DISSERTATION

HARDWARE-SOFTWARE CODESIGN OF SILICON PHOTONIC AI ACCELERATORS

Submitted by

Febin P. Sunny

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2024

Doctoral Committee:

Advisor: Sudeep Pasricha
Co-Advisor: Mahdi Nikdast

Haonan Chen
Yashwant K. Malaiya

ABSTRACT

HARDWARE-SOFTWARE CO-DESIGN OF SILICON PHOTONIC AI ACCELERATORS

Machine learning applications have become increasingly prevalent over the past decade across many real-world use cases, from smart consumer electronics to automotive, healthcare, cybersecurity, and language processing. This prevalence has been fueled by the emergence of powerful machine learning models, such as Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). As researchers explore deeper models with higher connectivity, the computing power and the memory requirement necessary to train and utilize them also increase. Such increasing complexity also necessitates that the underlying hardware platform should consistently deliver better performance while satisfying strict power constraints. Unfortunately, the limited performance-per-watt in today's computing platforms — such as CPUs, GPUs, and electronic neural network (NN) accelerators — creates significant challenges for the growth of new deep learning and AI applications. These electronic computing platforms face fundamental limits in the post-Moore's Law era due to increased ohmic losses and capacitance-induced latencies in interconnects, as well as power inefficiencies and reliability concerns that reduce yields and increase costs with semiconductor-technology scaling.

A solution to improving performance-per-watt for AI model processing is to explore more efficient hardware NN accelerator platforms. Silicon photonics has shown promise in terms of achievable energy efficiency and latency for data transfers. It is also possible to use photonic components to perform computation, e.g., matrix-vector multiplication. Such photonic-based AI accelerators can not only address the fan-in and fan-out problem with linear algebra processors, but their operational bandwidth can approach the photodetection rate (typically in the hundreds of

GHz), which is orders of magnitude higher than electronic systems today that operate at a clock rate of a few GHz. A solution to the data-movement bottleneck can be the use of silicon photonics technology to realize photonic networks-on-chip (PNoCs), which can enable ultra-high bandwidth, low latency, and energy-efficient communication.

However, to ensure reliable, efficient, and high throughput communication and computation using photonics, several challenges must be addressed first. Photonic computation is performed in the analog domain, which makes it susceptible to various noise sources and drives down the achievable resolution for representing NN model parameters. To increase the reliability of silicon photonic AI accelerators, fabrication-process variation (FPV), which is the change in physical dimensions and characteristics of devices due to imperfections in fabrication, must be addressed. For our proposed accelerator architectures to operate correctly, the fundamental devices involved (microring resonators (MRs)) have to be corrected for FPV induced resonant wavelength shifts. Without this correction, FPVs will cause increased crosstalk and data corruption during photonic communication and can also lead to errors during photonic computation. Accordingly, compensation for the impact of FPVs is an essential part of reliable computation in silicon photonic-based AI accelerators. Even with FPV-tolerant silicon photonic devices, the tuning latency incurred by thermo-optic (TO) tuning and the thermal crosstalk it can induce are significant. The latency, which can be in the microsecond range, impacts the overall throughput of the accelerator and the thermal crosstalk impacts its reliable operation. At the architectural level it is also necessary to ensure that the NN processing is done efficiently while making use of the photonic resources in terms of wavelengths, and NN model-aware decisions in terms of device deployment, arrangement, and multiply and accumulate (MAC) unit design have to be performed.

To address these challenges, the major contributions of this thesis are focused on proposing a hardware-software co-design framework to enable high throughput, low latency, and energy-efficient AI acceleration across various neural network models, using silicon photonics. At the architectural level, we have proposed wavelength reuse schemes, vector decomposition, and NN-aware MAC unit designs for increased efficiency in laser power consumption. In terms of NN-aware designs, we have proposed layer-specific acceleration units, photonic batch normalization folding, and fine-grained sparse NN acceleration units. To tackle the reliability challenges introduced by FPV, we have performed device-level design-space exploration and optimization to design MRs that are more tolerant to FPVs than the state-of-the-art efforts in this area. We also adapt Thermal Eigen-mode decomposition and have devised various novel techniques to manage thermal and spectral crosstalk sources, allowing our silicon photonic-based AI accelerators to reach up to 16-bit parameter resolution per MR, which enables high accuracy for most NN models.

ACKNOWLEDGEMENTS

Completing this thesis has been a profoundly challenging journey, yet immensely rewarding. This work represents not just the culmination of over five years dedicated to the evolving field of machine learning inference acceleration and silicon nanophotonics but a personal voyage into domains of knowledge I had never anticipated exploring when I began.

I thank my advisor and mentor during this journey, Dr. Sudeep Pasricha for his incredible mentorship and guidance. His patience, which I feel I stretched substantially, and encouragement had helped me tremendously through this process by preparing me for research and helping me find my area of interest, machine learning acceleration using silicon photonics. His advice and guidance helped me develop a critical and creative approach towards research problems, which I believe will help me in the long run.

I appreciate the support of my co-advisor Dr. Mahdi Nikdast, who has been very helpful in guiding my work through the ever evolving field of silicon photonics. His feedback and guidance have been invaluable through my PhD research.

I really appreciate all the help, guidance, and inspiration I received from both my advisors.

I would also like to thank the respected members of my PhD Committee: Dr. Haonan Chen and Dr. Yashwant K. Malaiya. Their feedback helped me to rediscover my research and refine my work from different perspectives. I am grateful to Asif Mirza and Amin Shafiee for their invaluable collaboration and valuable insights on nanophotonic devices. I must also mention Dr. Ishan Thakkar, Dr. Venkata Sai Praneeth Karempudi, Ebad Taheri, and Salma Afifi for their collaboration in various research publications. Additionally, I must also thank Poorna Kale and

Dr. Chris Neely who mentored me during my internships at Micron and AMD respectively for their valuable advice and technological insights.

I am thankful for all the friends who helped me in one way or other during my PhD journey: Sonu Dileep, Abhishek Balasubramanium, Pavana Prakash, Saideep Tiku, Sirui Qi, Vipin Kumar Kukkala, Yaswant Raparti, Danish Gufran, Sanmitra Banerjee, and many others for their camaraderie and encouragement.

Finally and importantly, I thank my family, my father, mother, and two sisters, whose unwavering support was my cornerstone while pursuing my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RESEARCH PUBLICATIONS

- A. Mirza, **F. Sunny**, S. Pasricha, and M. Nikdast, "Silicon photonic microring resonators: Design optimization under fabrication non-uniformity," Design, Automation & Test in Europe (DATE), 2020.
- **F. Sunny**, A. Mirza, S. Pasricha, and M. Nikdast, "LORAX: Loss-aware approximations for energy-efficienct silicon photonic networks-on-chip," ACM Great Lakes Symposium on VLSI, 2020. (Best Paper Candidate)
- **F. Sunny,** A. Mirza, I. Thakkar, M. Nikdast, and S. Pasricha, "ARXON: A Framework for Approximate Communication over Photonic Networks-on-Chip," IEEE Transactions on Very Large Integration Systems, vol. 29, no. 6, 2021.
- **F. Sunny,** E. Taheri, M. Nikdast, and S. Pasricha, "A Survey on Silicon Photonics for Deep Learning," ACM Journal of Emerging Technologies in Computing Systems, vol. 17, no. 4, pp. 1-57, 2021.
- **F. Sunny,** A. Mirza, M. Nikdast, and S. Pasricha, "ROBIN: A robust optical binary neural network accelerator," ACM Transactions on Embedded Computing Systems, vol. 20, no. 5s, pp. 1-24, 2021.
- A. Shafiee, A. Mirza, **F. Sunny**, S. Banerjee, K. Chakrabarty, S. Pasricha, and M. Nikdast, "Inexact Silicon Photonics: From Devices to Applications," Photonics in Switching and Computing, 2021.
- A. Mirza, **F. Sunny**, P. Walsh, K. Hassan, S. Pasricha, and M. Nikdast, "Silicon photonic microring resonators: A comprehensive design-space exploration and optimization under fabrication-process variations," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 10, 2021.

- **F. Sunny**, A. Mirza, M. Nikdast, and S. Pasricha, "CrossLight: A cross-layer optimized silicon photonic neural network accelerator," ACM/IEEE Design Automation Conference, 2021.
- **F. Sunny,** M. Nikdast, and S. Pasricha, "SONIC: A sparse neural network inference accelerator with silicon photonics for energy-efficient deep learning," Asia and South Pacific Design Automation Conference, 2022.
- V. S. P. Karempudi, **F. Sunny**, I. Thakkar, S. V. R. Chittamuru, M. Nikdast, and S. Pasricha, "Photonic networks-on-chip employing multilevel signaling: A cross-layer comparative study," ACM Journal on Emerging Technologies in Computing Systems, vol. 18, no. 3, 2022.
- **F. Sunny**, M. Nikdast, and S. Pasricha, "A silicon photonic accelerator for convolutional neural networks with heterogeneous quantization," Great Lakes Symposium on VLSI, 2022.
- **F. Sunny,** M. Nikdast, and S. Pasricha, "RecLight: A recurrent neural network accelerator with integrated silicon photonics," IEEE Computer Soceity Annual Symposium on VLSI, 2022.
- **F. Sunny**, E. Taheri, M. Nikdast, and S. Pasricha, "Machine learning accelerators in 2.5D chiplet platforms with silicon photonics," Design, Automation, and Test in Europe, 2023.
- S. Afifi, **F. Sunny,** M. Nikdast, and S. Pasricha, "TRON: Transformer neural network acceleration with non-coherent photonics," Great Lakes Symposium on VLSI, 2023.
- **F. Sunny,** M. Nikdast, and S. Pasricha, "Cross-layer design of AI acceleration with non-coherent optical computing," Great Lake Symposium on VLSI, 2023.

- A. Balasubramanium, **F. Sunny**, and S. Pasricha, "R-TOSS: A framework for real-time object detection using semi-structured pruning," ACM/IEEE Design Automation Conference, 2023.
- S. Afifi, **F. Sunny,** M. Nikdast, and S. Pasricha, "GHOST: A Graph Neural Network Accelerator using Silicon Photonics," ACM Transactions on Embedded Computing Systems, 2023.
- **F. Sunny**, A. Shafiee, B. Charbonnier, M. Nikdast, and S. Pasricha, "COMET: A Cross-Layer Optimized Phase Change Main Memory Architecture," [to appear], Design, Automation & Test in Europe, 2024.
- S. Afifi, **F. Sunny**, M. Nikdast, and S. Pasricha, "Accelerating Neural Networks for Large Language Models and Graph Processing with Silicon Photonics," [to appear], Design, Automation & Test in Europe, 2024.
- **F. Sunny**, E. Taheri, M. Nikdast, and S. Pasricha, "Silicon Photonic 2.5D Interposer Networks for Overcoming Communication Bottlenecks in Scale-Out Machine Learning Hardware Accelerators," [to appear], IEEE VLSI Test Symposium, 2024.
- **F. Sunny**, A. Shafiee, A. Balasubramaniam, M. Nikdast, and S. Pasricha, "OPIMA: Optical Processing-In-Memory for Convolutional Neural Network Acceleration," [under review], IEEE Transactions on Computer-Aided Design of Circuits and Systems.

CHAPTER 1: INTRODUCTION

Modern AI acceleration platforms, though capable of achieving significant acceleration throughput, may not be a scalable solution in terms of energy efficiency. In this thesis, we shall discuss how to utilize silicon photonics for energy-efficient, high throughput deep neural network (DNN) acceleration. In this chapter, we shall discuss preliminary concepts about machine learning acceleration and the challenges associated with it. Furthermore, we motivate the use of photonic systems for DNN acceleration, while presenting the various challenges which has to be overcome to realize such a system. Finally, we shall outline the contributions of this thesis towards tackling these challenges.

1.1. MACHINE LEARNING APPLICATIONS

Over the last decade, Machine Learning (ML), which is a sub-field of Artificial Intelligence (AI), has proved to be a paradigm-shifting technology in how we process and access information. Recently, with the advent of complex deep learning ML models such as Transformers along with the long-standing success of various models such as convolution neural networks (CNNs), ML has found unprecedented success across a variety of domains such as computer vision [1], natural language processing [2], robotics [3], and generative artificial intelligence [4]. This is especially impressive considering deep learning was abandoned as a viable form of ML in the 1990s, due to the difficulties in training DNN models [5]. However, the limitations researchers faced in the ‘90s were largely from hardware limitations and lack of training data. With the advent of big data and the graphics processing unit (GPU) architecture from Nvidia, these limitations were mostly met, leading to widespread research into, and subsequently the success of deep learning ML models. Today deep learning is ubiquitous with AI and is considered for advanced AI applications such as

artificial general intelligence (AGI). This has led to extensive research into deep neural networks (DNNs) which has produced computationally complex and memory-hungry models. As the demand for ML algorithms across application spaces escalates, the diversity in DNN architectures increases to address different challenges and requirements. Let's explore some fundamental types of DNNs that illustrate this diversity and their respective strengths in handling various computational tasks.

1.1.1. MULTILAYER PERCEPTRONS

Multilayer Perceptrons (MLPs), also known as fully connected neural networks, are the simplest form of deep neural networks. An MLP consists of at least three layers: an input layer, one or more hidden layers, and an output layer. Each layer is fully connected to the next, meaning every neuron in one layer connects to every neuron in the following layer. MLPs utilize nonlinear activation functions between layers, typically sigmoid, ReLU, or tanh, to capture complex relationships in the input data. They are widely used for simple regression and classification tasks where the data is well-structured and requires a straightforward, generalizable model.

1.1.2. CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) are specialized deep neural networks for processing data that has a grid-like topology, such as images. CNNs are distinguished from other neural networks by their use of convolutional layers that impose a local connectivity pattern between neurons of adjacent layers. The architecture exploits the spatial structure of the data, ensuring that the learned filters produce the strongest response to spatially local input patterns, which results in excellent capabilities for capturing visual patterns directly from pixel images with minimal preprocessing. They typically include layers that perform convolutions, pooling (subsampling or

down-sampling), nonlinear activation functions, and fully connected layers. CNNs have been highly successful in applications such as image classification, image and video recognition, medical image analysis, and natural language processing, where they can be applied to the analysis of both time-series and text data [6].

1.1.3. RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNNs) are a class of neural networks that are vital for processing sequential data, such as time series or linguistic data. Unlike feedforward neural networks, RNNs have connections that form directed cycles, allowing information to persist from one step of the network operation to the next. However, standard RNNs often suffer from vanishing and exploding gradient problems which make training them on long sequences difficult [7]. Gated Recurrent Units (GRUs) and Long Short-Term Memory Networks (LSTMs) are advanced RNNs that address these issues with specialized gating mechanisms. LSTMs include input, output, and forget gates that regulate the flow of information, allowing the network to retain or discard data dynamically, which is crucial for learning long dependencies. GRUs simplify the gating mechanism, using two gates (update and reset) and merging the cell state and hidden state, thus being more computationally efficient while achieving similar performance to LSTMs. These architectures are particularly powerful in applications such as speech recognition, language modeling, and translation.

1.1.4. GRAPH NEURAL NETWORKS

Graph Neural Networks (GNNs) are designed to perform machine learning on graph structures, including social networks, molecular structures, and communication networks. These networks take into account the connections between nodes (entities) and the edges (relationships), allowing

them to capture the dependencies that are not accessible to traditional deep learning algorithms. GNNs apply convolution directly to the graph structure, aggregating information from a node's neighbors to compute its next state [8]. This process, often repeated across several layers, effectively embeds the nodes into a low-dimensional space where machine learning tasks such as classification, prediction, and clustering can be applied. Additionally, GNNs can also make use of attention mechanisms to contextually understand the usefulness of specific edges with respect to a group of nodes [9]. GNNs have been particularly useful in drug discovery, social network analysis, and recommendation systems where the inherent data structure is a graph.

1.1.5. TRANSFORMERS

Transformers are an advanced model architecture that eschews recurrence and instead relies entirely on an attention mechanism to draw global dependencies between input and output [10]. The Transformer allows for much more parallelization than RNNs and has been crucial in achieving breakthroughs in various NLP tasks through models such as BERT, GPT, and T5 [2]. Its self-attention mechanism assigns a weight to each part of the input data depending on the other parts, allowing the model to evaluate the input as a whole. This ability to maintain contextual sensitivity, makes them ideal for tasks that require understanding the entire context of the data, such as translation, text summarization, and advanced content generation.

1.2. ML HARDWARE ACCELERATION AND CHALLENGES

Despite the emerging complexity, the layer-wise operation of DNNs remains fundamentally matrix-matrix or matrix-vector multiplication operations. These operations are excellent candidates for parallel computing and hence can be deployed for accelerated operation on many-core computing architectures, like GPUs. Since these matrix operations can be decomposed into

multiply-and-accumulate (MAC) operations, manycore architectures with a large count of energy-efficient but less computationally capable cores are well suited for DNN acceleration. Conventional GPUs and CPUs evolved to speed up deep learning model execution, e.g., Nvidia GPUs now include tensor cores [11], and CPUs support increasingly advanced vector instructions [12], both of which are designed to accelerate common matrix and vector operations in deep learning processing.

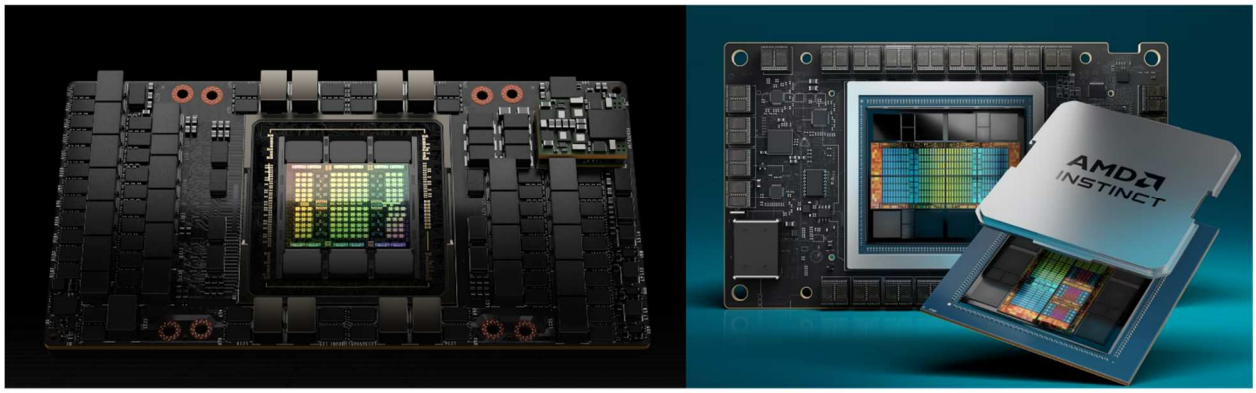


Figure 1. Commercially available ML acceleration platforms for server scale ML acceleration (left) Nvidia H100 [13]; (right) AMD MI300x [14].

However, along with computational challenges, AI acceleration also faces a memory challenge. Conventional memory systems are struggling to keep up with the increasing bandwidth, and latency demands of modern manycore architectures [15]. This causes the architecture to not function at its maximum possible throughput and energy efficiency, as several processing elements may be starved of data per cycle due to limited data delivery rate from main memory or local memories like caches. Also referred to as the memory wall challenge [16], this problem is inherent to all Von Neumann architectures, where computation and memory are separated. This is exacerbated as the DNN architecture parameter count grows, with large language models routinely reaching 100s of billions of parameters in size, and as architecture complexity (number of branches or even loops in DNN architecture) grows [17]. These factors increase both memory capacity

requirement and memory bandwidth requirement respectively. For server scale operation of these larger models, memory operation alone takes up over 30% of the overall power and energy consumption [18].

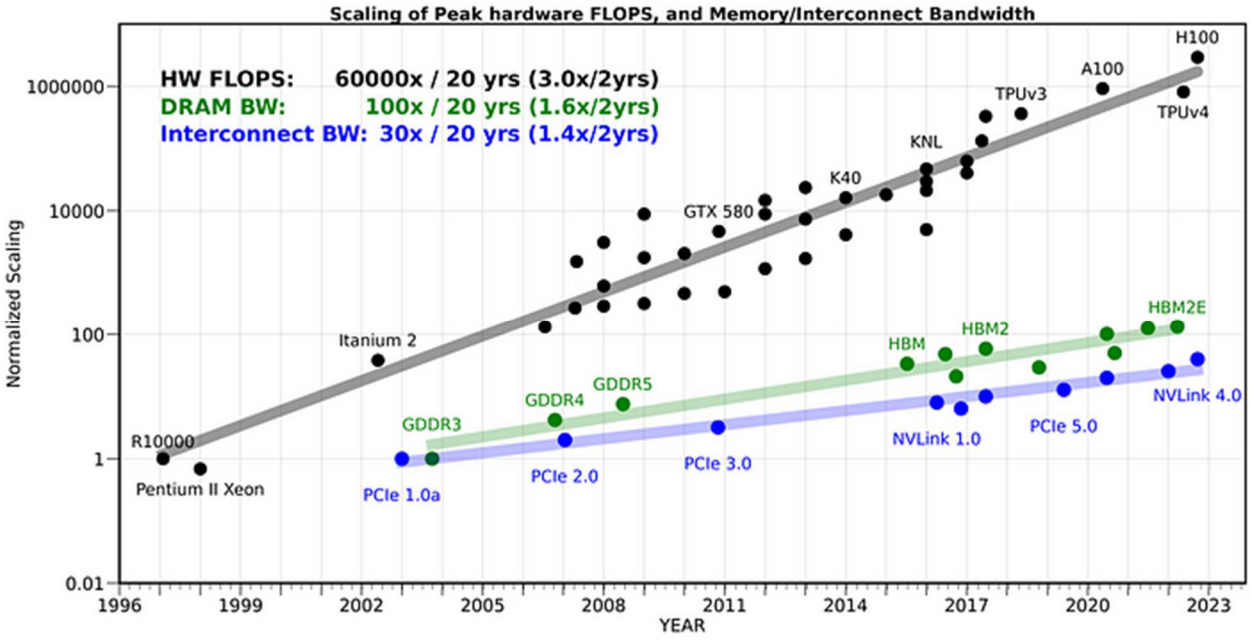


Figure 2. Study from [16] showcasing how data movement is lagging behind computing capabilities.

Additionally, the current dominant strategy for accelerating larger neural network models is to increase core count and hence the processing capabilities of the acceleration platform. To support such massive demands in core count, 2.5 D multi-chiplet solutions are considered [19] [20]. However, due to the slowdown in Dennard scaling, these systems are increasingly inefficient in their power and energy consumption. The latest iteration of the MI series of products from AMD consumes up to 750W of power, while the Blackwell architecture is predicted to consume upwards of 1kW of power [21]. This would severely impact the performance per watt and MAC/Joule metrics of these emerging platforms. Given these DNN applications are to be tackled at server scale and greener server operation is becoming critical [18], [22], exploring energy-efficient

alternatives to the current transistor-based electronic computation, communication, and main memory technologies is crucial.

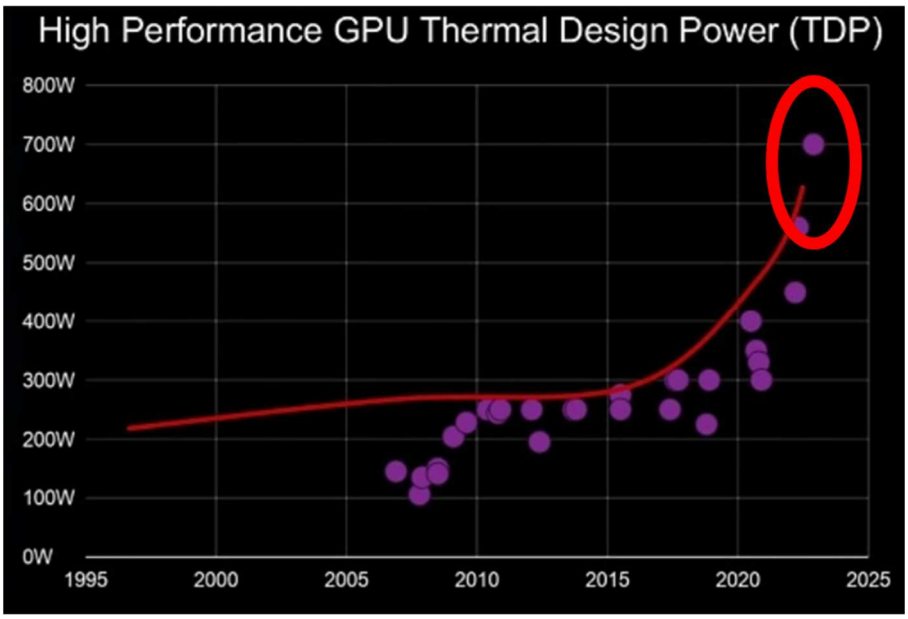


Figure 3. Study from the interview in [23], showcasing the dramatic increase in GPU TDP to provide computation support to growing DNN demands.

