.

[121] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, pp. 211–252, Dec. 2015.

[122] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, (New York, NY, USA), pp. 675–678, ACM, 2014.

[123] "MNIST Database by yann lecun." `http://yann.lecun.com/exdb/mnist/`. Accessed: 2018-08-01.

[124] J. M. Ramirez, Q. Liu, V. Vakarin, J. Frigerio, A. Ballabio, X. L. Roux, D. Bouville, L. Vivien, G. Isella, and D. Marris-Morini, "Graded sige waveguides with broadband low-loss propagation in the mid infrared," *Opt. Express*, vol. 26, pp. 870–877, Jan 2018.