1.3. POST-MOORE TECHNOLOGIES FOR ML ACCELERATION

As discussed in Section 1.2, conventional architectures are struggling to meet the computation demands of modern DNNs, owing to architectural deficiencies and physical deficiencies. There are innovations in the architectural space to tackle the challenges in that domain [19], [20]. However, the physical limitations imposed by transistor scaling causing diminishing returns in terms of compute power in the face of compute demand and the energy and power inefficiencies needs addressing. The general tendency in architecture to improve energy and power efficiency is to specialize the architecture for a specific task, and move away from general purpose computing. One can argue the success of the GPU in ML acceleration stems directly from its departure from general purpose computing capabilities, enabling a high core-count manycore system, which could

be repurposed to tackle ML operation rather than graphics generation for PCs. However, even this maynot be sufficient as Figure 3 points out.

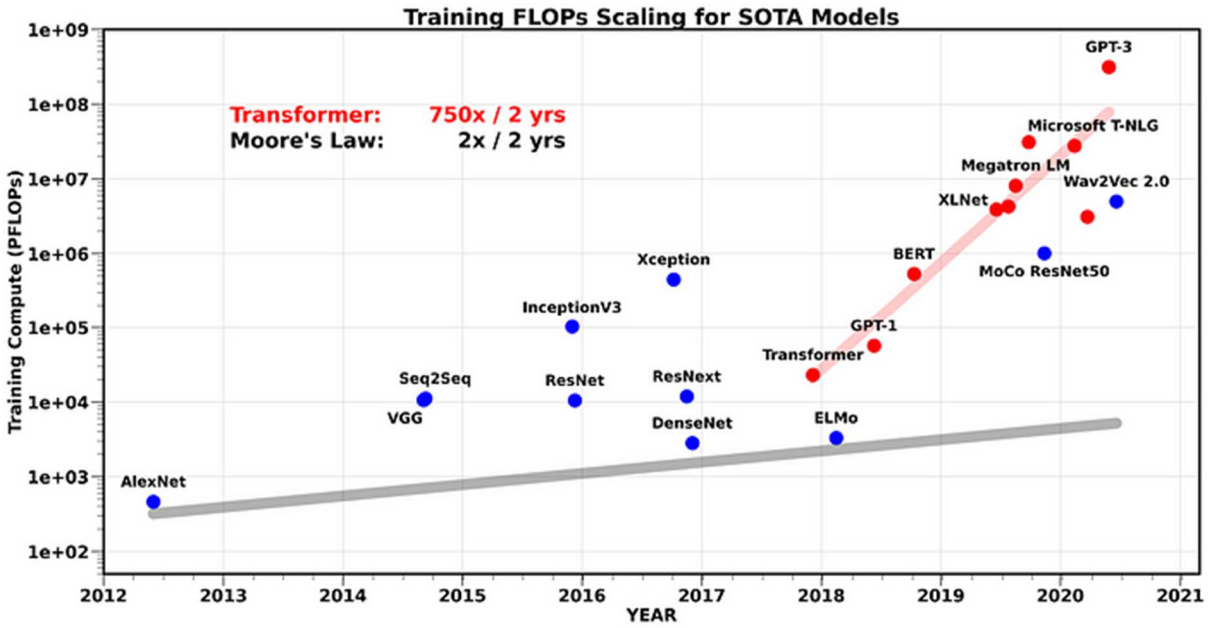


Figure 4. The widening gap between computation capabilities enabled by Moore's Law (grey line) and the computation demand from modern ML workloads (red line) [16].

To tackle this post-Moore era challenge, several new computing paradigms have been proposed, that departs from conventional computer architecture.

In the digital domain, architectures tailored for maximizing the throughput of low-precision linear algebra operations, which are the fundamental operations in any DNN, are proposed. Tensor operations as opposed to MAC operations are prioritized in the TPU architecture [24] and are added to GPUs for enhanced DNN support [11]. Specialized architectures like Apple's Neural Engine [25] and Intel's Habana GOYA [26] also follow the same architecture philosophy. However, these architectures still have architectural deficiencies owing to the separation of memory and processing, leading to the higher memory power and data movement costs discussed.

To tackle this novel computation paradigms are being considered, such as analog computation and neuromorphic computing.

In analog computation, continuous data values are used for computation rather than binary data representation. Analog devices compute directly through physical quantities such as charge, voltage, or current, which can represent data in a naturally parallel and highly efficient manner. This capability is especially advantageous for tasks involving vector-matrix multiplications, prevalent in deep learning algorithms. Analog computation for ML acceleration typically make use of non volatile memory (NVM), typically ReRAM, crossbar arrays for matrix-vector multiplication (MVM) operations [27]. These arrays perform multiplications and additions in a single operation by exploiting Ohm's Law and Kirchhoff's Rule, thereby speeding up the forward and backward propagation in neural networks while using less power than digital counterparts.

Neuromorphic computing is another such novel computation technique, which aims to mimic the human brain to implement systems for AI and AGI. IBM's 4096 core TrueNorth chip which was released in 2014 was one of the earliest high-profile neuromorphic deep learning accelerators [28]. There are many similarities between neuromorphic and analog computation, and one can call neuromorphic computing a subset of analog computation techniques. The difference is that neuromorphic architectures will implement synaptic and neuronal analogs to mimic brain function, which is not a necessity in analog computation. Currently, the most popular method of implementing analog computers is through analog in-memory computing [29], using NVMs.

In memory computing (IMC), also refered to as processing in memory (PIM), is in itself a new computing paradigm. Similar to neuromorphic computing, PIM aims to integrate memory and processing to minimize the data movement issues which plague von Neumann architectures. This

spans digital and analog domains, with PIM architectures proposed for DRAM architectures [30] and NVM memories alike [31].

However, all these architectures still rely on electronic devices, and has to face the fundamental physical issues related to device scaling. As transistor technology approaches the physical limits of material properties and lithography, several significant issues arise, challenging further scaling. This includes, but is not limited to: electron tunneling-induced increase in leakage current, increased process variation vulnerabilities due to lithography limitations, thermal reliability issues, voltage and frequency scaling limitations. All of these issues contribute to system efficiency and throughput. Hence, as throughput and efficiency demands increase, it is prudent to consider other technologies and platforms to implement computation systems on.

1.4. SILICON PHOTONICS FOR ML ACCELERATION

Photonics rely on photons and hence light waves to represent information. This technique, when implemented on a Si-SiO2 platform, is known as silicon photonics, and it's designed to be compatible with CMOS integration. Photonic interconnects are emerging as a leading solution for addressing data movement bottlenecks. Already, photonic links are replacing their metallic counterparts for ultra-fast data transmission across various levels of computing architecture, with current trends moving towards chip-scale integration [32]. The breakthrough of silicon photonics, facilitating the cost-efficient incorporation of optical components using CMOS electronics production methods, has significantly propelled the development of chip-scale photonic interconnects [33]. What's more, operations crucial to deep learning, such as matrix-vector multiplications, are now feasible within the optical domain [34], paving the way for deep learning accelerators powered by silicon photonics for both data transmission and processing. These deep

learning accelerators, based on silicon photonics, stand to offer unmatched energy efficiency and parallel processing capabilities. For example, in performing MAC (Multiply-Accumulate) operations, which are fundamental to deep learning algorithms, photonics-based accelerators can achieve energy efficiencies (MAC/Joules) nearly 1000× greater than most electronic accelerators [35]. Additionally, the throughput of photonic MAC operations can theoretically match the photodetection rates, typically reaching up to hundreds of GHz [36].

Computation in photonics can be executed via coherent or noncoherent (also known as incoherent) analog computational techniques [37]. Coherent photonic computation harnesses the phase of light waves systematically to encode and manipulate data, including operations like multiplication, through interference patterns. This method capitalizes on light's coherent characteristics, such as phase coherence and superposition, allowing for the execution of complex mathematical functions swiftly and precisely. Architectures utilizing coherent computation typically employ Mach-Zehnder interferometers (MZIs) to modify data via constructive or destructive interference using a single wavelength.

In contrast, noncoherent photonic computation does not depend on light's phase information. It instead manipulates light's intensity or amplitude for computation, offering resilience against phase fluctuations and coherence problems that could compromise coherent systems. Noncoherent methods provide a simpler data encoding process and greater robustness, due to fewer noise factors. These qualities render noncoherent computation ideal for various optical signal processing applications, including image processing, sensor data analysis, and basic arithmetic operations. They also facilitate large-scale arithmetic operations using Wavelength Division Multiplexing (WDM), positioning noncoherent photonics as a promising approach for MVM and General Matrix Multiply (GEMM) tasks. Devices in noncoherent systems must be wavelength-sensitive to

exploit WDM signals, making wavelength-selective Micro-Rings (MRs) preferred components in these architectures.

An MR is an on-chip optical resonator that resonates at an optical wavelength matching its resonant wavelength (λ_{MR}). By adjusting λ_{MR}, the resonator can increase losses for specific wavelengths, enabling amplitude modulation essential for noncoherent computation. The two primary adjustment methods are thermo-optic and electro-optic tuning, both affecting the resonator's effective refractive index (n_{eff}) and consequently λ_{MR} ($\lambda_{MR} = 2\pi n_{eff} R$; R = MR radius). Thermo-optic tuning heats the MR with microheaters, whereas electro-optic tuning uses free carrier injection via a PN junction in the MR [37]

Prior work demonstrates several noncoherent computational architectures leveraging MRs for efficient, high-throughput, low-energy machine learning inference acceleration through a technique known as broadcast and weight (B&W) [38]. In this setup, MRs are adjusted to reflect a fixed matrix, and vectors are introduced as either amplitude-modulated wavelengths or through a subsequent array of tunable MRs downstream from the initial MR array's output. The light's interaction with the MRs alters its amplitude, signifying a multiplication operation. Summing several such light signals with a photodetector facilitates simultaneous MAC operations. Here, the degree of WDM signals corresponds to the MR array size, enabling parallel processing of *n* MAC operations. Through dataflow orchestration to enable massively parallel MAC operations from the DNN's execution graph, it should be possible to execute training and inference for these models with significantly higher throughput and energy efficiency, using noncoherent photonics.

The body of works that form this thesis has focused on using noncoherent photonics for ML acceleration and addressing various challenges associated with it. We shall discuss these challenges briefly in the following subsection.

1.5. CHALLENGES IN NONCOHERENT SILICON PHOTONICS

Access to WDM-based communication and computation in noncoherent photonics is well suited for DNN acceleration, as it facilitates both large-scale data movement and highly parallel MAC operations. However, to realize such as system there are several challenges to be overcome:

1.5.1. RESONANCE TUNING CHALLENGES

The computation technique in photonics aims to imprint data onto some physical characteristics of light waves, making them analog computation techniques. As discussed, in noncoherent computing the data is imprinted onto the amplitude of the light wave. This is done by tuning the resonant wavelength (λ_{MR}) of the MR. As the resonant frequency is tuned, the incoming signal will interact with a different region of the MR's response curve, altering its amplitude.

To enact the λ_{MR} shift, one of two main techniques is employed conventionally: thermo-optic (TO) tuning or electro-optic (EO) tuning. TO tuning makes use of heaters to heat the MR bulk, altering its n_{eff}, and thus the λ_{MR}. This technique offers a large range of tunability, but because it relies on thermal conductivity this is a slow and energy-intensive process, with ~27 mW/FSR (FSR=free spectral range) power consumption and the tuning latency in the μs range [39]. EO tuning is faster with latencies in ns range, owing to its tuning mechanism relying on free carrier injection into MR bulk, and power consumption at ~4 μW/nm for λ_{MR} shift [40]. However, EO tuning offers significantly lower tuning range as the PN junction used for carrier injection can become saturated.

Selecting EO tuning is beneficial for throughput, and bringing operation frequency closer to the photodetection rate, however, in situations where the larger tuning range of TO tuning is necessary (Subsection 1.5.3), the latency must be endured.

1.5.2. ELECTRICAL-OPTICAL INTERFACE MANAGEMENT

While we have talked about photonic systems and their benefits, the fact remains that the analog domain operation is highly application-specific and is ill-suited for general-purpose computation, at least in the way general-purpose computing is today. Hence, as a system, photonic accelerators must co-exist with a digital CPU and main memory. This necessitates an electrical-optical interface. This interface is comprised of the following devices: analog-to-digital converters (ADCs), digital-to-analog converters (DACs), photodetectors, and trans-impedance amplifiers. Other than photodetectors which can operate at ps latencies [41], these devices are slow operating in comparison to photonic device speeds and become extremely power-hungry at higher frequencies of operation. Thereby limiting the achievable frequency of operation and thus the throughput of the noncoherent accelerator.

Additionally, noncoherent systems require several hundreds or more DACs per accelerator as each MR requires one to feed its tuning mechanism and a similar number of ADCs to gather the outputs to report back to the digital system. This limits the achievable energy efficiency of the system. Additionally, another aspect of DNN acceleration comes into prominence, parameter quantization.

Usually, DNNs are trained with 32-bit floating point (fp32) weight parameters. Employing 32-bit ADCs and DACs would increase the power consumption of the accelerator to prohibitively high levels. Additionally, floating point operations does not translate to amplitude based analog data representation.

1.5.3 RELIABILITY CHALLENGES

Photonic systems rely on analog domain operations. To ensure reliable operation of the system in analog domain, several device-level and architecture-level considerations are to be made. Main challenges that need to be addressed to ensure reliable operation of photonic systems include: fabrication process variation, thermal variation, and crosstalk noise.

1.5.3.1. FABRICATION PROCESS VARIATION

A photonic integrated circuit, such as a DNN accelerator, a PNoC, or a photonic memory, requires thousands of MRs, requires precise matching of these MRs to their resonance wavelengths. However, MR resonant wavelength is considerably sensitive to the dimensions of the waveguides which construct the MR. Any change in these dimensions will lead to a resonant wavelength shift ($\Delta\lambda_{MR}$), inducing errors in the data being represented by wavelength amplitude tuning (Section 1.5.1).

The changes in the MR dimensions from its design dimensions are incurred through lithography imperfection, and are referred to as fabrication process variations (FPVs). Substantial FPV can induce $\Delta\lambda_{MR}$ of several nms, which needs to be corrected to ensure reliable operation of the photonic DNN accelerator.

1.5.3.2. THERMAL VARIATION

MRs are significantly sensitive to thermal variations, i.e. the change in temperature of the chip or parts of it during chip operation. Presence of thermal variations induce $\Delta\lambda_{MR}$ as n_{eff} of the MR changes with temperature. The main sources of temperature in an electro-optic system are electrical circuitry operation and TO tuning mechanisms. Architectural or packaging techniques maybe implemented to separate electronic thermal sources from the optical circuitry, but the tuning

mechanism is integral to controlling the optical devices, and even for correcting reliability issues such as FPV (Section 1.5.3.1). TO tuning based thermal variation can also impact groups of MRs as the heat variation across the group, from varying tuning levels across MRs, can alter the amount of thermal energy reaching individual MRs and thus alter the tuning levels from desired levels, thereby acting as an additional source of noise.

1.5.3.3. CROSSTALK MITIGATION

Crosstalk noise in noncoherent photonics is classified into two: homodyne crosstalk and heterodyne crosstalk. Crosstalk, as the name indicates, is unwanted interaction between channels, leading to power leakage between them which in turn acts as noise. Homodyne crosstalk is power leak within the same channel due to imperfect filtering from MRs, the noise component can interact with MRs downstream with the same λ_{MR}, leading to imperfect calculations. Heterodyne crosstalk is crosstalk between different channels and is caused by narrow channel gaps and how spectrally wide the MR response is (also referred to as Q-factor; smaller the Q-factor, narrower the response).

As the number of wavelengths used in a photonic system increase, more channels need to be accommodated within the wavelength range of the laser (E.g. C-band i.e. the 1530-1565nm range). This combined with the wavelength response range of MRs can lead to scenarios where MR_1 which is designed to be responsive to $channel_1$ with a central waveguide of λ_1 may have some response overlap with $channel_2$. This will again lead to improper amplitude modulation operations and incorrect calculations.

1.5.4. LASER POWER CHALLENGES

As discussed in Section 1.5.2, the electrical-optical interface can be extremely power and energy intensive. However, the laser power required to feed the optical cicuit can be high as well.

Signal transmission within Si-SiO_2 waveguides is inherently lossy with various loss sources such as propogation loss and bending loss. Encountering photonic devices also lead to passive losses due to the mechanisms discussed in Sections 1.5.3.1 or 1.5.3.3. Tuning mechanisms can also incur losses, along with the presence of splitters and couplers.

It is essential to consider these losses and carefully architect around them to prevent prohibitively high laser power consumption in photonic systems.

1.5.5. ARCHITECTURAL CHALLENGES

Along with all the challenges mentioned so far, there is also architectural-level challenges to realize a DNN inference accelerator through noncoherent silicon nanophotonics. Due to laser power dissipation (Section 1.5.4) and reliability (Section 1.5.3) concerns having a "one-shot" architecture that encompasses the entire model is not practical. Additionally, conventional general-purpose processing and main memory systems are electronic in nature, necessitating frequent electrical-optical interfacing between the accelerator and the processor/memory systems. This means several considerations must be made to minimize data movement between optical and electronic domains, so as to minimize the energy and latency costs at the electrical-optical interface (Section 1.5.2).

Specifically for accelerator architectures, there is also the need to consider the DNN architecture itself, because of the above mentioned issues. DNN-specific pipelining to minimize the electrical-optical-electrical conversions, while maximizing the utilization of the data that is obtained per conversion is necessary for ensuring energy efficiency and throughput.

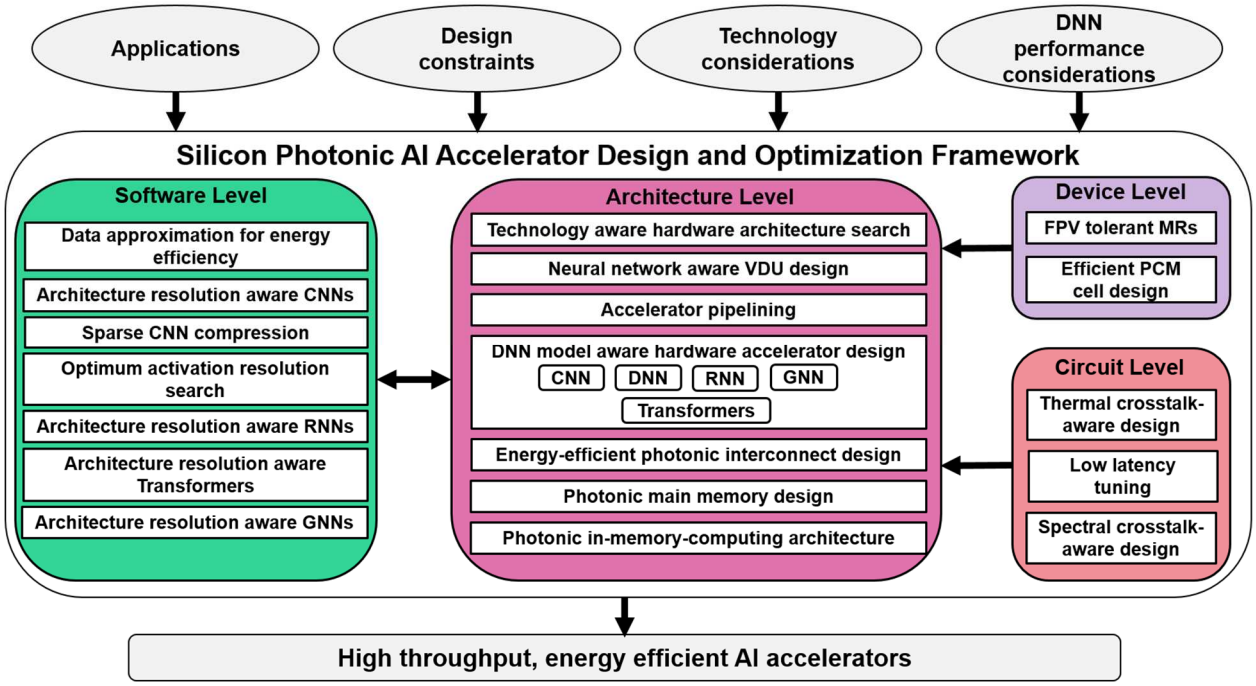


Figure 5. Outline of contributions of this thesis; Abbreviations used: VDU: vector dot product unit; DNN: Deep Neural Network; CNN: Convolutional Neural Network; RNN: Recurrent Neural Network; GNN: Graph Neural Network.

1.6. THESIS OVERVIEW

To address the challenges presented in the previous section, we propose a hardware-software codesign framework for silicon photonic AI accelerators, in this thesis. This framework is a crosslayer approach, considering solutions across multiple layers of the system to these challenges. Our framework considers solutions that combine enhancements at the system-level, architecture-level, circuit-level, and device-level towards the design of reliable, energy-efficient, and high throughput Photonic DNN accelerator architectures. Figure 5 gives a high-level overview of the thesis [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52]. The rest of this thesis is organized as follows:

In chapter 2, we propose a novel laser power management framework for PNoCs called ARXON [42], which performs communication loss- and quality-of-service-aware approximation

of data packets in the NoC. ARXON uses a LUT-based loss directory between source-destination pairs along with an optical link manager circuit to tune the vertical cavity surface emission lasers (VCSELs) to desirable power output levels. The framework dynamically assesses the packet loss and decides to either tune the VCSELs which generate n least significant bits (LSBs) of the data in the packet to P_r% output power or truncate these n LSBs, thereby turning of the corresponding VCSELs. The n and P_r values selected for this approximated data transfer is application dependant and must be determined by profiling the application. Additionally, we also considered a heterodyne crosstalk mitigation strategy by encoding the data packets. The encoding strategy, adopted from [53], ensures that the delivered data reaches with significantly reduced bit-error-rate (BER). Finally, we also explored multi-bit data transfer to further reduce laser power consumption. ARXON adopted 4 pulse amplitude modulation (4-PAM) over the conventional on-off keying (OOK), allowing 2 bits to be transferred per wavelength, thus reducing the wavelength requirement, the heterodyne crosstalk, and laser power requirement.

In chapter 3, we present the noncoherent photonic accelerator for CNN inference acceleration, entitles CrossLight [43]. The architecture presented in this chapter is cross-layer optimized for tackling many of the challenges mentioned in Section 1.5. At the device-level, CrossLight employs FPV resilient MRs, reducing the FPV correction power requirement. At the electro-optic circuit level, CrossLight proposed a tuning circuits which combined EO and TO tuning techniques, where TO tuning is used sparingly for FPV and thermal variation corrections, while EO tuning is used for amplitude modulation in MRs for CNN parameter representation. This combined tuning circuit reduces TO tuning power requirements, while allowing higher throughput owing to the lower ns delays offered by EO tuning. Additionally, to reduce thermal crosstalk from any TO tuning used during operation, CrossLight employed thermal Eigenmode decomposition (TED) algorithm

described in [54]. To reduce the laser power requirement and the heterodyne crosstalk, CrossLight made use of a wavelength reuse strategy, where the same set of wavelengths were reused across vector dot product units (VDUs). The VDUs themselves were designed for varying vector sizes so that smaller and larger vector sizes across layers (convolution layers and fully connected layers) can be operated on efficiently. Finally, the CNN models were quantized to 16 bits by considering the thermal and heterodyne crosstalk sources to ensure correct operations in the analog photonic domain and so that electrical-optical interface energy and power requirements are reduced.

In chapter 4 we present ROBIN architecture [44] which was designed for binarized neural network (BNN) acceleration. BNNs are DNNs which utilize single parameters rather than the conventional multi-bit parameters. BNNs offer significantly lower memory footprint and can offer higher energy efficiency in hardware operation owing to the much simpler single-bit operations involved. However, BNNs have reduced inference accuracy over DNNs. BNN-specific acceleration specifically offers reduction in power and energy requirement at the electrical-optical interface, owing to the single bit parameters in BNNs. This ensured ROBIN can scale out significantly in terms of number of VDUs and vector sizes, while ensuring high energy efficiency and throughput. To increase BNN accuracies, we considered a partially binarized neural network, where the weight parameters remained single bit while the activation parameters were 4-bit values. ROBIN also employed high bandwidth MR filters, which could tune all the wavelengths in a waveguide simulataneously, to implement a photonic batch normalization operation, in order to reduce data movement between electrical and optical domains.

In chapter 5, we tackle how to appraoch unstructured sparsity in DNNs in photonic accelerators through the SONIC architecture [45]. Sparsity in a DNN refers to presence of zero valued parameters in the weight and activation matrices, these values incur energy and throughput loss in

accelerators as they when they are encountered no operation is performed, but the associated dat movement and processing unit allocation happens. Unstructured sparsity appears without any specific pattern to it, making it significantly more difficult to tackle efficiently at the hardware level. However, unstructured sparsity offers better neural network accuracy at the same sparsity level, when compared to a structured sparsification approach. To tackle this SONIC employed directly tuned VCSELs in its VDUs, where if the parameter is 0, the VCSEL does not generate corresponding wavelength, saving on laser power. Additionally, since SONIC performs vector dot product operations instead of MAC operations, the loss of throughput is minimal.

So far the architectures presented had fixed parameter sizes they could handle. If the software model were to be quantized to a lower parameter size b, the architectures would incur unnecessary operational losses by employing ADC and DAC units which would have a resolution of the original larger parameter size a. To tackle this issue we propose the HQNNA architecture [46], which is presented in chapter 6. HQNNA combined the crosslayer approaches employed before, and additionally made use of a combination of WDM and time division multiplexing (TDM) to handle a variety of parameter sizes. This allows HQNNA to tackle heterogeneously quantized DNNs effectively with high energy efficiency, while suffering minimal loss in throughput.

In chapter 7, we propose a noncoherent photonic accelerator designed to tackle any recurrent neural network (RNN) variant [47]. RNNs were prominent neural networks used for temporal sequence learning and even natural language processing, before the advent of the attention mechanism [10]. RNNs are not feedforward neural networks, and has directed circular loops within architecture, which allows them retention and memory capabilities. Simple RNNs are susceptible to the vanishing/exploding gradient problem, limiting the size of the sequence they could retain for processing and the data complexity they could tackle. Gated recurrent units (GRUs) and long

short-term memories (LSTMs) are complex variants of RNNs with more complex layers. To tackle these complex DNN architectures, RecLight employs RNN-specific hardware pipelining, along with optical non-linearity-based activation function implementations. RecLight implements $sigmoid$ and $tanh$ functions optically, as RNN, GRU, and LSTM cells make use of these nonlinear operations. We also present a detailed crosstalk analysis so that we can determine which parameter resolution is suitable for photonic RNN operations, along with helping energy-efficiently design the RNN-specific VDU and RecLight's electrical-optical interface.

In chapter 8, we present GHOST [48] a noncoherent photonic graph neural network (GNN) accelerator. GHOST ememploys GNN-aware hardware pipeline which minimizes electrical-optical conversions. We perform detailed architectural optimization for the efficient handling and acceleration of diverse graph structures and GNN model architectures on GHOST. Finally, GHOST also employs detailed photonic device and circuit-level optimizations to mitigate crosstalk noise so that error-free GNN operations can be ensured. GHOST additionally reduces DAC requirement through employing a DAC sharing technique. Finally data movement between various processing blocks in the hardware pipeline is minimized through intelligent workload balancing. Similarly, chapter 9 presents TRON [49], which was designed to tackle transformer-specific challenges in noncoherent photonic hardware pipelining.

In chapter 10, we explore how to scale out photonic architectures across chiplets [50]. The proposed 2.5D architecture makes use of photonic interposers for inter-chiplet communication, ensuring higher bandwidth of communication, thus increasing throughput of operation.

Throughout this discussion, we have mentioned how important for the architecture it is to have an efficiently designed electrical-optical interface to reduce energy and latency costs associated with data movement across domains. However, if the data is stored optically and can be retrieved

optically, much of the conversion costs can be significantly reduced, if not removed entirely, thus enabling significantly better throughput and energy efficiency in photonic accelerator architectures. To enable this, there is the need for a photonic main memory. We tackle the challenge of architecting a photonic main memory using phase change materials (PCMs) in COMET [51], which is presented in chapter 11. COMET is crosslayer optimized to ensure error free storage in memory, and reliable and error-free data retrieval from the main memory architectures. COMET also has several architecture-specific optimizations to enable low power operation, so that it can be competent among DRAM architectures.

A viable main memory architecture opens the possibility of processing-in-memory (PIM) architectures. Using COMET as a backbone, we explore to how to exploit the inherent hardware parallelism in a memory architecture for CNN inference acceleration, in OPIMA [52], in chapter 12. OPIMA identifies several challenges in using a PCM-based main memory architecture for PIM and then offers architectural solutions to those challenges, resulting in a high throughput, highly energy efficienct photonic PIM architecture capable of CNN inference acceleration.

Finally, chapter 13 concludes the thesis. We summarize our contributions throughout the body of research and make recommendations for future research.

CHAPTER 2. ENERGY EFFICIENT PHOTONIC NETWORK-ON-CHIP USING MULTI-LEVEL SIGNALING AND QOS-AWARE DATA TRANSFER

To match the increasing demand in processing capabilities of modern applications, the core count in emerging manycore systems has been steadily increasing. For example, Intel Xeon processors today have up to 56 cores [12], while NVIDIA's GPU's have reported over 8000 shader cores [55]. Emerging application-specific processors are pushing these numbers to new highs, e.g., the Cerebras AI accelerator has over 400,000 light weight cores [56]. The increasing number of cores creates greater core-to-core and core-to-memory communication.

Conventional metallic interconnects and electrical networks-on-chip (ENoCs) already dissipate very high power to support the high bandwidths and low-latency requirements of data-driven parallel applications today and are unlikely to scale to meet the demands of future applications [57]. Fortunately, chip-scale silicon photonics has emerged in recent years as a promising development to enhance ENoCs with light speed photonic links that can overcome the bottlenecks of slow and noise-prone electrical links. Silicon photonics can enable photonic NoCs (PNoCs) with a promise of much higher bandwidths and lower latencies than ENoCs [58].

Typical PNoC architectures employ several photonic devices such as photonic waveguides, couplers, splitters, and multi-wavelength laser sources, along with microring resonators (MRs) as modulators, detectors, and switches [58]. A laser source (either off-chip or on-chip) generates light with one or more wavelengths, which is coupled by an optical coupler to an on-chip photonic waveguide. This waveguide guides the input optical power of potentially multiple carrier wavelengths (referred to as wavelength-division-multiplexed (WDM) transmission), via a series of optical power splitters, to the individual nodes (e.g., processing cores) on the chip. Each

wavelength serves as a carrier for a data signal. Typically, multiple data signals are generated at a source node in the electrical domain as sequences of logical 0 and 1 voltage levels. These input electrical data signals are coupled with (i.e., modulated onto) the wavelengths using a group (bank) of modulator MRs (e.g., 64-bit data modulated on 64 wavelengths), typically using On-Off Keying (OOK) modulation. Subsequently, the carrier wavelengths are routed over the PNoC till they reach their destination node, where the wavelengths are filtered and dropped into the waveguide by a bank of filter MRs that maneuver the wavelengths to photodetectors to recover the data in the electrical domain. Each node in a PNoC can communicate to multiple other nodes through such WDM-enabled photonic waveguides in PNoCs.

Unfortunately, optical signals accumulate losses and crosstalk noise as they traverse PNoCs, necessitating high signal power from the laser for signal-to-noise ratio compensation and to guarantee that the signal can be received at the destination node with sufficient power to enable error-free recovery of the transmitted data. Moreover, the sensitivity of an MR to the wavelength it is intended to couple with is related to its physical properties (e.g., radius, width, thickness, refractive index of the device material) that can vary with fabrication and thermal variations. To rectify these problems, MRs must be "tuned" to correct the impact of variations either by free-carrier injection (electro-optic tuning) or thermally tuning the device (thermo-optic tuning). Such tuning entails energy and power overheads, which can become significant as the number of MRs in PNoCs increases. Novel solutions are therefore urgently needed to reduce these power overheads, so that PNoCs can serve as a viable replacement to ENoCs in emerging and future manycore architectures.

One promising direction towards this goal is approximate computing. As computational complexity and data volumes increase for emerging applications, ensuring fault-free computing

for them is becoming increasingly difficult, for various reasons including: (i) increasing resource demands for big-data processing limit the resources available for traditional redundancy-based fault tolerance, and (ii) the ongoing scaling of semiconductor devices makes them increasingly sensitive to variations, e.g., due to imperfect fabrication processes. Approximate computing, which trades off "acceptable errors" during execution to reduce energy and runtime, is a potential solution to both these challenges [59]. With diminishing performance-per-watt gains from Dennard scaling, leveraging such aggressive techniques to achieve higher energy-efficiency is becoming increasingly important.

In this chapter, we explore how to leverage data approximation to benefit the energy and power consumption footprints of PNoC architectures. To achieve this goal, we analyze how data approximation impacts the output quality of various applications, and how that will impact energy and power requirements for laser operation, transmission, and MR tuning. Our proposed framework, called ARXON, implements an aggressive loss-aware approximated-packet-transmission solution that reduces power overheads due to the laser, crosstalk mitigation, and MR tuning.

The novel contributions of this chapter are as follows:

- We develop an approach that relies on approximating a subset of data transfers for applications, to reduce energy consumption in PNoCs while still maintaining acceptable output quality for applications;
- We propose a strategy that adaptively switches between two modes of approximate data transmission, based on the photonic signal loss profile along the traversed path;

- We evaluate the impact of utilizing multilevel signaling (pulse-amplitude modulation) instead of conventional on-off keying (OOK) signaling during approximate transfers for achieving even greater energy-efficiency;
- We explore how adapting existing approaches towards MR tuning and crosstalk mitigation can help further reduce power overheads in PNoCs;
- We evaluate ARXON on multiple applications and show its effectiveness over the best-known prior work on approximating data transfers over PNoC architectures;

2.1. RELATED WORK

By carefully relaxing the requirement for computational correctness, it has been shown that many applications can execute with a much lower energy consumption and without significantly impacting application output quality. Some examples for approximation-tolerant applications that can save energy through this approach include audio transcoding, image processing, encoding/decoding during video streaming [60], and big-data applications [61]. The fast-growing repository of machine-learning (ML) applications represents a particularly promising target for approximation because of the inherent resilience to errors in most ML applications. As an example, it is possible to approximate the weights (e.g., from 32-bit floating-point to 8-bit fixed point) in convolutional and deep neural networks and with negligible changes in the output classification accuracy [62]. Many other approaches have been proposed for ML algorithm-level approximations [63], [64]. With the introduction of ML applications into resource-constrained environments such as mobile and IoT platforms, there is growing interest in utilizing approximated versions of ML applications for faster and lower-energy inference [65].

In general, the approximate computing solutions proposed to date can be broadly categorized into four types based on their scope [66]: hardware, storage, software, and systems. The approximation of hardware components allows for a reduction in their complexity, and consequently their energy overheads [67]. For instance, an approximate full adder can utilize simpler approximated components such as XOR/XNOR based adders and pass transistor-based multiplexers [68], [69]. Additional reduction in circuit complexity and power dissipation can be enabled by avoiding XOR operations [70]. Techniques for storage approximation can include reducing refresh rates in DRAM [71], which results in a deterioration of stored data, but at the advantage of increased energy-efficiency. Approaches for software approximation include algorithmic approximation that leverages domain specific knowledge [72]. They may also refer to approximating annotated data, variables, and high-level programming constructs (e.g., loop iterations), via annotations in the software code [73], [74]. At the system level, approximation involves modification of architectures to support imprecise operations. Attempts to design approximate NoC architectures fall under this category.

Several efforts have attempted to approximate data transfers over ENoC architectures by using strategies that reduce the number of bits or packets being transmitted to reduce NoC utilization, and thus reduce communication energy. An approximate ENoC for GPUs was presented in [75], where similar data packets were coalesced at the memory controller, to reduce the packets that traverse over the network. A hardware-data-approximation framework with an online data error control mechanism, which facilitates approximate matching of data patterns within a controllable value range, for ENoCs was presented in [76]. In [77], traffic data was approximated by dropping values from a packet before it is sent on to the ENoC, at a set interval. The data is then recreated at the destination nodes using a linear interpolator-based predictor. A

dual voltage ENoC is proposed in [78], where lower-priority bits in a packet are transferred at a lower voltage level, which can save energy at the cost of possible bit flips. In contrast, the higher priority bits of the packet, including header bits, are transmitted with higher voltage, ensuring a lower bit-error rate (BER) for them. All of these approaches focus on approximations for ENoCs. The complex and unique design space of approximation techniques for PNoCs remains relatively unexplored.

A recent work [79]explored the use of approximate data communication in PNoCs for the first time. The authors explored different levels of laser power for transmission of bits across a photonic waveguide, with a lower level of laser power used for bits that could be approximated, but at the cost of higher BER for these bits. The work focused specifically on approximation of floating-point data, where the least significant bits (LSBs) were transmitted at a lower laser-power level. However, the specific number of these bits to be approximated as well as the laser-power levels were decided in an application-independent manner, which ignores application-specific sensitivity to approximation. Moreover, the laser-power level is set statically and without considering the dynamic optical loss that photonic signals encounter as they traverse the network. In [80], we proposed LORAX framework that improved upon the work in [79] by using a loss-aware approach to adapt laser power at runtime for approximate communication in PNoCs. We analyzed the impact of adaptive approximation, varying laser-power levels, and the use of 4-pulse amplitude modulation (PAM4) on application output quality, to maximize application-specific energy savings in an acceptable manner. There may be apparent similarities between these approaches and works in say, [77] or [78], but the design considerations, modeling, and implementation in hardware required for PNoC are very different from an ENoC. For example, considering PAM4 for energy savings is only possible in PNoCs.

The ARXON (AppRoXimation framework for On-chip photonic Networks) framework presented in this article improves upon LORAX in multiple ways through: (i) considering integer data for approximation in addition to floating-point data (LORAX only considered floating-point data); (ii) integrating the impact of fabrication-process variations (PV) and thermal variations (TV) on MR tuning and leveraging it for energy savings; (iii) approximating error correction techniques, which are commonly used in PNoCs, to save more energy; and (iv) analyzing the potential for approximation for a much broader set of applications, and across multiple PNoC architectures. Section 2.4 describes our proposed ARXON framework in detail with evaluation results presented in Section 2.5.

2.2 DATA FORMATS AND APPROXIMATIONS

2.2.1 FLOATING POINT DATA

In many applications, floating-point data can be safely considered for approximation and without impacting the overall quality of the output from the approximation, as explored and demonstrated in [80]. The IEEE-754 standard defines a standardized floating-point data representation, which consists of three parts: sign (S), exponent (E), and mantissa (M), as shown in Figure 6. The value of the data stored is:

$$X = (-1)^S \times 2^{E-bias} \times (1 + M) , \quad (1)$$

Where X is the floating-point value. The *bias* values are 127 and 1203, respectively, for single and double precision representation, and are used to ensure that the exponent is always positive, thereby eliminating the need to store the exponent sign bit. The single precision (SP) and double precision (DP) representations vary in the number of bits allocated to the exponent and mantissa (see Figure 6). E is 8 bits for SP and 11 bits for DP, while M is 23 bits for SP and 52 bits for DP.

Also, S is 1 bit for both cases. From (1), we can observe that the S and E values notably affect the value of X. But X is typically less sensitive to alterations in M in many cases. M also takes up a significant portion of the floating-point data representation. We consider S and E as MSBs that should not be altered, whereas M makes up the LSBs that are more suitable for approximation to save energy during photonic transmission.

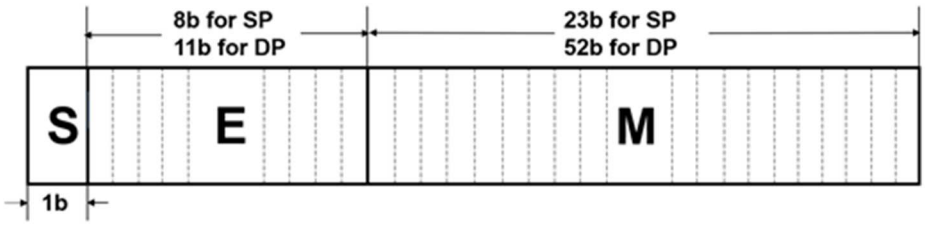


Figure 6. IEEE-754 floating-point representation.

2.2.2. INTEGER DATA

It is more challenging to approximate integer data as it does not have a standard separation similar to the IEEE754 standard for floating-point data. An integer data value is usually represented as an N-bit chunk of data that can be signed or unsigned. If unsigned, the N-bits of data can be used to represent an integer value in the range from 0 to 2^N-1. If signed, the most significant bit represents the sign bit, and the remaining N-1 bits represent an integer value in the range from – (2^{N-1}-1) to + (2^{N-1}-1). The number of bits, N, in an integer data word can change depending on the usage or application. N is usually in the range from 8 to 64 bits in today's platforms. Therefore, a generalized approach to approximate the integer data values is challenging. As a result, we have opted for an application-specific approach, where we identify possible integer variables that have larger than required size, depending on the values they handle. We deem the size of an unsigned integer variable as larger than required, if the most significant bits of the variable are not holding any useful information. We approximate such unnecessarily large

unsigned-integer variables by truncating their MSBs. We also consider LSB approximation for integer packets, when viable. We found that integer data is generally not as tolerant to LSB approximation as floating-point data, so this approach cannot be as aggressive as LSB approximation in floating-point data and is thus used sparsely in our proposed framework.

2.2.3 APPLICATIONS CONSIDERED FOR APPROXIMATION

We evaluate the breakdown of integer and floating-point data usage across multiple applications, to establish how effective an approach that focuses on approximating floating-point LSB data and integer MSB data can be. We selected the ACCEPT benchmark suite [72], which consists of several applications, including some from the well-known benchmark suite PARSEC [81], that have been shown to have a relatively strong potential for approximations. While the applications in this suite may be executed on a single core, to adapt these to a PNoC-based multi-core platform with 64 cores, we used a multi-application simulation approach where the applications were replicated across the 64 cores to emulate multi-application workloads on real systems. Along with the applications from [72] we also considered several neural-network applications from the tinyDNN [82] benchmark suite to see how our approximation framework would fare for ML applications. The multi-application simulation approach was adopted, as in a real many-core system, multiple applications will be running and competing for on-chip resources simultaneously.

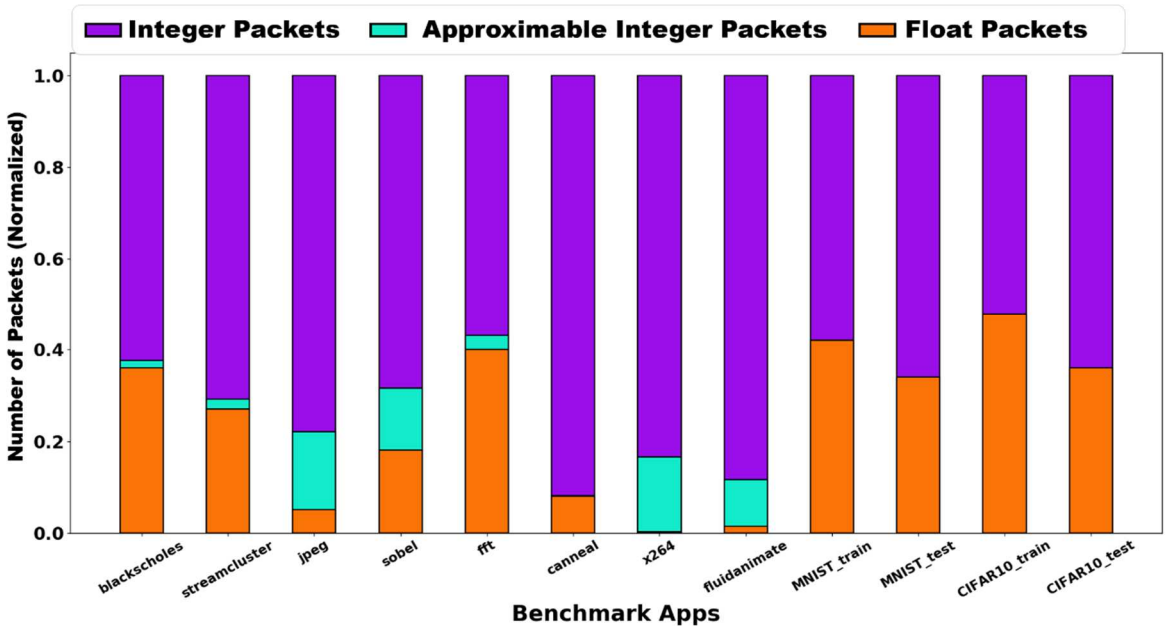


Figure 7. Characterization of applications considered for evaluation.

We used the gem5 [83] full system-level simulator and the Intel PIN tool [84] in tandem to count the total number of integer and floating-point packets in transit across the memory hierarchy during the simulations. Figure 7 shows the breakdown of the floating-point and integer packets across the applications for large input workloads. We considered all floating-point data packets as candidates for approximation. As for integer packets, we identified specific variables and a subset of their bits ("approximable integer packets") that can be approximated safely. The goal while selecting floating-point and integer packets for approximation was to keep application-specific error to below 10% of the original output. It can be observed that while a majority of the applications have integer packets that cannot be approximated without hurting output quality significantly, most of these applications have a non-trivial percentage of their packets that can be approximated. This is a promising observation that motivates our framework. But before we describe our proposed framework in detail (Section 2.4), we briefly cover challenges in PNoCs

related to crosstalk and signal loss (Section 2.5), which our approximation approach can leverage for energy savings.

2.3 CROSSTALK AND OPTICAL LOSS IN PNOCS

The overall data movement on the chip increases as the number of on-chip processing elements increases, and applications utilize more data. This requires a larger number of photonic waveguides, wavelengths, and MR devices to support the increased communication. However, using a larger number of photonic components makes it challenging to maintain acceptable BER and achieve sufficient signal-to-noise ratio (SNR) in any PNoC architecture due to signal optical loss and crosstalk noise accumulation in photonic building blocks [32].

Light propagation in photonic interconnects relies significantly on the precise geometry adjustment of photonic components. Any distortion in waveguide geometries and shape can notably impact the optical power and energy-efficiency in waveguides. For instance, sidewall roughness due to inevitable lithography and etching-process imperfections can result in scattering, and hence optical losses in waveguides [85]. In addition to such propagation loss, there is optical loss whenever a waveguide bends (i.e., bending loss), or when a wavelength passes (i.e., passing loss) or drops (i.e., drop loss) into an MR device. High signal losses require increased laser power to compensate for the loss and ensure appropriate optical-power levels at destination nodes where the signals are detected.

Crosstalk is another inherent phenomenon in photonic interconnects that degrades energy-efficiency and reliability. Crosstalk occurs due to variations in MR geometry or refractive index and imperfect spectral properties of MRs, which can cause an MR to couple optical power from another optical channel/wavelength (which acts as noise) in addition to its own optical channel

(i.e., resonant wavelength). Such crosstalk noise is of concern in dense-wavelength-division multiplexed (DWDM) waveguides, necessary to support a higher bandwidth for emerging manycore platforms, where multiple optical channels exist with a small (*e.g.*, <1 nm) channel spacing. In such DWDM systems, not only will optical signals in each channel suffer from optical loss, but inter- and intra-channel crosstalk accumulating on optical signals can severely reduce SNR and increase BER. Reducing crosstalk is challenging and techniques to minimize crosstalk (e.g., [86]) introduce further power and latency overheads.

It should be noted that the optical-power loss and crosstalk noise from a single silicon photonic device (*e.g.*, MR) can be very small, and hence negligible [87]. However, in PNoCs integrating a large number of such devices (e.g., hundreds of thousands of MRs), the small power loss and crosstalk noise at the device-level accumulate to a point that they can severely reduce the performance and energy-efficiency in such architectures. In our proposed *ARXON* framework, as we are considering approximated data packets, we can intelligently relax crosstalk-mitigation mechanisms and optical loss compensation for the approximated bits, to aggressively reduce power and energy consumption overheads.

2.4. ARXON FRAMEWORK: OVERVIEW

This section discusses the components of our *ARXON* framework. Section 2.4.1 provides an overview of our loss-aware laser power optimization strategy. Sections 2.4.2 and 2.4.3 discuss how crosstalk mitigation and tuning can be relaxed to save power during approximate-bit transfers. Lastly, Section 2.4.4 describes the integration of multilevel signaling to further reduce power dissipation during approximate communication in PNoCs.

2.4.1. LOSS-AWARE LASER POWER MANAGEMENT FOR APPROXIMATION

Optical signals transmitted over a waveguide (photonic link) undergo attrition due to various optical losses they encounter along the path from a source to a destination, as discussed in section 2.3. To express how these optical losses tie in with the initial laser power provisioned to the optical signals in the waveguide, we can use the following model [88]:

$$P_{laser} - S_{detector} \geq P_{phot_loss} + 10 \times \log_{10} N_{\lambda}. \quad (2)$$

Here, P_{laser} is the laser power in dBm, $S_{detector}$ is the receiver sensitivity, and N_{λ} is the number of wavelength channels in the link. Also, P_{phot_loss} is the total optical loss accumulated on the optical signal during its transmission, which includes propagation, crossing, and bending losses in the waveguides, through- and drop-port losses of MR modulators and filters, and modulating loss in modulator MRs due to imperfect modulation [86]. P_{laser} thus depends on the link bandwidth in terms of N_{λ}, and the total loss P_{phot_loss} encountered by each optical signal traversing the network. The P_{phot_loss} encountered along the network reduces the optical signal power. A signal can only be accurately recovered at the destination node if the received signal power is higher than $S_{detector}$. Ensuring this requires a high-enough P_{laser} to compensate for all optical losses.

To approximate data transmission for floating-point data transfers, [79] used lower P_{laser} for transmitting LSBs while keeping P_{laser} unchanged for MSBs. However, if the destination node is relatively farther along a waveguide from a source node, the signals would encounter high losses and the signal power at the detector MRs would be lower than $S_{detector}$, which would result in detecting logic '0' for all the approximated signals at the destination node (e.g., with OOK modulation). In the scenario where the destination is closer to the source, it may be possible to detect the approximated signals accurately, as long as the losses encountered are low enough that

the signal power at the detector MRs would be higher than $S_{detector}$, even with the reduced P_{laser} for the approximated bits. For each data transfer on a waveguide, if we are aware of the distance of the destination from the source, it is possible to calculate the losses encountered for the signals, which can allow us to determine whether the signals can be recovered accurately, or if they will be detected as '0's. In such a scenario, it is more efficient to simply truncate all the approximated bits (i.e., reduce P_{laser} to 0 for approximated signals) when the destination is farther along the waveguide and there is no likelihood of the signal being recovered accurately. Moreover, in the cases where the destination is closer to the source, we can transmit the approximated signals with a lower P_{laser}. This intelligent distance-aware transmission model for approximate data allows for some of the data to be detected accurately at the destination, while approximating other data depending on its content and distance to the destination.

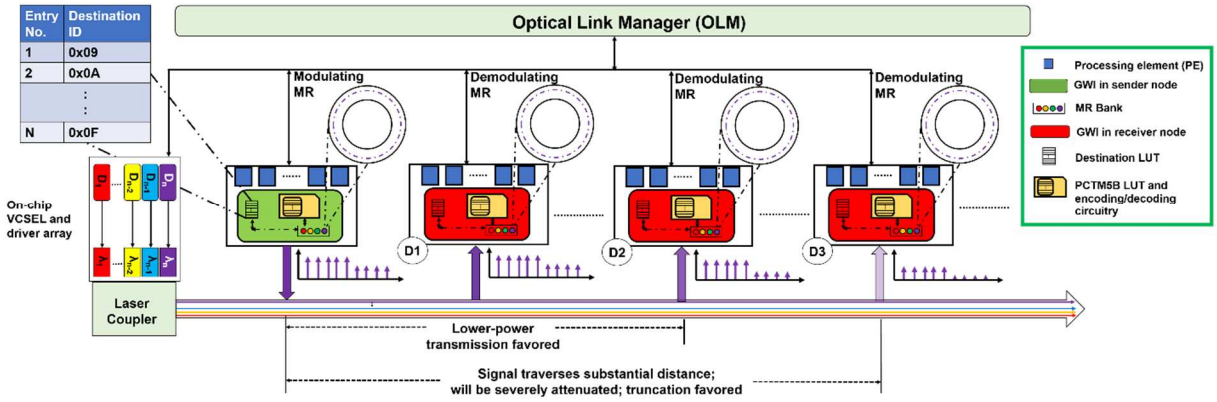


Figure 8. Overview of the proposed ARXON framework.

Figure 8 shows the operational details of the distance-aware transmission model in our framework, on a single-writer-multiple-reader (SWMR) waveguide that is part of a PNoC architecture. Note that while we illustrate our framework with an SWMR waveguide, our

framework is also applicable (with minimal changes) to multiple-writer-multiple-reader (MWMR) and multiple-writer-single-reader (MWSR) waveguides that are also used in many PNoCs. In Figure 8, only one sender node is active per data transmission phase and there is one receiver node (out of three in the figure) that is the destination for the transmission. In a pre-transmission phase (called receiver-selection phase), the sender notifies the receivers about the destination for the upcoming data transmission, and only the destination node will activate its MR banks, whereas the other nodes will power down their MR banks to save power in the transmission phase. As shown in Figure 8, if the destination node is close to the sender node, (e.g., D1), we can transmit the approximated bit signals with a lower P_{laser}. Otherwise, if the destination node is farther away from the sender node, (e.g., D3), we determine that it would not be possible to detect the approximated signals at that destination due to the greater losses the signals will encounter. Therefore, we dynamically turn off P_{laser}, essentially truncating the bits.

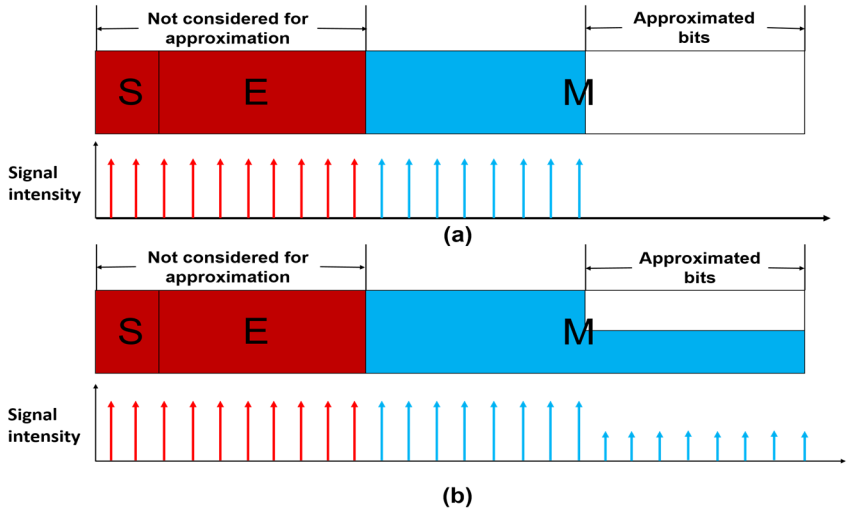


Figure 9. Floating-point data transmission on a photonic waveguide (a) truncation and (b) lower laser power.

We consider both integer and floating-point data for approximation. For floating-point data, we perform distance aware transmission of the LSBs of the data in such a way that it will not impact the overall output quality of the application. Figure 9 shows how transmission of data will conform to the distance aware transmission policy of our framework. In the case where substantial losses are expected to be encountered between a source and destination, we adopt the strategy shown in Figure 9(a), where the data is truncated, as the approximated bits would have been lost during transmission anyway. When the data can have enough power to be successfully received at the destination node, we adopt the strategy shown in Figure 9(b), where the data is transmitted at a lowered-laser power than its non-approximated counterparts. The power at which the bits can be transmitted, and the number of the approximated bits will depend on the application, as discussed in Section 2.5.

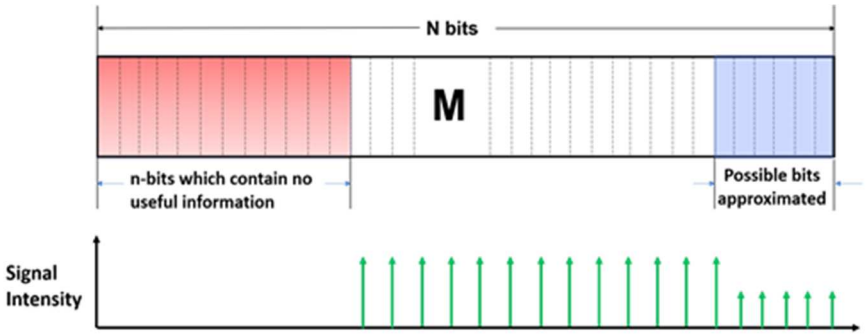


Figure 10. Approximated-integer-data transmission adopted.

For approximating integer variables, we take a different approach. Based on our analysis, indiscriminate approximation to integer data in an application can significantly reduce output quality. Therefore, we instead profile applications and log the range of values stored in each integer variable. If the range of values is smaller than the bit size allotted to the variable (e.g., the case

where a 32-bit integer variable only stores values up to 24 bits throughout the run of the application), we consider it a candidate for approximation. We can remove or truncate the MSBs that are unused in such variables that will otherwise take up modulation/demodulation and tuning energy necessary for transmission. We can also try and approximate the LSBs of the integer packets, and this approach can work in integer variables that store very large values where slight errors in the LSBs have minimal impact. But integer variables amenable to LSB approximation without significantly reducing output quality are rare. Nonetheless, for any such approximated LSB bits, the distance aware transmission model is applied as well. Figure 10 summarizes our approximation strategy for integer packets.

To implement these strategies, we require: *(i)* a laser-control mechanism that can dynamically control the laser power being injected into the on-chip waveguides, and *(ii)* a mechanism to annotate approximable variables in the application source code, for runtime adaptation of transfers involving these variables.

We utilize an on-chip laser array with vertical-cavity surface-emitting lasers (VCSELs) [89], which can be directly controlled using on-chip laser drivers. With the laser drivers, we can control the power fed into each individual VCSEL, thus controlling the power of the laser output for a particular wavelength corresponding to that VCSEL. The gateway interface (GWI) that connects the electrical layer of the chip to the PNoC (see Figure 8), communicates the desired P_{laser} power level (including 0 for truncation) to the drivers, via an optical link manager, similar in structure to the one proposed in [90].

Identification of candidate packets to be approximated is done at the processing-element level, via source-code annotations [72], to generate necessary flags for data that is approximable. The main considerations while generating the flags for the packet, in our framework, are to allow for

proper decoding of approximated or truncated packets at the destination. For this, two additional flags must be included in the packet header, at the processing-element level. The first (1 bit) flag indicates whether the approximable packet contains integer or float data and the second (1 bit) flag indicates whether the approximation is to be done for LSBs or MSBs. The number of bits that can be safely approximated or truncated are determined offline for each application and stored in lookup tables (LUTs) at the network interface (NI) which connects processing elements to routers that are in turn connected to GWIs. The number of bits approximated/truncated in a packet is also passed as part of the header flit of the packet to the GWI. This information can be used to gate (i.e., prevent) those bits from being passed into encoding/decoding circuitry. Note that as the number of bits truncated/approximated is necessary information for decoding, we must convey this information to the destination GWI as well. For this, we use six bits in the packet header. These six bits represent the number of approximated/truncated bits in the range from 0 to 32 bits, which is the range of approximation/truncation in our chapter.

Usually the header flit contains the routing information, which can just be the destination address. We consider a flit size of 64 bits, i.e., 64 bits are transmitted per transmission cycle. The number of used bits in the header flit do not exceed 16 bits (for the destination and source addresses), thus making it possible to incorporate the 8 necessary bits containing the two bits for the necessary flags and six bits for the approximation/truncation size information without causing any additional latency overheads. Once the header flit is received at the destination GWI, the flags and the approximated/truncated bits information are used to select the appropriate LSB/MSB to not be considered for decoding. The packet ID from the flit can then be used to track the remaining flits in the packet and treat them accordingly, if they were approximated/truncated.

Once the approximable bits have been identified, we must determine whether the approximation during their transfer is to be accomplished via reduced power transmission or truncation. This requires a LUT at each GWI (see Figure 8) with the IDs of all the destination GWIs. The table at a source is populated with the destination IDs to which the loss values are sufficiently large enough to warrant truncation. The values can be easily calculated post-fabrication at design time, as the location of destination nodes as well as the cumulative loss to their GWI from the source does not significantly change at runtime. Once the decision to truncate or transmit at a lower laser power is made, depending on the destination node, the required power levels for the wavelengths are communicated to the VCSEL drivers via the optical-link manager. We discuss the overheads of the tables and the application specific P_{laser} for the approximated signals in Section 2.5.

2.4.2. RELAXED CROSSTALK MITIGATION STRATEGY

Due to the challenges with signal crosstalk outlined in Section 2.5, PNoCs must utilize one or more crosstalk mitigation strategies to reduce and achieve high SNR. We consider a state-of-the-art crosstalk mitigation strategy from [53] that can be applied at the link level in PNoCs. Analyses from [53] showed that a '1' carried by the wavelengths in the DWDM wavelength group adjacent to the resonant wavelength of an MR causes higher crosstalk in that MR. An encoding strategy was proposed to reduce inter-channel crosstalk noise by replacing instances of '1' values in adjacent wavelengths with '0' values, which helped reduce the optical signal-strength of immediate non-resonant wavelengths and improve SNR. Two encoding techniques were proposed that encoded nibbles (4-bits) of data. The PCTM5B technique encoded the nibble to 5-bit data, while the PCTM6B technique encoded the nibble to 6-bit data. Table 1 shows the code words used in these encoding techniques. Note that to implement PCTM5B on a photonic link with 64-bit

word parallel transfers, 16 additional bits are required, which increases the number of MRs by 25%. Similarly, for PCTM6B, 32 additional bits are required for a 64-bit data word, and this increases the number of MRs by 50%. We assume that the lower-overhead PCTM5B technique is integrated into PNoCs by default, to meet BER goals.

In order to mitigate crosstalk, we assume the baseline configuration of the PNoC to implement PCTM5B. This means the encoder/decoder circuitry and the LUT, containing the data word-code word pairs, are incorporated into the GWI. Using these additions, the incoming packets from the processing elements can be encoded before they are transmitted to their destination, and at the destination, the packets are decoded using the LUTs. In our framework, applying crosstalk mitigation via PCTM5B technique to the truncated or approximated bits is an unnecessary overhead as it does not provide any benefits towards BER. By relaxing crosstalk mitigation for the truncated or approximated bits, it is possible to reduce the energy costs of the mitigation strategy. We do this by leveraging the approximation information gathered using our offline analysis of applications, where we consider that some LSB/MSB of the data can be approximated/truncated. During the encoding process, we do not consider these bits by gating their access to the encoder. Similarly, at the destination, when an approximated/truncated packet is received, the information from our LUTs are used to gate the approximated/truncated bits from being passed into the decoder circuitry.

Table 1 Data word to code word conversion [53].

Code Words for PCTM5B Technique			
Data Word	Code Word	Data Word	Code Word
0000	00000	1000	01000
0001	00001	1001	01001
0010	00010	1010	01010
0011	10101	1011	10100
0100	00100	1100	01100

0101	00101	1101	10010
0110	00110	1110	10001
0111	10110	1111	10000
Code Words for PCTM6B Technique			
Data Word	Code Word	Data Word	Code Word
0000	000000	1000	001000
0001	000001	1001	001001
0010	000010	1010	001010
0011	100000	1011	010100
0100	000100	1100	100010
0101	000101	1101	010010
0110	010101	1110	010001
0111	100001	1111	010000

2.4.3. RELAXED MR TUNING STRATEGY

Thermal or electrical tuning of MRs in a PNoC is crucial for ensuring reliable communication, by counter-acting the effects of PV and TV. We assume the use of thermo-optic tuning in PNoCs, due to its better range of $\Delta\lambda_R$ correction. Electro-optic tuning can provide a tuning range of at most 1.5 nm [91]. In contrast, thermo-optic tuning can provide a tuning range of about 6.6 nm corresponding to the temperature range of up to 60K [92] at 0.11 nm/K sensitivity [93]. This comes at the price of higher energy consumption (~mW/nm) and slower operation (in units of μs). In our framework, we aim to reduce the overhead of tuning the MRs associated with truncated bits. We do not consider approximated bits for relaxed MR tuning, as the added noise this approach generates, due to thermal drift of λ_R, may render the approximated bits unreadable at the destination

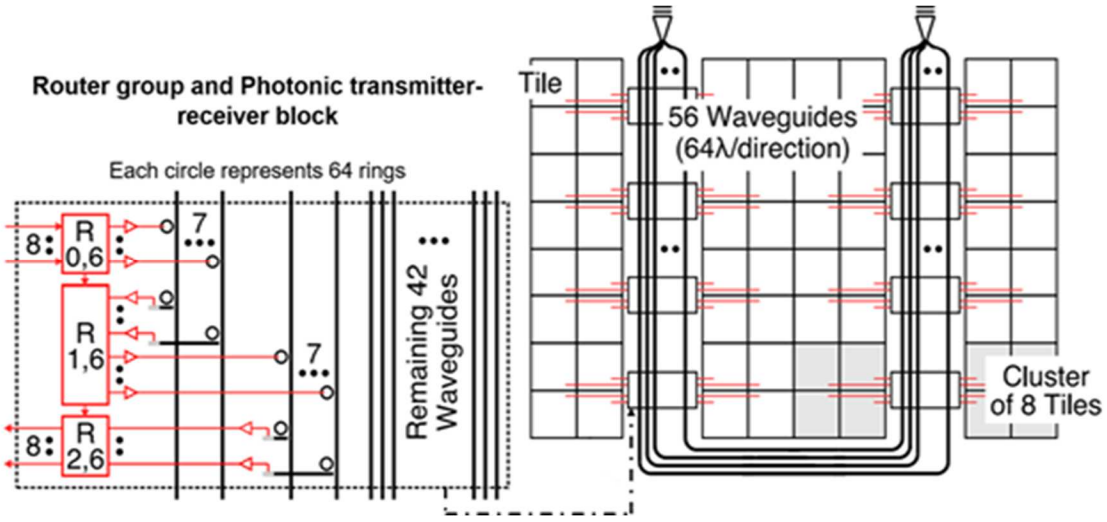
GWI. We do however relax the requirement for tuning MRs associated with the truncated bits, by temporarily turning off the tuning mechanism for those MRs.

2.4.4 INTEGRATED MULTI_LEVEL SIGNALLING

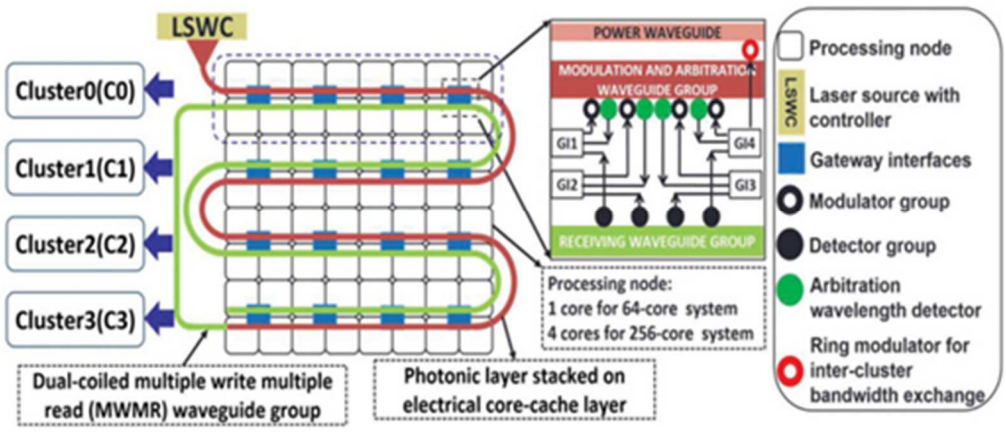
The discussion in the previous sub-sections assumes the use of conventional on-off keying (OOK) signal modulation, where each photonic signal can have one of two power levels: high or on (when transmitting a '1'), and low or off (when transmitting a '0'). In contrast, multilevel signaling is a signal-modulation scheme where more than two levels of voltage can be used to modulate multiple bits of data simultaneously in each optical signal. The obvious benefit with such multilevel signaling is an increase in the bandwidth. Leveraging this technique in the photonic domain has, however, traditionally been a cumbersome process with high overheads, e.g., when using the signal superposition techniques from [94]. But with advances such as the introduction of Optical Digital to Analog Converter (ODAC) circuits [95] that are much more compact and faster than Mach-Zehnder Interferometers (MZIs) used in techniques involving superimposition [94], multilevel signaling has been shown to be more energy efficient than OOK [88], making it a promising candidate for more aggressive energy savings in silicon photonic networks.

Four-level pulse amplitude modulation (PAM4) is a multilevel signal modulation scheme where two extra levels of voltage (or optical signal power in case of optical modulation) are added in between the '0' and '1' levels of OOK. This allows PAM4 to transmit two bits per modulation as opposed to one bit per modulation in OOK. This in turn increases the bandwidth when compared to OOK. We are interested in evaluating the impact of using PAM4 in PNoCs and how its use will impact the effectiveness of our approximation strategies in *ARXON*. While PAM4 promises better energy-efficiency than OOK, it is prone to higher BER due to having multiple levels of the signal close to each other in the spectrum. Thus, we cannot reduce the laser power level of the LSB bits

to the level used in OOK, as it would significantly reduce the likelihood of accurate data recovery even when destination nodes are relatively close to the source. Thus, when PAM4 is used, we need to increase the laser power compared to OOK. We used an empirically determined value of approximately 1.5× the laser power that was used for OOK, to prevent the degradation of approximated signals transmitted with PAM4. This may seem like a backward step in conserving energy, but the reduced-operational cost per modulation and the reduced-wavelength count for achieving the same bandwidth as OOK can reduce the overall laser power consumption. Also, while it is possible to add more signaling levels (e.g., to use a PAM8 modulation scheme), as the number of amplitude levels increases, the optical signal becomes extremely susceptible to noise and causes increase in BER [96]. To ensure reliable communication when using PAM8, the bandwidth and speed of operation must be sacrificed [97]. Considering these constraints, we limit the extent of multilevel signaling integration in our framework to PAM4.The experimental results in the next section quantify the impact and trade-off when using PAM4 signaling in our framework.



(a)



(b)

Figure 11. PNoC architectures considered for analyses. (a) Eight-ary three-stage Clos architecture with 64 cores [98]; and, (b) Schematic overview of SwiftNoC architecture [99].

2.5. ARXON EVALUATION AND SIMULATION RESULTS

2.5.1. EVALUATION SETUP

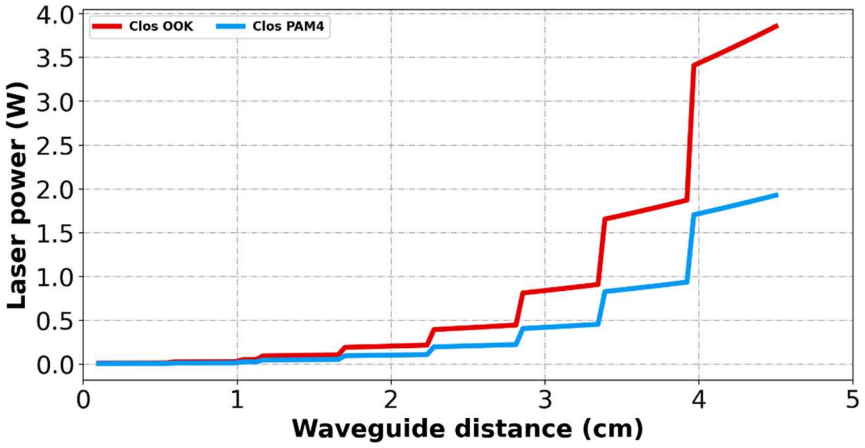
To evaluate our proposed ARXON framework, we implement it in Clos [98] and SwiftNoC PNoC architectures [99] for a 64 core processor, with baseline OOK signaling, PCTM5B crosstalk mitigation, and thermo-optic tuning in MRs.

The Clos PNoC, shown in Figure 11(a), has an 8-ary 3-stage topology for a 64-core system with eight clusters and eight cores per cluster. It utilizes an optical crossbar topology with point-to-point photonic links utilizing SWMR waveguides for inter-cluster communication. Each cluster has two concentrators and a group of four cores is connected to each concentrator, where concentrators communicate with each other using an electrical router.

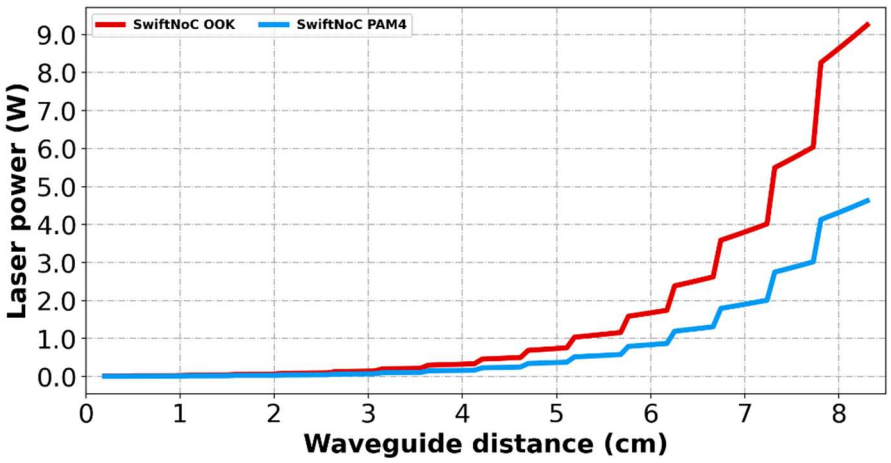
For the SwiftNoC PNoC, as shown in Figure 11(b), we have again considered a 64-core system. Each node here has four cores and communication within the node happens through a 5×5 router, with the fifth port of the router connected to a GWI, which facilitates transfers between the CMOS-

electrical layer and the photonic layer. Each GWI connects four nodes. The architecture utilizes eight waveguide groups with four MWMR waveguides per group in a crossbar topology. In order to support the MWMR communication, SwiftNoC utilizes a concurrent token stream arbitration that provides multiple simultaneous tokens and increases channel utilization.

The Clos PNoC has a waveguide length of 4.5cm and the SwiftNoC PNoC has a waveguide length of 8.3cm over the considered 400mm2 chip. In both PNoCs, the first MR is encountered at ~1cm and the last MR is encountered at ~3.8cm for Clos PNoC and ~7.8cm for SwiftNoC. These distances have a key relationship to the laser power consumption, which we try to capture using the power model in (2). This relationship is visualized in Figure 12, where the sudden jumps in power indicate a new GWI with the optical devices being encountered along the waveguide.



(a)



(b)

Figure 12/ Laser power consumption behavior over the length of the waveguide in (a) Clos PNoC and (b) SwiftNoC.

Table 2 64-core architecture configuration.

Simulated component	Specification
No. of cores, processor type	64, x86
DRAM	8GB, DDR3
Memory controllers	8
L1 I/D cache, line size	128KB each, direct mapped, 64B
L2 cache, line size, coherence	2MB, 2-way set associative, 64B, MESI

The considered PNoC architectures were modeled and simulated using an in-house SystemC based cycle-accurate simulator. A combination of gem5 full-system simulator [83] and Intel PIN toolkit [84] was utilized to generate traces for the entire application that were replayed on the PNoC simulators to determine energy savings in the PNoC. The PIN tool was used to obtain the addresses of the variables we deemed suitable for approximation from our analysis of applications and then to track accesses to them. Using this information in gem5 simulation, we track the relevant data flow at various levels of the simulated system (processor level, memory controller level, DRAM level and cache level). The information generated while the simulation is running was

consolidated and custom python scripts were created to extract the necessary information about the data packets (e.g., timestamp at origin, their source, destination, data values, and control values from the packet header) and to generate the traces necessary for our cycle accurate simulator to simulate the applications on these PNoC architectures. Then, details of the approximate data communication (i.e., whether a packet was truncated or transmitted at lower power) were used to modify data in a subsequent gem5 simulation, to estimate the impact of the approximation on output quality for the application being considered. Table 2 shows gem5 architectural parameters considered in our experiments. We have based our simulations on x86 cores, but these simulations and our approach is applicable to systems having other types of cores as well, for e.g., ARM cores. Twelve applications from the ACCEPT and tinyDNN benchmark suites were used in our evaluations. The performance was evaluated at the 22 nm CMOS node for 400 mm2 chips, with cores and routers operating at 5 GHz clock frequency. DSENT [100] was used to calculate the energy consumption of routers and the GWI at each node. Each GWI holds two LUTs for our framework; these are: one which holds the information regarding which destination addresses are preferred for truncation, and another for PCTMB5 encoding scheme. The size of both the LUTs at GWI level is fixed and is application independent. The PCTM5B LUT takes up only 144 bits for storing encoding decoding information at each GWI. The destination ID LUT can take up a maximum of 32 bits at each GWI for Clos PNoC and 64 bits for SwiftNoC variants.

The table containing information regarding number of bits to be approximated/truncated for integer/float approximable packets is stored at the network interface (NI) of each processor. The maximum number of bits required in these LUTs for the worst case (application with the highest number of approximable variables) is a few hundred bits for the applications we considered. CACTI v6.5 [101] and scaling equations from [102] were used to evaluate the power, area, and

delay for the lookup tables in NIs and GWIs. These values were found to be 0.236 mm2 for the area consumption for all the tables, with a total power overhead, for reading from and writing into the tables, of 0.135 mW for Clos and 0.472 mm2 and 0.27 mW respectively, for SwiftNoC. The combined power and area consumption of associated circuitry necessary for accessing information in the LUTs, calculated using gate-level analysis, is 0.0274 mm2 and 4.224 mW for Clos, and 0.0548 mm2 and 8.448 mW for SwiftNoC. LUTs in both Clos and SwiftNoC have the same number of entries as both architectures have the same number of processing elements. The encoding/decoding scheme is the same and the approximations done depend on the output error quality of the application and not the architecture, while SwiftNoC has double the number of GWIs. The access time for scratchpad RAMs designed with 22 nm technology node was under 1 cycle from synthesis estimates.

VCSEL control in ARXON was modeled after the optical link manager in [90], where the channel management for their PNoC design was described. However, since we are considering PNoCs from prior works with their own channel management systems in place for our analysis, we only adapt the approach for VCSEL control from [90]. The VCSEL control described in [90] uses a combination of MRs and PDs, but we only require the MR based switching mechanism for the VCSEL output. From the data available in [90] we calculated the area overhead necessary for implementing the VCSEL control, which was 0.093 mm2 for OOK variants and 0.047 mm2 for PAM4 variants of both the architectures.

Clos and SwiftNoC PNoC architectures with PCTM5B are used as baselines for our analyses in this chapter. We have also considered a two-cycle overhead for PCTM5B encoding and decoding of the signals, as calculated in [53]. We considered N_λ = 64 for OOK, which would enable 64-bit transmission across a waveguide per cycle. For PAM4, we only need to consider N_λ

= 32 to achieve the same bandwidth as with OOK modulation. Table 3 shows the energy values for losses and power dissipation in different photonic devices. We use a "standard" set of values for these parameters from existing prototyping efforts, and a more "aggressive" set of values as per future projections from various research efforts. Our approach sacrifices reliability of approximated bits in floating point data and selected integer variable data, for EPB and laser power savings, as discussed in Sections 2.4.2, 2.4.3, and 2.4.4.

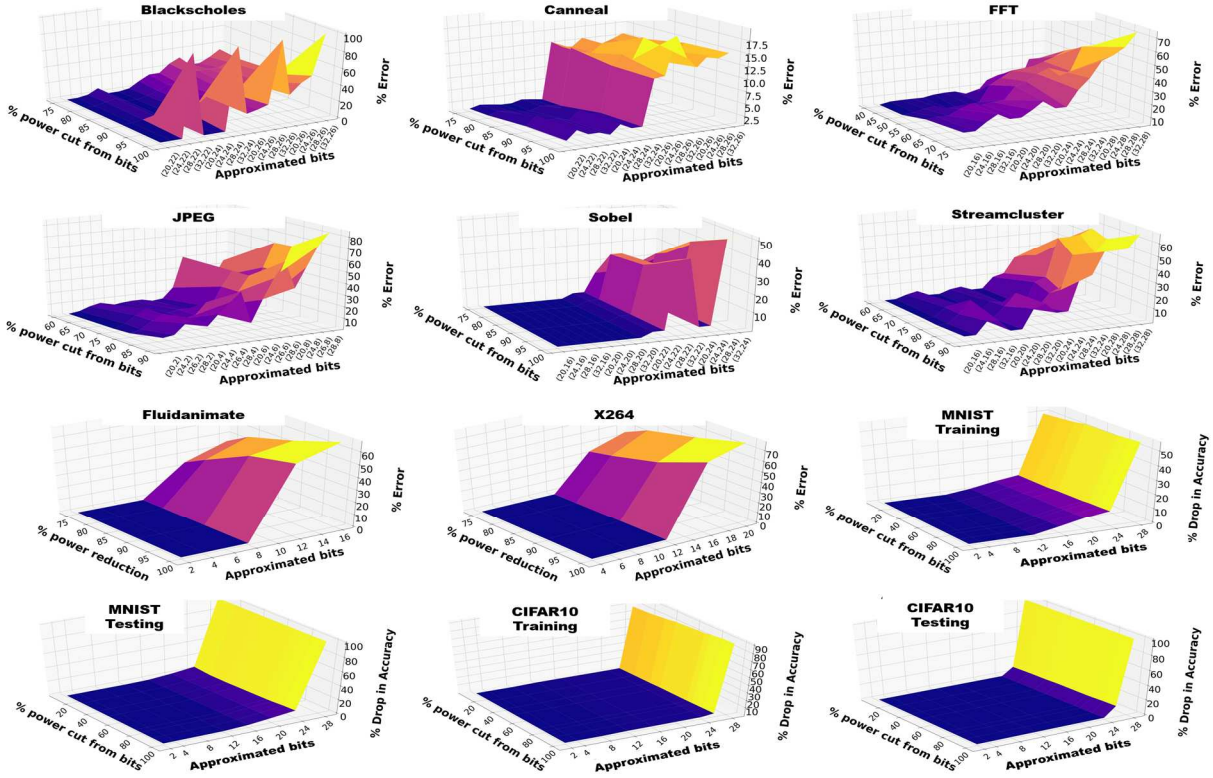


Figure 13. Percentage error (PE)/Drop in accuracy in application output as a function of the number of approximated bit signals (y axis) and reduction in laser power (x axis) for the approximated signals, for blackscholes, canneal, fft, jpeg, sobel, streamcluster, fluidanimate, and X264 benchmarks with large input workloads and MNIST (training and testing) and CIFAR10 (training and testing) models.

We use the standard values for most of our simulations and use the aggressive values in Section 2.5.4. These values are used to calculate laser power from (2) and total power after considering tuning and lookup-table overheads. We consider a laser efficiency of 10% for our on-chip VCSELs, which is midway, the initial and worst-case efficiencies mentioned in [89]. We additionally consider a PAM4-induced-signaling loss of 5.8 dB in Pphot_loss for laser power calculations for PAM4 [88]. To compensate for the increased sensitivity of PAM4 to bit errors, we also consider laser-power levels that are 1.5× than those used for OOK signaling. For ensuring reliable communication, we have considered a BER of 10-9 in our designs. Lastly, we calculated application output error for the non-machine learning applications due to our approximation approach as:

$$Percentage\ (Output)\ Error = \frac{|approximatedvalue - exactvalue|}{exact\ value} \times 100. \qquad (3)$$

Table 3 Loss and power parameters

Parameters considered	Standard values	Aggressive values
Receiver sensitivity	-20 dBm	-23.4 dBm
MR through loss	0.02 dB	0.02 dB
MR drop loss	0.7 dB	0.5 dB
Propagation loss	1 dB/cm	0.25 dB/cm
Bending loss	0.01 dB/90^o	0.005 dB/90^o
Thermo-optic tuning	6.67 mW/nm	240 µW/nm

The "exact value" refers to the original output values, which can be a set of values presented in the output files, like in the case of Blackscholes, or it can be pixel values of output images/frames, like in the case of JPEG, Sobel or X264. The "approximated value" refers to the value of these outputs once the approximation approach is applied to the applications. For our analysis, we assume an error threshold of 10% output error, which was seen experimentally to be the limit at which the errors became apparent in the outputs of the majority of the applications [80]. For example, artifacts become noticeable in JPEG output as we cross the 10% error threshold.

Thus, we want to ensure that none of the approximation strategies degrade output quality by more than 10%. For our machine-learning applications, we have considered the drop in accuracy to measure the impact of our framework and we have set the threshold as 10% drop in the accuracy.

2.5.2. IMPACT OF ARXON ON APPLICATIONS CONSIDERED

Our first set of experiments involve analyzing the sensitivity of an application to varying degrees of approximation of their floating-point data. We are interested in studying the impact on output error from approximating a number of bits in the packets carrying data deemed approximable. Additionally, we are also interested in studying the impact on output error of varying levels of lowered laser power for those approximated bits.

Figure 13 shows the results of our comprehensive study for the applications we considered (as depicted earlier in Figure 7). The z-axis shows the percentage error (PE) in application output, or drop in accuracy for ML applications, as a function of the reduction in Plaser level for the photonic signals that carry the approximated bits (x-axis; varying from 0% to 100%, where 100% refers to truncation), and the number of bits that were considered for approximation (y-axis; with the number of approximated float and integer bits given in [float, integer] format). The subset of combination of these values were selected for enabling viable trade-offs between output quality and power consumption. It should be noted that not all applications consider both floating-point and integer data for approximation. For example, Fluidanimate only considers integers for approximation while the ML applications (CIFAR10 and MNIST) only considers floating-point data. This selection of datatypes to be approximated was made after profiling the application and determining the datatypes that do not have adequate impact on the traffic (e.g., floating-point data in the case of Fluidanimate and X264) or the functionality of the application (integers in the case

of the ML applications considered). This is a more comprehensive version of the experiments in our earlier work, presented in [80]. In those experiments in [80] we had determined how much floating-point approximation can be tolerated by the applications from ACCEPT benchmark. Here we not only consider a larger number and variety of applications, but also use more comprehensively determined thresholds than in LORAX to explore how approximating the integer bits along with the float bits affects the output quality. It is clear from our analyses that not all applications can tolerate the same level of approximation. From the PE values, we can observe that FFT with a large volume of floating-point data traffic (see Figure 7) reaches the error threshold of 10% rather quickly as the number of approximated bits increases and laser power-levels reduce, whereas Canneal with a lower floating-point traffic-volume observed seems to have very low PE values across the various experiments. The edge detection algorithm Sobel performs well in approximated conditions, possibly owing to the lowered data accuracy requirements to construct the output. Streamcluster involves an approximation strategy for data streams and is also observed to be quite resilient to greater levels of approximation. Blackscholes, which performs market options calculations is particularly sensitive to the approximated number of bits and the laser-power levels. JPEG performs image compression, and the output image quality is also more sensitive to approximation. Fluidanimate generates a video of flowing liquid depending on the input data provided. X264 is a video codec, which generates compressed video from the input, which is raw video data. Fluidanimate and X264 applications were subjected to only integer MSB approximation, and threshold is quickly breached after the amount of MSBs approximated start taking up bits which contain values, the quick rise in error can be explained by the fact that we are approximating MSBs which would cause very large shift in values. Moreover, we considered implementations of deep convolutional neural networks for classification of CIFAR-10 and

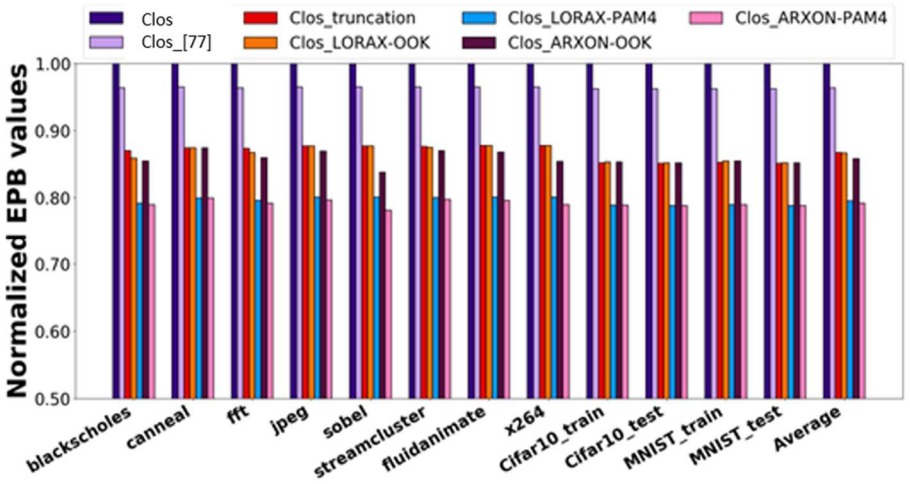
MNIST datasets, from tinyDNN. The machine learning applications used single precision floats, and we were able to approximate till the point where we encroached on the exponent, but the decay of output accuracy ramped up very quickly once we tried to approximate any further. From Figure 8 we can see that there is a sharp increase in percentage output error (PE), as we approximated beyond a certain number of bits, in the case of many of the applications considered, e.g., the applications in the bottom two rows. The erratic jumps in error rate for the six applications in the top two rows of Figure 8 are because we are considering discrete combinations of approximated bits for floating point and integer variables, along the 'Approximated bits' axis.

Table 4 summarizes the best combination of approximable bits and the laser-power-transmission levels for these bits and for each application while ensuring that the application output error does not exceed 10% for our proposed framework (ARXON). Table 4 also shows the number of bits that can be truncated, selected to meet the <10% PE constraint. For the approach in [79], we perform approximations on 16 LSBs transmitted at 20% laser power (advocated as an optimal choice in that work), which also satisfies the <10% PE constraint.

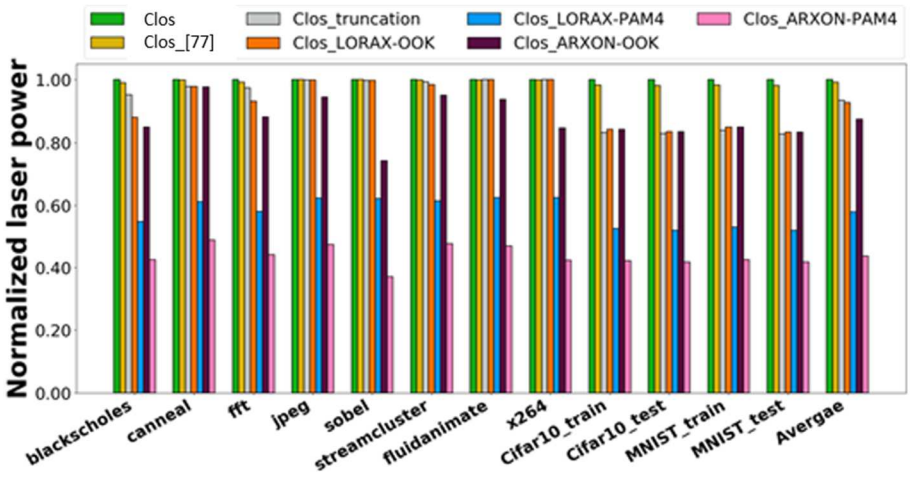
Table 4 Number of bits considered for approximation and laser-transmission-power level for the corresponding signals across benchmarks and frameworks considered.

Application Name	Truncation	[79]	*LORAX* [80]		*ARXON* (proposed)		
	Truncated Bits (float)		Approximated Bits (float)	% Power reduction	Approximated bits in floating-point packets	Approximated bits in floating-point packets	% Power reduction
Blackscholes	12	16 bits approximated, with 20% power reduction	32	90	32	32	90
Canneal	32		32	100	32	32	100
FFT	8		32	50	32	32	50
JPEG	20		24	80	22	22	80
Sobel	32		32	100	32	32	100
Streamcluster	12		28	80	28	28	80
Fluidanimate	-		-	-	-	-	100
X264	-		-	-	-	-	100
MNIST_train	24		24	100	24	24	100
MNIST_test	24		24	100	24	24	100
CIFAR10_train	24		24	100	24	24	100
CIFAR10_test	24		24	100	24	24	100

Figure 14 shows the EPB and laser power comparison results for the various frameworks in the Clos PNoC architecture These analyses consider the benefits from distance-aware transmission and the relaxed encoding technique for approximated packets for ARXON. Figure 14(a) shows that using ARXON-OOK results in lower EPB than the previous approaches, including our previous framework LORAX-OOK. The better EPB for LORAX and ARXON can be attributed the fact that they avoid wasteful transmission at lower laser power when it is unlikely that the destination can recover the transmitted data due to high optical losses. Also, [79] has noticeably higher EPB values for which we are not considering the benefits of relaxed encoding and distance-aware transmission for the framework to be consistent with the framework presented in that chapter.



(a)



(b)

Figure 14. (a) Energy-per-bit (EPB) and (b) laser power comparison across different frameworks for Clos PNoC architecture.

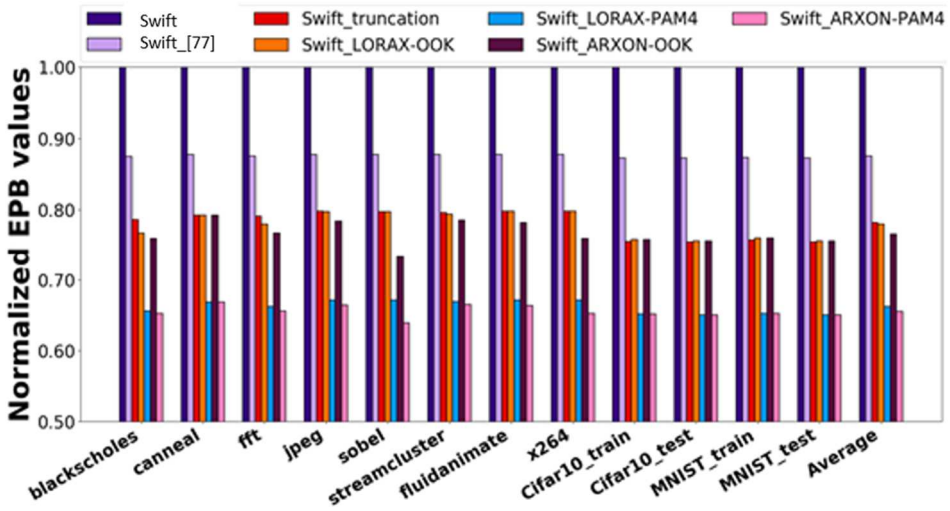
The ARXON-OOK framework improves upon LORAX-OOK, [79] and truncation, by adaptively switching between truncation and an application-specific laser-power-intensity level for approximated bits of both floating-point and integer packets. The ARXON-PAM4 variant of our framework achieves the largest reduction in EPB, even though it uses 1.5× higher laser-power levels for the approximated bits. The use of fewer wavelengths in PAM4 allows for more energy savings, despite greater losses and the use of more laser power per wavelength than OOK variant.

On average, ARXON-PAM4 shows 21%, 17.2%, 9.7%, 9.2%, and 1.2% lower EPB compared to the baseline Clos, [79], truncation, LORAX-OOK, and LORAX-PAM4 approaches, respectively. ARXON-OOK exhibits lower EPB on average while having a 6% higher EPB than the LORAX-PAM4 approach. In the best case scenarios for the Blackscholes and Sobel applications, ARXON-PAM4 has 21.2% and 23.5% lower EPB than the Clos baseline; and 17.4% and 15.6% lower EPB than [79]; 9.8% and 11.5% lower EPB when compared to truncation; 8.6%

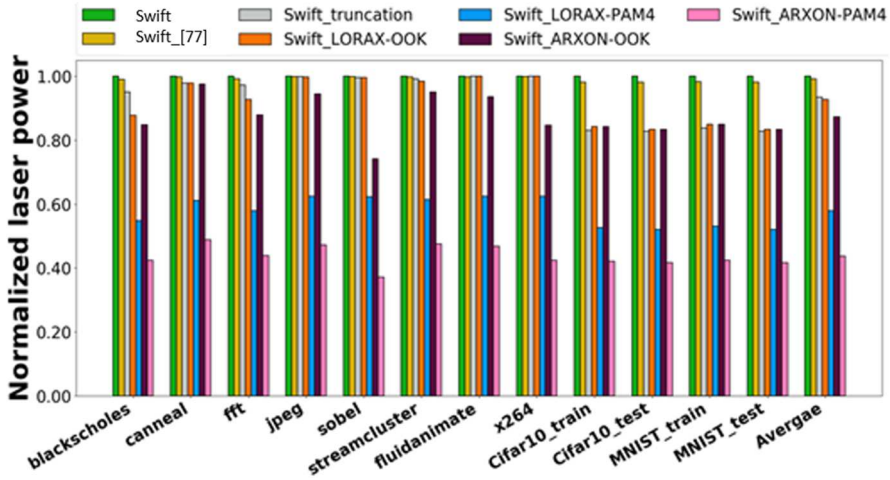
and 10.25% lower EPB than LORAX-OOK, and 1.24% and 2.5% lower EPB than LORAX-PAM4 for these two applications.

Figure 14(b) shows the laser power reduction. On average, ARXON-PAM4 uses 50.45%, 49.5%, 43.2%, 42.5%, and 7.7% lower laser power compared to the baseline Clos, [79], truncation, LORAX-OOK, and LORAX-PAM4, respectively. ARXON-OOK exhibits lower average laser-power consumption on average while exhibiting 28% higher laser power consumption than LORAX-PAM4. For the best case Blackscholes and Sobel applications, laser power for ARXON-PAM4 is 51.7% and 59.2% lower than the Clos baseline and 50.8% and 57.9% lower than [79], while against truncation it is 51% and 58.5% lower, against LORAX-OOK we see 38% and 57% lowered laser-power utilization and against LORAX-PAM4 we have 6.5% and 20% lower laser-power utilization.

Figure 15 shows the same analyses but done for the frameworks implemented on the SwiftNoC architecture. The larger data rate and the larger number of GWIs in the architecture has impacted the packets and their distance aware transmission profile, creating more avenues to truncate the packets, yielding better EPB results in this architecture. The general trend in EPB and laser-power savings is similar to that for the Clos architecture, with Blackscholes and Sobel applications again exhibiting the best EPB and laser-power saving values. From Figure 15(a), ARXON-PAM4 exhibits 36%, 23.8%, 13.5%, 12.9%, and 1.8% lower EPB on average than baseline SwiftNoC, [79], truncation, LORAX-OOK, and LORAX-PAM4, respectively.



(a)



(b)

Figure 15. (a) Energy-per-bit (EPB) and (b) laser power comparison across different frameworks for SwiftNoC architecture.

The results for SwiftNoC show the same trend as the Clos architecture for normalized laser power (Figure 15(b)), albeit with lower laser power across applications with average laser power consumption for ARXON-PAM4 at 57.2%, 56.4%, 50.8%, 49.3%, and 15.7% better than baseline SwiftNoC, [79], truncation, LORAX-OOK, and LORAX-PAM4, respectively.

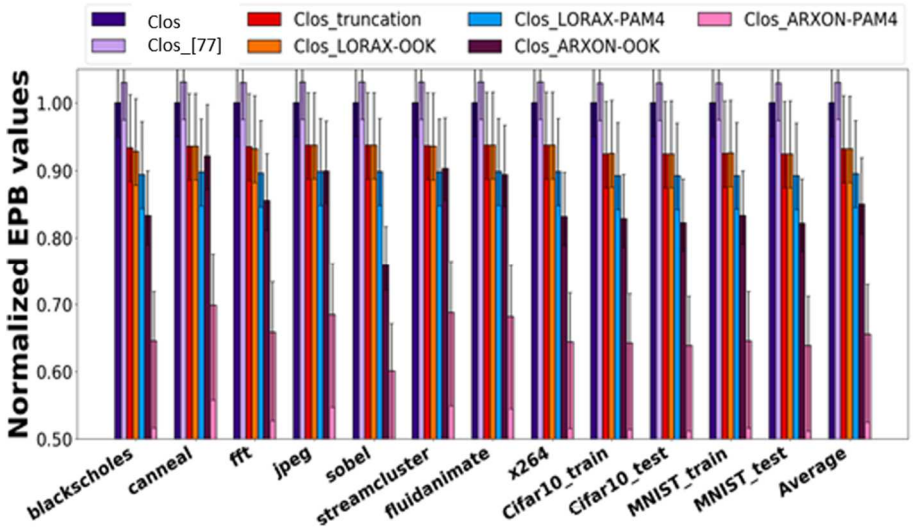
These results highlight the promise of our ARXON framework, as it improves upon the ability LORAX exhibited to trade-off output correctness with energy-efficiency and laser-power savings in PNoC architectures executing selected applications.

2.5.3. MR TUNING RELAXATION-BASED ANALYSES

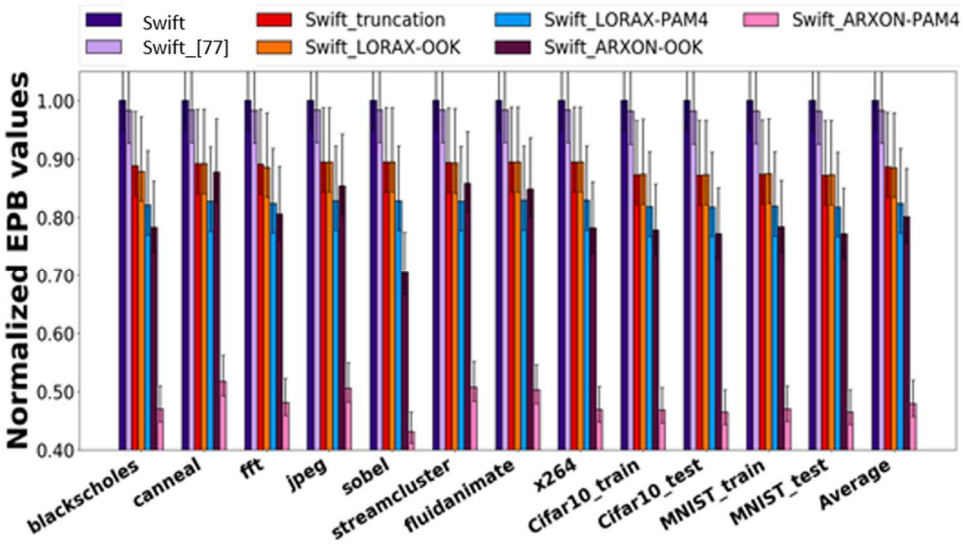
In addition to distance-aware transmission for float and integer packets and relaxing crosstalk-mitigation encoding techniques, we also consider the potential for relaxed thermo-optic tuning for truncated bits. We have considered thermal MR tuning in our chapter for its larger range of operation over other tuning methods such as electro-optic tuning. However, thermal tuning strategies are much slower in operation when compared to electro-optic tuning (microseconds for operation as opposed to nano to picoseconds for electro-optic tuning). But, this overhead cannot be avoided, as using just the electro-optic tuning method will not offer sufficient coverage for the thermal and process variations encountered by MRs, the effect of which must be mitigated for correct operation.

But with the increasing maturity of silicon photonics, we envision faster thermo-optic tuning strategies or a combination of different tuning strategies to reduce this tuning latency. Therefore, in this section we explore the potential of energy savings due to relaxed MR tuning, i.e., by turning off the tuning mechanism for MRs associated with truncated bits. For this experiment, we utilize thermal and process-variation information. For thermal variations (TV), we have referred to the study conducted in [103] and have adopted the worst-case TV induced shift to be 6.5 nm. For analysis of process variations (PV), we utilized the PV analysis method as described in [103], where PV is considered as a Gaussian random distribution. As the granularity of the method is at 30 nm, we have opted for analyzing PV at the GWI level rather than for individual MR devices.

We have generated PV maps for the architectures using the method from [104] and have selected locations corresponding to the GWIs in the layouts. We took the average of device variations (i.e., width and thickness) in that location. This was repeated over 100 different PV maps.



(a)



(b)

Figure 16. EPB values for ARXON implemented on (a) Clos and (b) SwiftNoC while considering thermal-tuning relaxation.

Utilizing the PV and TV information obtained, we implement the tuning-relaxation approach, where we turn off thermo-optic tuning for all truncated bits. In order to implement the control

necessary for relaxing the tuning, which in our case is to turn off tuning mechanisms to the MRs of truncated bits, we use a gating mechanism similar to the one utilized for the encoding strategy, as mentioned in Section 2.4.2. With this mechanism, we can power gate the tuning circuits to the MR, as per the information from LUTs, again similar to the description in Section 2.4.2. From our analysis, this had a substantial impact on the EPB values of our ARXON framework, as shown in Figure 16. Our observations in Figure 16(a) for Clos PNoC and Figure 16(b) for SwiftNoC, show that the ARXON variants have substantial savings over the other frameworks considered, a trend maintained even while using the aggressive values as it was with standard values. This is because the tuning based approach is again dependent on the traffic profile of the applications, with higher truncated packets meaning better savings. So, we see Blackscholes and Sobel as the best performing applications again. We do not consider laser-power savings in this scenario, as the tuning relaxation approach does not impact the laser power. On average, ARXON-PAM4 has 38.1%, 36.1%, 26.8%, 26.4%, and 19.2% better EPB values than baseline Clos, [79], truncation, LORAX-OOK and LORAX-PAM4. When implemented in SwiftNoC, ARXON-PAM4 exhibits 48.6%, 39.3%, 29%, 28.5%, and 16.9% better EPB than baseline, [79], truncation, LORAX-OOK and LORAX-PAM4, respectively. This only adds to the significant reduction in the overall laser power consumption achieved by ARXON, showing how our framework achieves better laser power and EPB values for all the applications considered in our analyses.

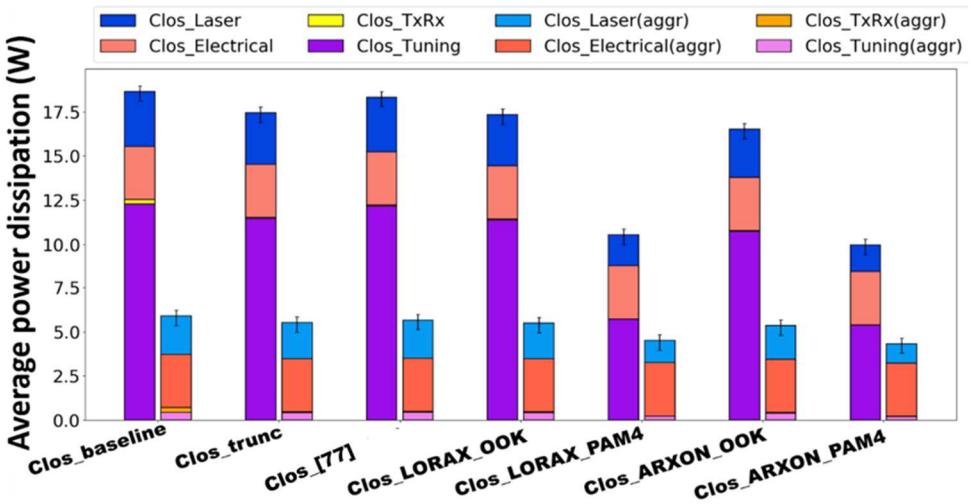
2.5.4. POWER DISSIPATION BREAKDOWN

We performed an experiment to determine how much more power can be saved as silicon photonics technology matures and devices with improved characteristics become available. For this, we contrast the power dissipation with our framework on the Clos and SwiftNoC architectures, for the standard and aggressive values of parameters in Table 3. As the EPB and

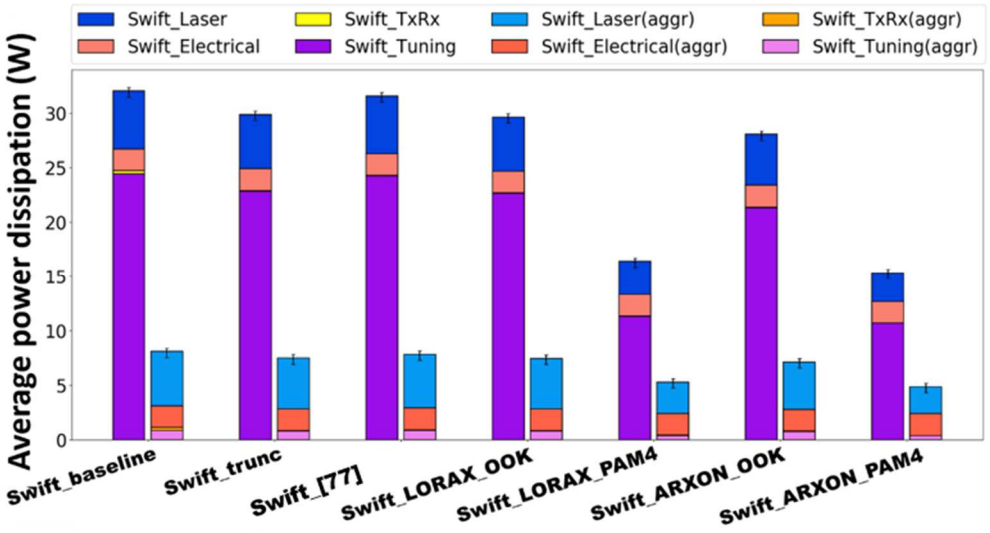
laser power once normalized follows the same trends, we decided to use a detailed power-dissipation breakdown to show how much ARXON improves the power consumption in PNoC and in which areas.

Figure 17 shows the detailed power breakdown for the frameworks, averaged across the applications. From the figures we can clearly observe how ARXON impacts both laser power and tuning-power dissipation, having the lowest power dissipation in both those categories and in total, be it while considering standard loss and power utilization values or while considering aggressive values.

Lastly, Table 5 shows the power consumption at the 64-core chip level when using the PNoC variants. For this comparison, we have assumed the individual core to be a 14nm x86-64 core from Intel, with the power consumption of the 64-core chip being 77.75 W. This assumption includes power consumption of 128 KB private L1 caches, 2 MB L2 cache (shared between four cores), and memory controllers (shared between four cores) [105]. This assumption sets the total power consumption for the baseline Clos PNoC-based system at 95.75 W and for the baseline SwiftNoC PNoC-based system at 109.75 W. Table 5 considers power and loss values for PNoC variants calculated using the standard parameter values from table 3. It can be seen that even at the entire chip-granularity, the ARXON framework provides notable reduction in overall power consumption, with ARXON-PAM4 based Clos and SwiftNoC PNoCs saving 10.97% and 16.86% power compared to the Clos and SwiftNoC PNoC baselines, respectively.



(a)



(b)

Figure 17. Power dissipation breakdown for standard and aggressive values ('aggr' in the plots) for (a) Clos and (b) SwiftNoC PNoCs.

Table 5 Comparison of power savings between systems implementing the discussed PNoC variants

PNoC variant	Total power (W)		Power savings (%)	
	Clos	SwiftNoC	Clos	SwiftNoC
truncation	94.25	106.25	1.57 %	3.19 %
[79]	94.95	108.25	0.84 %	1.37 %
LORAX-OOK	94.0	106.0	1.83 %	3.42 %
LORAX-PAM4	86.25	93.25	9.92 %	15.03 %
ARXON-OOK	90.75	100.25	5.22 %	8.66 %
ARXON-PAM4	85.25	91.25	10.97 %	16.86 %

2.6. CONCLUSIONS

In this chapter, we proposed a new framework called ARXON for loss-aware approximation of data communicated over PNoC architectures. We also studied how multilevel signaling can assist with the proposed approximation framework. We considered MR tuning and crosstalk mitigation strategies as avenues to save energy while our distance aware transmission technique is in effect. Our results indicate that using multilevel signaling as part of our framework can reduce laser-power consumption by up to 57.2% over a baseline PNoC architecture. Our framework also shows up to 56.4% lower laser power and up to 23.8% better energy-efficiency compared to the best-known prior work on approximating communication in PNoCs. These results highlight the potential of approximation in PNoC architectures to reduce energy and power consumption in emerging manycore platforms.

CHAPTER 3: ARCHITECTING NON-COHERENT PHOTONIC ACCELERATORS FOR HIGH ACCURACY, HIGH THROUGHPUT CONVOLUTION NEURAL NETWORK ACCELERATION

Many emerging applications such as self-driving cars, autonomous robotics, fake news detection, pandemic growth and trend prediction, and real-time language translation are increasingly being powered by sophisticated machine learning models. With researchers creating deeper and more complex deep neural network (DNN) architectures, including multi-layer perceptron (MLP) and convolution neural network (CNN) architectures, the underlying hardware platform must consistently deliver better performance while satisfying strict power dissipation limits. Such an endeavor to achieve higher performance-per-watt has driven hardware architects to design custom accelerators for deep learning, e.g., Google's TPU [24] and Intel's GOYA [26], with much higher performance-per-watt than conventional CPUs and GPUs.

Electronic accelerators architectures, unfortunately face fundamental limits in the post Moore's law era where processing capabilities are no longer improving as they did over the past several decades [106]. In particular, moving data electronically on metallic wires in these accelerators creates a major bandwidth and energy bottleneck [57]. Silicon photonics is a promising technology to enable energy-efficient, ultra-high bandwidth, and low-latency communication solutions [107]. CMOS-compatible photonic interconnects have already replaced metallic ones for light-speed data transmission at almost every level of computing, and are now actively being considered for chip-scale integration [32].

Remarkably, it is also possible to use optical components to perform computation, e.g., matrix-vector multiplication [34]. By employing on-chip waveguides, electro-optic modulators,

photodetectors, and lasers to build photonic interconnects and photonic integrated circuits, it is now possible to conceive a new class of DNN accelerators which are effective for low-latency and energy-efficient optical domain data transport and communication. Not only can such photonics-based accelerators address the fan-in and fan-out problems with linear algebra processors, but their operational bandwidth can approach the photodetection rate (typically in the hundreds of GHz), which is orders of magnitude higher than electronic systems today that operate at a clock rate of a few GHz [36].

Despite the above benefits, a number of obstacles must be overcome before viable photonic DNN accelerators can be realized. Fabrication process and thermal variations can adversely impact the robustness of photonic accelerator designs by introducing undesirable crosstalk noise, tuning overheads, resonance drifts, optical phase shifts, and photo-detection current mismatches. For example, experimental studies have shown that micro-ring resonator (MR) devices used in chip-scale photonic interconnects can experience significant resonant drifts (e.g., ~9 nm reported in [108]) within a wafer due to process variations. This matters because even a 0.25 nm drift can cause the bit-error-rate (BER) of photonic data traversal to degrade from 10-12 to 10-6. Moreover, thermal crosstalk in silicon photonic devices such as MRs can significantly reduce DNN model accuracy by limiting the achievable precision (i.e. resolution) of weight and bias parameters to a few bits. Common tuning circuits that rely on thermo-optic phase-change effects to control photonic devices, e.g., when imprinting activations or weights on optical signals, also place a limit on the achievable throughput and parallelism in photonic accelerators. Lastly, at the architecture level, there is a need for a scalable, adaptive, and low-cost computation and communication fabric that can handle the demands of diverse MLP and CNN models.

In this chapter, we introduce CrossLight, novel silicon photonic neural network accelerator that addresses the challenges highlighted above through a cross-layer design approach. By cross-layer, we refer to the design paradigm that involves considering multiple layers in the hardware-software design stack together, for a more holistic optimization of the photonic accelerator. CrossLight involves device-level engineering for resilience to fabrication-process variations and thermal crosstalk, circuit-level tuning enhancements for inference latency reduction, and an optimized architecture-level design that also integrates the device- and circuit-level improvements to enable higher resolution, better energy-efficiency, and improved throughput compared to prior efforts on photonic accelerator design. Our novel contributions in this chapter include:

- Improved silicon photonic device designs that we fabricated to make our architecture more resilient to fabrication-process variations;
- An enhanced tuning circuit to simultaneously support large thermal-induced resonance shifts and high-speed, low-loss device tuning;
- Consideration of thermal crosstalk mitigation methods to improve the weight resolution achievable by CrossLight architecture;
- Increased throughput and energy-efficiency by improving wavelength reuse and further use of matrix decomposition at the architecture-level;
- A comprehensive comparison with state-of-the-art accelerators that shows the efficacy of our cross-layer optimized solution.

3.1. RELATED WORK

Silicon-photonics based DNN accelerator architectures represent an emerging paradigm that can immensely benefit the landscape of deep learning hardware design [109], [110], [111], [112],

[113]. A photonic neuron in these architectures consists of three components: a weighting, a summing, and a nonlinear unit which is analogous to an artificial neuron. Noncoherent photonic accelerators, such as [110], [111], [112], typically employ the Broadcast and Weight (B&W) protocol [109] to manipulate optical signal power for setting and updating weights and activations. The B&W protocol is an analog networking protocol that uses wavelength-division multiplexing (WDM), photonic multiplexors, and photodetectors to combine outputs from photonic neurons in a layer. Coherent photonic accelerators, such as [36], [113], typically use only a single wavelength to manipulate the electrical field amplitude rather than signal power. Weighting occurs with electrical field amplitude attenuation proportional to the weight value, and phase modulation that is proportional to the sign of the weight. The weighted signals are then coherently accumulated with cascaded Y-junction combiners. For both types of accelerators, non-linearity can be implemented with devices such as electro-absorption modulators [36].

Due to the scalability, phase encoding noise, and phase error accumulation limitations of coherent accelerators [114], there is growing interest in designing efficient noncoherent photonic accelerators. In particular, the authors of DEAP-CNN [110] have described a noncoherent neural network accelerator that implements the entirety of the CNN layers using connected convolution units. The tuned MRs in these units assume the kernel values by using phase tuning to manipulate the energy in their resonant wavelengths. Holylight [111] is another noncoherent architecture that uses microdisks (instead of MRs) for its lower area and power consumption. It utilizes a "whispering gallery mode" resonance for microdisk operation, which unfortunately is inherently lossy due to a phenomenon called tunneling ray attenuation [115]. More generally, these noncoherent architectures suffer from susceptibility to process variations and thermal crosstalk, which are not addressed in these architectures. Microsecond-granularity thermo-optic tuning

latencies further reduce the speed and efficiency of optical computing [39]. We address these shortcomings as part of our proposed cross-layer optimized noncoherent photonic accelerator architecture in this chapter.

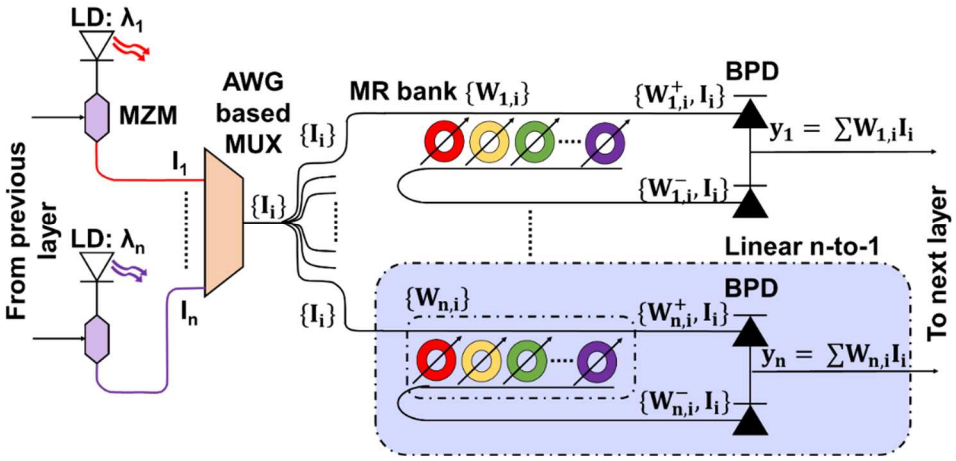


Figure 18. Noncoherent Broadcast-and-weight (B&W) based photonic neuron.

3.2. NONCOHERENT PHOTONIC COMPUTATION OVERVIEW

As mentioned earlier, noncoherent photonic accelerators typically utilize the Broadcast and Weight (B&W) photonic neuron configuration with multiple wavelengths. Figure 18 shows an example of this B&W configuration with *n* neurons in a layer where the colored-dotted box represents a single neuron. Each input to a neuron is imprinted onto a unique wavelength (λ_i) emitted by a laser diode (LD) using a Mach–Zehnder modulator (MZM). The wavelengths are multiplexed (MUXed) into a single waveguide using arrayed waveguide grating (AWG), and split into *n* branches that are each weighted with a micro-ring resonator (MR) bank that alters optical signal power proportional to weight values. Summation across positive and negative weight arms at each branch is performed using a balanced photodetector (BPD). Optoelectronic devices such as electro-absorption modulators (not shown for brevity) introduce non-linearity after the multiplication and summation operations.

MRs are the fundamental components that impact the efficiency of this configuration. Weights (and biases) are altered by tuning MRs so that the losses experienced by wavelengths—on which activations have been imprinted—can be modified to realize matrix-vector multiplication. MR-weight banks have groups of these tunable MRs, each of which can be tuned to drain energy from a specific resonant wavelength so that the intensity of the wavelength reflects a specific value (after it has passed near the MR). As an example of performing computation in the optical domain, consider the case where an activation value of 0.8 must be weighted by a value of 0.5 as part of a matrix-vector multiplication in a DNN model inference phase. Let us assume that the red wavelength (λ_1) is imprinted with the activation value of 0.8 by using the MZM in Figure 18 (alternatively, MRs can be used for the same goal, where an MR will be tuned in such a way that 20% of the input optical signal intensity is dropped as the wave traverses the MR). When λ_1 passes through an MR bank, e.g., the one in the dotted-blue box in Figure 18, the MR in resonance with λ_1 can be tuned to drop 50% of the input signal intensity. Thus, as λ_1 passes this MR, we will obtain 50% of the input intensity at the through port, which is 0.4 (=0.8×0.5). The BPD shown in Figure 18 then converts the optical signal intensity from that wavelength (and other wavelengths) into an electrical signal that represents an accumulated single value.

An MR is essentially an on-chip resonator which is said to be in resonance when an optical wavelength on the input port matches with the resonant wavelength of the MR, generating a Lorentzian-shaped signal at the through port. An all-pass MR and its output optical spectrum is shown in Figure 19. The free-spectral range (FSR) and extinction ratio (ER) are two primary characteristics of an MR. These depend on several physical properties in the MR, including its width, thickness, radius, and the gap between the input and ring waveguide [116]. Changing any of these properties changes the effective index (n_{eff}) of the MR, which in turn causes a change in

the output optical spectrum. It is crucial to maintain the central wavelength at the output optical spectrum for reliable operation of MRs. However, MRs are sensitive to fabrication-process variations (FPVs) and variations in surrounding temperature. These cause the central wavelength of the MR to deviate from its original position, causing a drift in the MR resonant wavelength ($\Delta\lambda_{MR}$). Such a drift (due to FPV or thermal variations) can be compensated using electro-optic (EO) or 73hermos-optic (TO) tuning mechanisms. Both of these have their own advantages and disadvantages. EO tuning is faster (~ns range) and consumes lower power (~4 μW/nm) but with a smaller tuning range [40]. In contrast, TO tuning has a larger tunability range, but consumes higher power (~27 mW/FSR) and has higher (~μs range) latency [39].

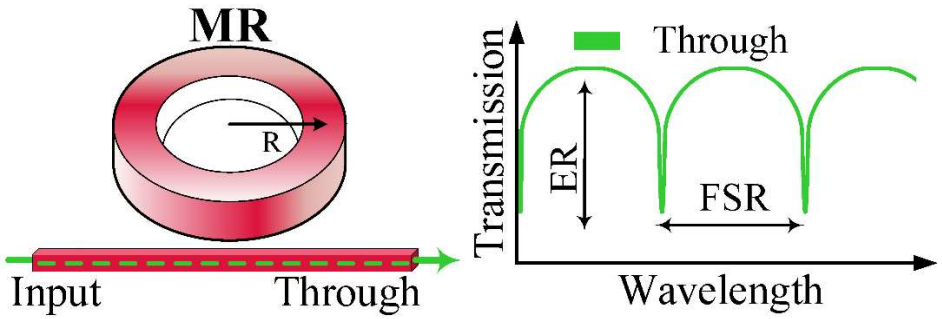


Figure 19. An all-pass MR with output spectral characteristics at the through port with extinction ratio (ER) and free spectral range (FSR) specified in the figure.

A large number of MRs must be used at the architecture-level to support complex MLP and CNN model executions. As the number of MRs increase, so does the length of the waveguide which hosts the banks. Unfortunately, this leads to an increase in the total optical signal propagation, modulation, and through losses experienced, which in turn increases the laser power required to drive the optical signals through the weight banks, so that they can be detected error-free at the photodetector. An excessive number of parallel arms with MR weight banks (the dotted box in Figure 18 represents one arm working in parallel with other arms) also increases optical

splitter losses. Moreover, without considering crosstalk mitigation strategies (as is the case with previously proposed photonic accelerators), the weight resolution of the architecture goes down with increased crosstalk noise in optical signals.

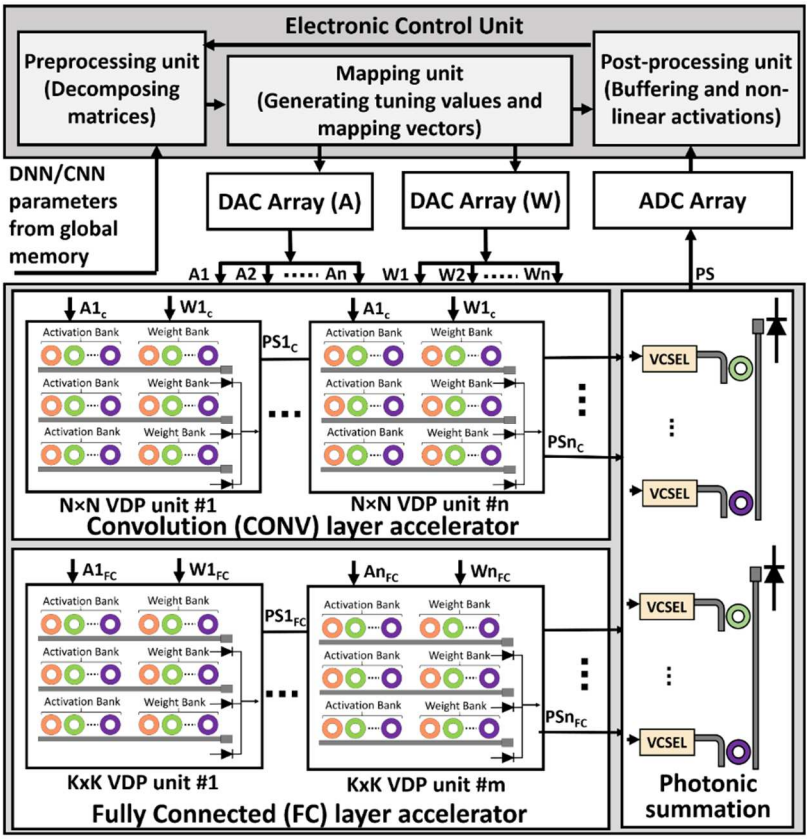


Figure 20. An overview of CrossLight, showing dedicated vector dot product (VDP) units for CONV and FC layer acceleration, and the internal architecture.

In summary, to design efficient photonic accelerators, there is a need for (1) improved MR device design to better tolerate variations and crosstalk; (2) efficient MR tuning circuits to quickly and reliably imprint activation and parameter values; and (3) a scalable architecture design that minimizes optical signal losses. Our novel CrossLight photonic accelerator design addresses all of these concerns and is discussed next.

3.3. CROSSLIGHT ARCHITECTURE

Figure 20 shows a high-level overview of our CrossLight noncoherent silicon photonic neural network accelerator. The photonic substrate performs vector dot product (VDP) operations using silicon photonic MR devices, and summation using optoelectronic photodetector (PD) devices over multiple wavelengths. An electronic control unit is required for the control of photonic devices, and for communication with a global memory to obtain the parameter values, partial sum buffering, and for mapping of the vectors. We use digital to analog converter (DAC) arrays to convert buffered signals into analog tuning signals for MRs. Analog to digital converter (ADC) arrays are used to map the output analog signals generated by PDs to digital values that are sent back for post-processing and buffering. We break down the discussion of this accelerator into three parts (subsections 3.4.1- 3.4.3), corresponding to the contributions at the device, tuning circuit, and architecture levels, as discussed next.

3.3.1 MR DEVICE ENGINEERING AND FABRICATION

Process variations are inevitable in CMOS-compatible silicon photonic fabrications, causing undesirable changes in resonant wavelength of MR devices ($\Delta\lambda_{MR}$). A 1.5×0.6 mm2 chip was fabricated using high-resolution Electron Beam (EBeam) lithography and we performed a comprehensive design-space exploration of MRs to compensate for FPVs while improving MR device insertion loss and Q-factor. In this exploration, we varied the input and ring waveguide widths to find an MR device design that was tolerant to FPVs. We found that in an MR design of any radii and gap, when the input waveguide is 400 nm wide and the ring waveguide is 800 nm wide at room temperature (300 K), the undesired $\Delta\lambda_{MR}$ due to FPVs can be reduced from 7.1 to 2.1 nm (70% reduction). This is a significant result, as these engineered MRs require less

compensation for FPV-induced resonant wavelength shifts, which can reduce the power consumption of architectures using such MRs.

Unfortunately, the impact of FPVs is not completely eliminated, even with such optimized MR designs, and there is still a need to compensate for FPVs. Thermal variations are another major factor to cause changes in MR n_{eff} which also leads to undesirable $\Delta\lambda_{MR}$. Thermo-optic (TO) tuners are used to compensate for such deviations in $\Delta\lambda_{MR}$. These TO tuners use microheaters to change the temperature in the proximity of an MR device, which then alters the n_{eff} of the MR, changing the device resonant wavelength, and correcting the $\Delta\lambda_{MR}$. High temperatures from such heaters can unfortunately cause thermal energy dissipation, creating thermal crosstalk across MR devices placed close to each other. One can avoid such thermal crosstalk by placing devices at an appropriate distance from each other, typically 120 µm to 200 µm (depending on the number of MR devices in proximity within an MR bank). But such a large spacing hurts area efficiency and also increases waveguide length, which increases propagation losses and its associated laser power overhead. We propose to address this challenge at the circuit level, as discussed in the next section.

3.3.2 TUNING CIRCUIT DESIGN

To reduce thermal crosstalk, we must reduce the reliance on TO tuning, an approach that is used in all prior photonic neural network accelerators, but one that entails high overheads. We propose to use a hybrid tuning circuit where both thermo-optic (TO) and electro-optic (EO) tuning are used to compensate for $\Delta\lambda_{MR}$. Such a tuning approach has previously been proposed in [117] for silicon photonic Mach–Zehnder Interferometers with low insertion loss. Such an approach can be easily transferred to an optimized MR for hybrid tuning in our architecture. The hybrid tuning approach supports faster operation of MRs with fast EO tuning to compensate for small $\Delta\lambda_{MR}$ shifts and, using TO tuning when necessary to compensate for large $\Delta\lambda_{MR}$ shifts. To further reduce

the power overhead of TO tuning in this hybrid approach, we adapt a method called Thermal Eigen Decomposition (TED), which was first proposed in [54]. Using TED, we can collectively tune all the MRs in an MR bank to compensate for large $\Delta\lambda_{MR}$ shifts. By doing so, we can cancel the effect of thermal crosstalk (i.e., an undesired phase change) in MRs with much lower power consumption. The TO tuning power can be calculated by the amount of phase shift necessary to apply to the MRs in order for them to be at their desired resonant wavelength. The extent of phase crosstalk ratio (due to thermal crosstalk) as a function of the distance between an MR pair is shown in Figure 21, for our fabricated MR devices. The results are based on detailed analysis with a commercial 3D heat transport simulation EDA tool for silicon photonic devices (Ansys Lumerical HEAT [118]). It can be seen from the orange line that the amount of phase crosstalk reduces exponentially as the distance between an MR pair increases. Such a trend has also been observed in [119]. To find a balance between tuning power savings while having reduced crosstalk, we perform a sensitivity analysis based on the distance between two adjacent MRs in our architecture. We placed the optimized MRs (described in the previous section) in such a manner that maximum tuning power is saved when they are close to each other while compensating for thermal crosstalk. Results from our analysis (the solid-blue line in Figure 21) indicate that placing each MR pair at a distance of 5 μm is optimal, as decreasing or increasing such a distance causes an increase in power consumption of individual TO heaters in the MRs. Figure 21 also shows the tuning power required without using the TED approach (blue dotted line), which can be seen to be notably higher.

The workflow of our circuit-level hybrid tuning approach can be summarized as follows. When the accelerator is first booted at runtime, a one-time compensation for design-time FPVs is applied using TO tuning. The extent of compensation for crosstalk is calculated offline during the test phase, where the required phase shift in each of the MRs is calculated, and once the system is

online, the respective phase shift values are applied to cancel the impact of thermal crosstalk. Subsequently, we apply EO tuning due to its extremely low latency to represent vector elements in each vector operation with MRs (discussed in more detail in the next section). If large shifts in temperature are observed at runtime, we can perform a one-time calibration with TO tuning to compensate for it. In our analysis, runtime TO tuning would be required rarely beyond its first use after the initial bootup of the photonic accelerator platform.

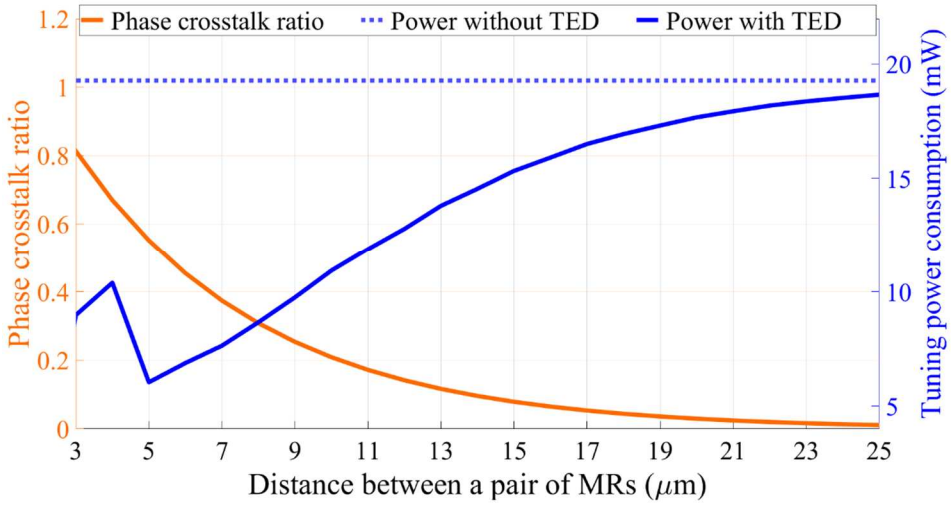


Figure 21. Phase crosstalk ratio and tuning power consumption in a block of 10 fabricated MRs with variable distance between adjacent pair of MRs.

3.3.3 ARCHITECTURE DESIGN

The optimized MR devices, layouts, and tuning circuits are utilized within optical vector dot product (VDP) units, which are shown in Figure 20. We use banks (groups) of MRs to imprint both activations and weights onto the optical signal. At the architecture level, we compose multiples of VDP units into two architectural sub-components: one to support convolution (CONV) layer acceleration and the other to support fully connected (FC) layer acceleration. We focus on these two types of layers as they are the most widely used and consume the most

significant amount of latency and power in computational platforms that execute DNNs. In contrast, other layer types (e.g., pooling, batch normalization) can be implemented very efficiently in the electronic domain. Note also that we focus on inference acceleration, as done in all photonic DNN accelerators, and almost all electronic DNN accelerators.

3.3.3.1 DECOMPOSITION VECTOR IN CONV/FC LAYERS

To map CONV and FC layers from DNN models to our accelerator, we first need to decompose large vector sizes into smaller ones. In CONV layers, a filter performs convolution on a patch (e.g., 2×2 elements) of the activation matrix in a channel to generate an element of the output matrix. The operation can be represented as follows:

$$K \otimes A = Y \qquad (4)$$

For a 2×2 filter kernel and weight matrices, (4) can be expressed as:

$$\begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix} \otimes \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} = k_1a_1 + k_2a_2 + k_3a_3 + k_4a_4 \qquad (5)$$

Rewriting (5) as a vector dot product, we have:

$$[k_1 \ k_2 \ k_3 \ k_4] . \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = k_1a_1 + k_2a_2 + k_3a_3 + k_4a_4 \qquad (6)$$

Once we represent the operation as a vector dot product, it is easy to see how it can be decomposed into partial sums. For example:

$$[k_1 \ k_2] . \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = k_1a_1 + k_2a_2 = PS_1$$

$$[k_3 \ k_4] . \begin{bmatrix} a_3 \\ a_4 \end{bmatrix} = k_3a_3 + k_4a_4 = PS_2$$

$$PS_1 + PS_2 = Y \qquad (7)$$

In FC layers, typically much larger dimension vector multiplication operations are performed between input activations and weight matrices:

$$AW = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} [w_1 \quad w_2 \dots w_n] \tag{8}$$

$$AW = \begin{bmatrix} a_1 \cdot w_1 & + a_1 \cdot w_2 & + \cdots & a_1 \cdot w_n \\ a_2 \cdot w_1 & + a_2 \cdot w_2 & + \cdots & a_2 \cdot w_n \\ & \vdots & & \\ a_n \cdot w_1 & + a_n \cdot w_2 & + \cdots & a_n \cdot w_n \end{bmatrix} \tag{9}$$

In (8), a_1 to a_n represent a column vector of activations (A) and w_1 to w_n represent a row vector of weights (W). The resulting vector is a summation of dot products of vector elements (9). Much like with CONV layers, these can be decomposed into lower dimensional dot products.

3.3.3.2. VECTOR DOT PRODUCT (VDP) UNIT DESIGN

We separated the implementation of CONV and FC layers in *CrossLight* due to different orders of vector dot product computations required to implement each layer. For instance, typical CONV layer kernel sizes vary from 2×2 to 5×5, whereas in FC layers it is not uncommon to have 100 or more neurons (requiring 100×100 or higher order multiplication). State-of-the-art photonic DNN accelerators, e.g., [110], only consider the scales involved at the CONV layer, and either only support CONV layer acceleration in the optical domain, or use the same CONV layer implementation to accelerate FC layers. This leads to increased latencies and reduced throughput as the larger vectors involved with FC layer calculation must be divided up into much smaller chunks, in the order of the filter kernel size of the CONV layer.

For improved efficiency, we separately support the unique scale and requirements of vector dot products involved in CONV and FC layers. For CONV layer acceleration, we consider *n* VDP units, with each unit supporting an *N*×*N* dot product. For FC layer acceleration, we consider *m*

units, with each unit supporting a $K \times K$ dot product. Here $n > m$ and $K > N$, as per the requirements of each of the distinct layers. In each of the VDP units, the original vector dimensions are decomposed into N or K dimensional vectors, as discussed above. We performed an exploration to determine the optimal values for N, K, n, and m. The results of this exploration study are presented in Section 3.5.

3.3.3.3 OPTICAL WAVELENGTH REUSE IN VDP UNITS

Prior work on photonic DNN accelerator design typically considers a separate wavelength to represent each individual element of a vector. This approach leads to an increase in the total number of lasers needed in the laser bank (as the size of the vectors increases) which in turn increases power consumption. Beyond employing the decomposition approach discussed above, we also consider wavelength reuse per VDP unit to minimize laser power. In this approach, within VDP units, the N or K dimensional vectors are further decomposed into smaller sized vectors for which dot products can be performed using MRs in parallel, in each arm of the VDP unit. The same wavelengths can then be reused across arms within a VDP to reduce the number of unique wavelengths required from the laser. PDs perform summation of the element-wise products to generate partial sums from decomposed vector dot products. The partial sums from the decomposed operations are then converted back to the photonic domain by VCSELs (bottom right of Figure 20), multiplexed into a single waveguide, and accumulated using another PD, before being sent for buffering. Thus, our approach leads to an increase in the number of PDs compared to other accelerators but significantly reduces both the number of MRs per waveguide and the overall laser power consumption.

In each arm within a VDP unit, we used a maximum of 15 MRs per bank for a total of 30 MRs per arm, to support up to a 15×15 vector dot product. The choice of MRs per arm considers not

only the thermal crosstalk, layout spacing issues (discussed earlier), and the benefits of wavelength reuse (discussed in previous para), but also non-negligible optical splitter losses as the number of MRs per arm increases, which in turn increases laser power requirements. Thus, the selection of MRs per arm within a VDP unit was carefully adjusted to balance parallelism within/across arms, and laser power overheads.

3.4. EVALUATION AND SIMULATION SETUP

3.4.1 SIMULATION SETUP

To evaluate the effectiveness of our *CrossLight* accelerator, we conducted several simulation studies. These studies were complemented by our MR-device fabrication and optimization efforts on real chips, as discussed in Section 3.4. We considered the four DNN models shown in Table 6 for execution on the accelerator. Model 1 is Lenet5 [120] and models 2 and 3 are custom CNNs with both FC and CONV layers. Model 4 is a Siamese CNN utilizing one-shot learning. The datasets used to train these models are also listed in the table. We designed a custom *CrossLight* accelerator simulator in Python to estimate its performance and power/energy. We used Tensorflow 2.3 along with Qkeras [121], for analyzing DNN model accuracy across different parameter resolutions.

Table 6 Models and datasets considered for evaluation

Model no.	CONV layers	FC layers	Parameters	Datasets
1	2	2	60,074	Sign MNIST
2	4	2	890,410	CIFAR10
3	7	2	3,204,080	STL10
4	8	4	38,951,745	Omniglot

We compared *CrossLight* with the DEAP-CNN [110] and Holylight [111] photonic DNN accelerators from prior work. Table 7 shows the optoelectronic parameters considered for this simulation-based analysis. We considered photonic signal losses due to various factors: signal propagation (1 dB/cm [32]), splitter loss (0.13 dB [122]), combiner loss (0.9 dB [123]), MR through loss (0.02 dB [124]), MR modulation loss (0.72 dB [125]), microdisk loss (1.22 dB [126]), EO tuning loss (6 dB/cm [40]), and TO tuning loss (1 dB/cm [39]). We also considered the 1-to-56-Gb/s ADC/DAC-based transceivers from recent work [127]. To calculate laser power consumption, we use the following laser power model:

$$P_{laser} - S_{detector} \geq P_{photo\_loss} + 10 \times \log_{10} N_{\lambda} \quad (10)$$

where P_{laser} is laser power in dBm, $S_{detector}$ is the PD sensitivity in dBm, and P_{photo_loss} is the total photonic loss encountered by the optical signal, due to all of the factors discussed above.

Table 7 Parameters considered for analyses of photonic accelerators

Devices	Latency	Power
EO Tuning [40]	20 ns	4 μW/nm
TO Tuning [39]	4 μs	27.5 mW/FSR
VCSEL [128]	10 ns	0.66 mW
TIA [129]	0.15 ns	7.2 mW
Photodetector [130]	5.8 ps	2.8 mW

3.4.2 RESULTS: CROSSLIGHT RESOLUTION ANALYSIS

We first present an analysis of the resolution that can be achieved with *CrossLight*. We consider how the optical signals from MRs impact each other due to their spectral proximity, also known as inter-channel crosstalk. For this, we use the equations from [131]:

$$\varphi(i,j) = \frac{\delta^2}{(\lambda_i - \lambda_j)^2 + \delta^2} \quad (11)$$

In (11), $\varphi(i,j)$ describes the noise content from the j^{th} MR present in the signal from the i^{th} MR. As the noise content increases, the resolution achievable with *CrossLight* will decrease. Also, $(\lambda_i - \lambda_j)$ is the difference between the resonant wavelengths of i^{th} MR and j^{th} MR, while δ (= $\lambda_i/2Q$) denotes the 3dB bandwidth of the MRs, with Q being the quality factor (Q-factor) of the MR being considered. The noise power component can thus be calculated as:

$$P_{noise} = \sum_{i}^{n-1} \varphi(i,j)\, P_{in}[i] \qquad (12)$$

For unit input power intensity, resolution can then be computed as:

$$Resolution = \frac{1}{max|P_{noise}|} \qquad (13)$$

From this analysis, we found that with the FSR value of 18 nm, the Q-factor value of ~8000 in our optimized MR designs, and the wavelength reuse strategy in *CrossLight*, which allows us to have large ($\lambda_i - \lambda_j$) values (>1 nm), our MR banks will be able to achieve a resolution of 16 bits for up to 15 MRs per bank (Section 3.4.3.2). This is much higher than the resolution achievable by many photonic accelerators. For instance, DEAP-CNN can only achieve a resolution of 4 bits, whereas Holylight can only achieve a 2-bit resolution per microdisk (they however combine 8 microdisks to achieve an overall 16-bit resolution). Higher resolution ensures better accuracy in inference, which can be critical in some applications. Figure 22 shows the impact of varying the resolution across the weights and activations from 1 bit to 16 bits (we used quantization-aware training to maximize accuracy), for the four DNN models considered (Table 6). A crucial observation is that model inference accuracy is sensitive to the resolution of weight and activation parameters. Models such as the one for STL10 are particularly sensitive to the resolution. Thus, the high resolution afforded by *CrossLight* can allow achieving higher accuracies than other photonic DNN accelerators, such as DEAP-CNN.

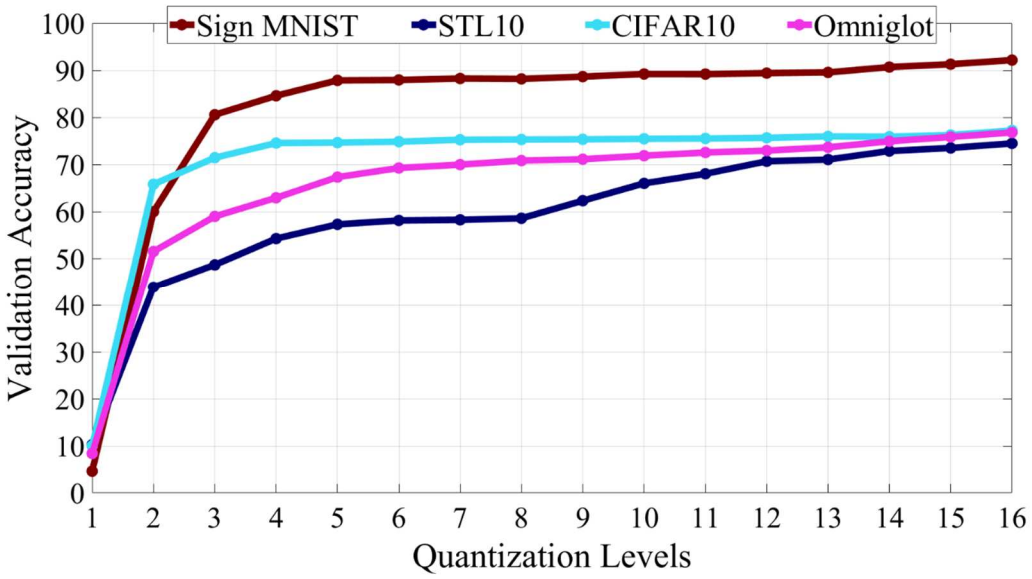


Figure 22. Inference accuracy of the four DNN models considered, across quantization (resolution) range from 1 bit to 16 bits (for both weights and activations).

3.4.3. RESULTS: CROSSLIGHT SENSITIVITY ANALYSIS

We performed a sensitivity analysis by varying the number of VDP units in the CONV layer accelerator (n) and FC layer accelerator (m), along with the complexity of the VDP units (N and K, respectively).

Figure 23 shows the frames per second (FPS; a measure of inference performance) vs. energy per bit (EPB) vs. area of various configurations of *CrossLight*. We selected the best configuration as the one which had the highest value of FPS/EPB. In terms of (N, K, n, m), the values of the four parameters for this configuration are (20, 150, 100, 60). This configuration also ended up being the one with the highest FPS value, but had a higher area overhead than other configurations. Nonetheless, this area is comparable to that of other photonic accelerators. This configuration was used for comparisons with prior work, as discussed next.

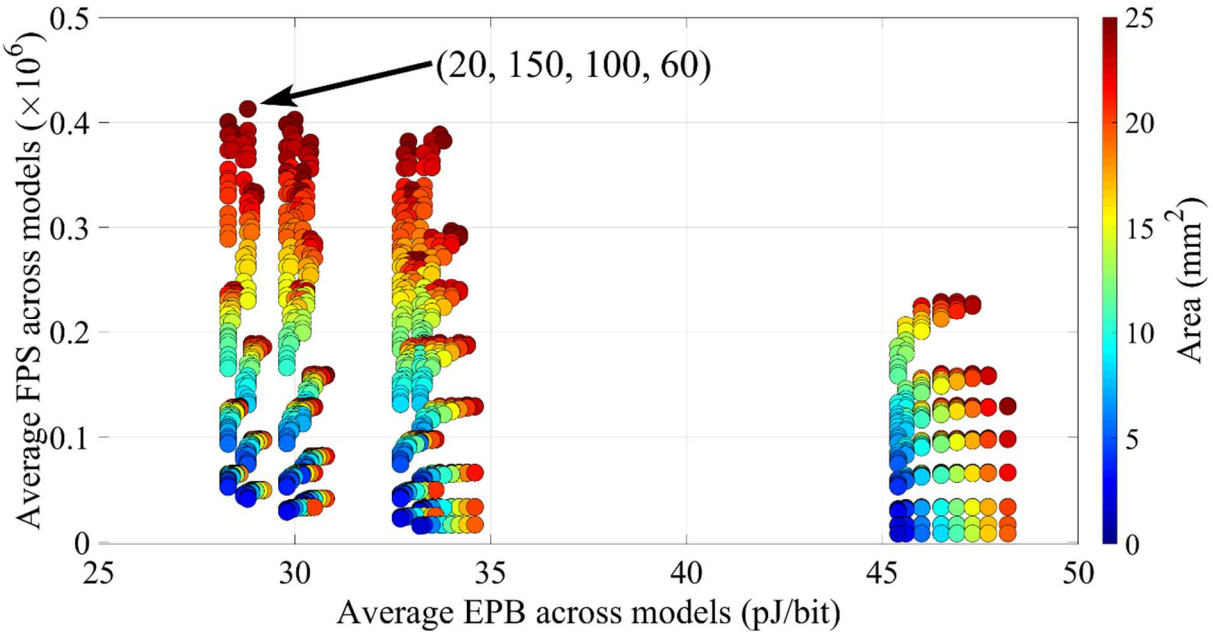


Figure 23. Scatterplot of average FPS vs. average EPB vs. area of various CrossLight configurations. The configuration with highest FPS/EPB (and FPS) is highlighted.

3.4.4. RESULTS: COMPARISON WITH STATE-OF-THE-ART ACCELERATORS

We compared our *CrossLight* accelerator against two well-known photonic accelerators: DEAP-CNN and Holylight, within a reasonable area constraint for all accelerators (~16-25 mm^2). We present results for four variants of the *CrossLight* architecture: 1) *Cross_base* utilizes conventional MR designs (without FPV resilience) and traditional TO tuning; 2) *Cross_opt* utilizes the optimized MR designs from Section 3.4.1, and traditional TO tuning; 3) *Cross_base_TED* utilizes the conventional MR designs with the hybrid TED-based tuning approach from Section 3.4.2; and 4) *Cross_opt_TED* utilizes the optimized MR designs and the hybrid TED-based tuning approach.

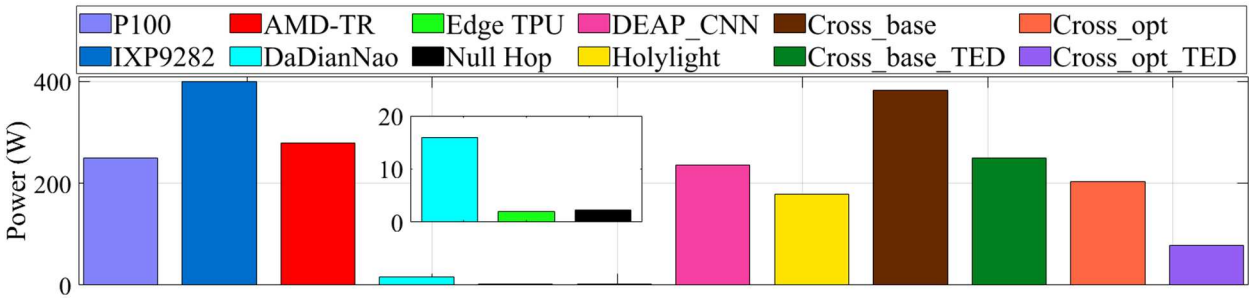


Figure 24. Power consumption comparison among variants of CrossLight vs. photonic accelerators (DEAP-CNN, Holylight), and electronic accelerator platforms (P100, Xeon Platinum 9282, Threadripper 3970x, DaDianNao, EdgeTPU, Null Hop).

Figure 24 shows the power consumption comparison across the four *CrossLight* variants and the two photonic accelerators from prior work. We also included numbers for comparing electronic platforms: three deep learning accelerators (DaDianNao, Null Hop, and EdgeTPU), a GPU (Nvidia Tesla P100), and CPUs (Intel Xeon Platinum 9282 denoted as IXP9282, and AMD Threadripper 3970x denoted as AMD-TR) [132]. The difference in power values between the *CrossLight* variants arises due to the optimization approaches adopted in each of the variant. The variants which considered conventional MR design instead of the optimized designs have larger power consumption for compensating for FPV. This value becomes non-trivial as the number of MRs increase, and thus having reduced tuning power requirement per MR (in *Cross_opt* and *Cross_opt_TED*) becomes a significant advantage. Using the TED based hybrid tuning approach provides further significant power benefits for *Cross_opt_TED* over *Cross_opt*, which uses conventional TO tuning. *Cross_opt_TED* can be seen to have lower power consumption than both photonic accelerators, as well as the CPU and GPU platforms, although this power is higher than that of the edge/mobile electronic accelerators.

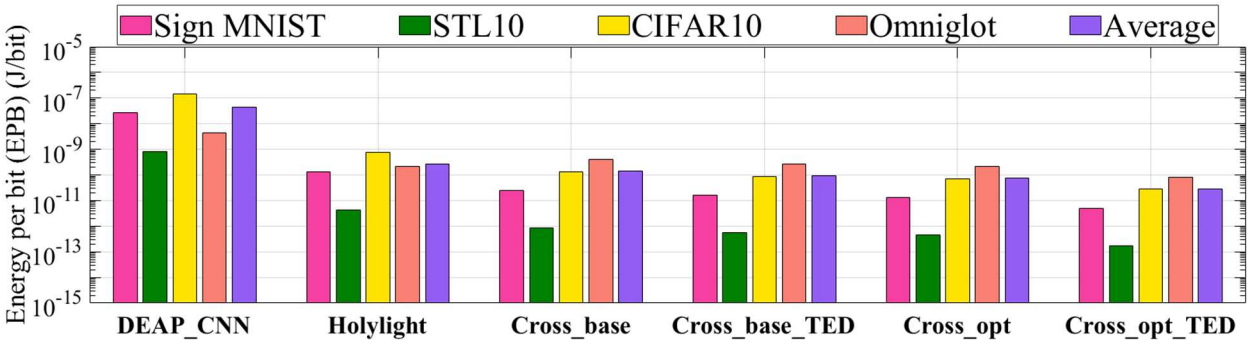


Figure 25 Comparison of EPB values of the photonic DNN accelerators.

EPB compared to DEAP-CNN and Holylight, respectively. *CrossLight* has lower EPB is because we comprehensively took into consideration various losses and crosstalk that a photonic DNN accelerator would experience, and put in place novel approaches at the device, circuit, and architecture layers to counteract their impact in *CrossLight*. The utilization of TED-based thermal crosstalk management allows us to have MRs placed much closer together, which in turn reduces propagation losses. Additionally, *CrossLight* considers a combination of TO and EO tuning which enables the reduction of power and EPB as well. The use of EO tuning in our hybrid tuning approach also provides the advantage of lower latencies, which is apparent in the EPB values.

Table 8 summarizes the average values of EPB (in pJ/bit) and performance-per-watt (in kiloFPS/Watt) of the photonic accelerators as well as the electronic accelerators considered in this chapter. It can be observed that the best *CrossLight* configuration (*Cross_opt_TED*) achieves significantly lower EPB and higher performance-per-watt values than all of the accelerators considered. Specifically, against Holylight, which is the best out of the two photonic DNN accelerators considered, *CrossLight* achieves 9.5× lower energy-per-bit and 15.9× higher performance-per-watt. Our chapter demonstrates the effectiveness of cross-layer design of deep learning accelerators with the emerging silicon photonics technology.

Table 8 Average EPB and kiloFPS/Watt values across accelerators

Accelerator	Avg. EPB (pJ/bit)	Avg. kiloFPS/watt
P100	971.31	24.9
IXP 9282	5099.68	2.39
AMD-TR	5831.18	2.09
DaDianNao	58.33	0.65
Edge TPU	697.37	17.53
Null Hop	2727.43	4.48
DEAP_CNN	44453.88	0.07
Holylight	274.13	3.3
Cross_base	142.35	10.78
Cross_base_TED	92.64	16.54
Cross_opt	75.58	20.25
Cross_opt_TED	28.78	52.59

3.5. CONCLUSION

In this chapter, we presented a novel cross-layer optimized photonic neural network accelerator called *CrossLight*. Utilizing silicon photonic device-level fabrication-driven optimizations along with circuit-level and architecture-level optimizations, we demonstrated 9.5× lower energy-per-bit and 15.9× higher performance-per-watt compared to state-of-the-art photonic DNN accelerators. *CrossLight* also shows improvements in these metrics over several CPU, GPU, and custom electronic accelerator platforms considered in our analysis. *CrossLight* shows the promise of cross-layer optimization strategies in countering various challenges such as crosstalk, fabrication-process variations, high laser power, and excessive tuning power. The results presented in this chapter demonstrates the promise of photonic DNN accelerators in addressing the need for energy-efficient and high performance-per-watt DNN acceleration.

CHAPTER 4: FABRICATION PROCESS VARIATION RESILIENT PHOTONIC BINARIZED NEURAL NETWORK ACCELERATOR

Over the last decade, machine learning (ML) applications have become increasingly prevalent, with many emerging applications, such as autonomous transportation, medical prognosis, real-time speech translation, network anomaly detection, and audio/video synthesis. This prevalence is fueled by the emergence of sophisticated and powerful machine learning models over the past decade, such as Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs). More sophisticated CNN models usually warrant deeper models with higher connectivity, which in turn increase the compute power and the memory requirement necessary to train and deploy them. Such increasing complexity also necessitates that the underlying hardware platforms consistently deliver better performance while satisfying strict power requirements. This endeavor to achieve high performance-per-watt has driven hardware architects to design custom accelerators for deep learning, e.g., Google's TPU [24] and Intel's GOYA [26], with much higher performance-per-watt than CPUs and GPUs. The performance-per-watt requirement still remains a challenge in resource-constrained environments, where computational power, energy expenditure, and available memory are often limited, such as many embedded devices. Binarized Neural Networks (BNNs) [133], [134] can reduce memory and computational requirements of DNN and CNN models while offering competitive accuracies with full precision models. As such, they are a possible solution to the performance requirement challenge, when executed on custom accelerators.

Exploring more efficient hardware accelerator platforms is another potential solution to reduce performance-per-watt for neural-network processing. Conventional electronic accelerator platforms face fundamental limits in the post-Moore era where the high costs and diminishing

performance improvements with semiconductor-technology scaling prevent significant improvements in future product generations [106]. Moving data in accelerators is a well-known bottleneck in these accelerators, due to the bandwidth and latency limitations of electronic interconnects, which puts limits on achievable performance and energy savings [57]. A solution to the data-movement bottleneck has presented itself in the form of silicon photonics technology, which enables ultra-high bandwidth, low-latency, and energy-efficient communication [99], [86], [103], [135], [136]. CMOS-compatible optical interconnects have already replaced metallic ones for light-speed data transmission at almost every level of computing, and are now actively being considered for chip-scale integration [135]. Recent research work has also shown that, it is also possible to use optical components to efficiently perform computation, e.g., matrix-vector multiplication [34], [43], [45]. Due to the emergence of both chip-scale optical communication and computation, it is now possible to conceive photonic integrated circuits (PICs) that offer low latency and energy-efficient optical domain data transport and computation.

Despite the benefits of utilizing photonics for computation and communication, there are several challenges that must be addressed before photonic accelerators become truly viable. One of the main obstacles that impacts the robustness and reliability of photonic accelerators is the sensitivity of photonic devices to fabrication process and thermal variations. These variations introduce undesirable crosstalk, optical phase shifts, frequency drifts, tuning overheads, and photodetection current mismatches, which adversely affect the reliable and robust operation of photonic accelerators. In order to correct the impact of variations, thermo-optic (TO) or electro-optic (EO) tuning circuits are often used, which have notable power overheads. Because of the phase-change effects it has on photonic devices, tuning mechanisms may also be used to control weight/activation imprinting via microring resonators (MRs). But the high latency of operation (in

μs range [39]) of TO tuning can limit the achievable throughput and parallelism in photonic accelerators.

In this chapter, we discuss *ROBIN* [44], a novel optical-domain BNN accelerator that addresses the challenges highlighted above by optimizing electro-optic components across the device, circuit, and architecture layers. *ROBIN* combines novel device- and circuit-level techniques to achieve more efficient fabrication-process-variation (FPV) correction in optical devices, which helps with reducing energy and improving accuracy in BNNs that utilize these devices. Additionally, circuit-level tuning enhancements for inference latency reduction, and an optimized architecture-level design help improve performance and also energy consumption compared to the state-of-the-art. The novel contributions from [44] include:

- The design of a novel optical-domain BNN accelerator architecture that is robust to fabrication process variations (FPVs) and thermal variations, and utilizes efficient wavelength reuse and a modular structure to enable high throughput and energy-efficient execution across BNN models;
- A novel integration of heterogeneous optical microring resonator (MR) devices; we also conduct design space exploration for these MR designs to determine device characteristics for efficient BNN execution;
- An enhanced tuning circuit to simultaneously support large thermal-induced resonance shifts and high-speed, low-loss device tuning to compensate for FPVs ;
- A comprehensive comparison with state-of-the-art BNN and non-BNN accelerator platforms from the optical and electronic domains, to demonstrate the potential of our BNN accelerator platform.

The rest of the chapter is organized as follows: Section 4.1 briefly explores the related works in the field of BNN acceleration. Section 4.2 gives a brief overview of non-coherent optical computation for photonic accelerators similar to ours. Section 4.3 provides an overview of BNNs and the partially binarized approach we have adopted for better accuracy in models. Section 4.4 describes the *ROBIN* architecture and our optimization efforts in tuning circuits, photonic devices, and photonic system level. Details of the experiments conducted, simulation setup, and the obtained results are provided in Section 4.5. Finally, Section 4.6 presents some concluding remarks.

4.1. BACKGROUND AND RELATED WORK

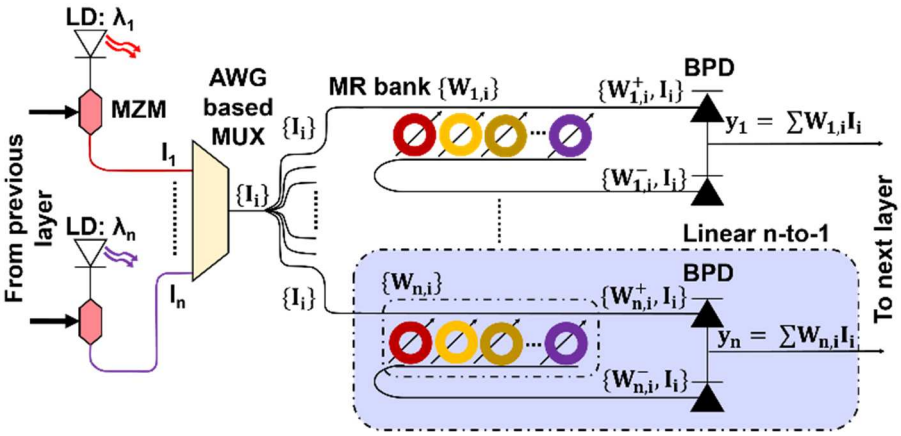
Silicon-photonic-based DNN accelerator architectures are becoming increasingly prominent with significant interest from both academic and industrial research communities [37]. This growth in interest can be attributed to the previously discussed benefits of photonic acceleration over electronic acceleration. Optical DNN accelerator architectures can be broadly classified into two types: coherent architectures and non-coherent architectures. Coherent architectures use a single wavelength to operate and imprint weight/activation parameters onto the electrical field amplitude of the light wave [36], [113]. These architectures mainly use on-chip optical interferometer devices called Mach-Zehnder Interferometers (MZIs). For imprinting the parameters, optical phase-change mechanisms are introduced to MZI devices. These mechanisms use heating or carrier injection to change the refractive index in the MZI structure. Weighting occurs with electrical field amplitude attenuation proportional to the weight value, and phase modulation that is proportional to the sign of the weight. The weighted signals are then accumulated with cascaded optical combiners, through coherent interference. Here the term coherent refers to the physical property of the wave, where it is possible for waves of the same wavelength to interfere constructively or destructively.

Non-coherent architectures, such as [43], [45], [110], [111], [112], use multiple wavelengths. These architectures are referred to as non-coherent architectures as they use different optical wavelengths, the interaction among which can be non-coherent. A large number of neuron operations can be represented simultaneously in non-coherent architectures by using wavelength-division multiplexing (WDM) or dense WDM (DWDM). In these architectures, parameter values are imprinted on to the signal amplitude directly, and to manipulate individual wavelengths, wavelength-selective devices such as microring resonators (MRs) or microdisks are used. The optical signal power is controlled, for imprinting parameter values, by controlling the optical loss in these devices through tuning mechanisms (Section 4.4.1). The Broadcast and Weight (B&W) protocol [109] is typically employed for setting and updating the weight and activation values. The *ROBIN* architecture we present in this chapter is a non-coherent architecture, i.e., it uses multiple wavelengths that are routed to photonic computation units in waveguides using WDM in accordance with the B&W protocol. The growing interest in non-coherent architectures can be attributed to the limitations in scalability, phase encoding noise, and phase error accumulation in coherent architectures [37], [114].

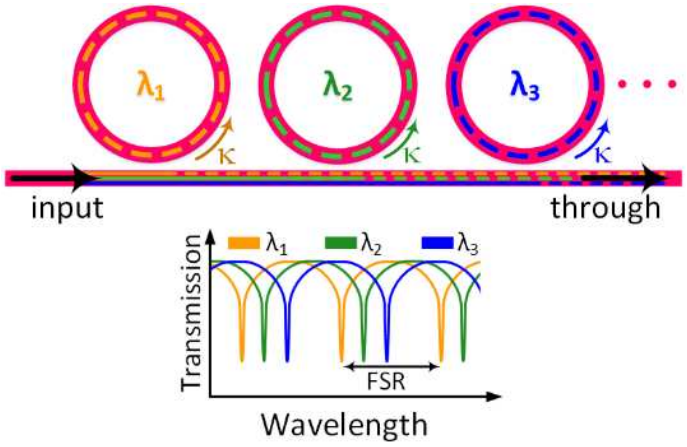
For optical DNN acceleration using non-coherent mechanics, [110] introduced a photonic accelerator for CNNs where all the layers of CNN models are implemented using connected photonic convolution units. In these units, MRs are used to tune wavelength amplitudes to desired kernel values. Another such work, in [111], utilizes microdisks instead of MRs due to the lower area and power consumption they offer. But microdisks use 'whispering gallery mode' resonance which is inherently lossy due to the tunneling ray attenuation phenomenon [115]; which reduces reliability and energy-efficiency with microdisks. There are very few works which focus on implementations of BNN accelerators using silicon photonics. The work in [137] proposed an MR-

based accelerator for discretized neural network acceleration, with an encoding scheme to enable positive and negative product considerations. The authors in [138] leveraged microdisks for implementing an accelerator with a design similar to [111]. This work considered an accelerator for fully binarized neural networks, i.e., both weights and activations and considered to be single-bit parameters. Because of this simplification, [138] was able to utilize energy-efficient photonic XOR and population count operations instead of conventional multiply and accumulate operations. The work also made use of photonic non-volatile memory and claimed operating frequencies of up to 50 GHz. All of these existing works on non-coherent optical-domain DNN/BNN acceleration have several shortcomings. They suffer from susceptibility to fabrications process variations (FPVs) and thermal crosstalk, which are not addressed in these architectures. Microsecond-granularity thermo-optic tuning latencies further can reduce the speed and efficiency of optical computing [39], which is also not considered when analyzing accelerator performance. We address these crucial shortcomings as part of our *ROBIN* optical-domain BNN accelerator architecture in this work.

In this work, we aim to ensure the robustness of the architecture against process and thermal variations by using MR design-space exploration and photonic tuning-circuit optimizations, which will be further explained in Section 4.4. We also utilize the broadband capabilities of the key photonic device in our work, microring resonators (MRs), to perform batch normalization folding, which moves batch normalization operations from the electrical domain to the photonic domain. Section 4.4.3 further details the modular architectural design aiming at ensuring wavelength reuse, to reduce VCSEL usage and splitter losses and waveguide length reduction. We also explore how the architecture performs in the presence of FPVs and how we may further reduce energy consumption in terms of device tuning in this scenario, in Section 4.5.2.



(a)



(b)

Figure 26. (a) A recurrent noncoherent B&W MAC based design [109]; (b) An MR bank consisting of MRs with individual resonant wavelength (λi) coupled to the MRs at cross-over coupling (κ) and the output spectrum, showing free spectral range (FSR).

4.2. OVERVIEW OF NON-COHERENT OPTICAL COMPUTATION

Non-coherent optical accelerators leverage the low-latency and energy-efficient optical computation for multiply and accumulate (MAC) operations, which consumes substantial computational power and incurs high latencies in electronic accelerators. These accelerators

typically utilize the B&W protocol with multiple wavelengths. Figure 26(a) (from [35]) gives an overview of a B&W-based optical MAC unit. The figure depicts a recurrent MAC unit which is employed repeatedly to compute different layers of a neural network model. The layer parameters such as weights or activations can be imprinted on to the wavelengths using the MRs that are tuned to modify the optical signal amplitude to represent those values. The MRs are placed in MR banks where multiple parameters can be imprinted onto wavelengths simultaneously. In the MR banks, each MR is tuned to a specific optical wavelength and can be used to alter the amplitude of the wavelength to represent the imprinted parameter. There can be separate wavelengths which carry positive and negative parameters, as discussed in [137]; these parameters are summed using balanced photodetectors (BPDs), as shown in Figure 26(a).

The output from the MAC unit is passed on to a Mach-Zehnder Modulator (MZM) which tunes the output from a designated laser diode (LD) to this output. Multiple MZMs and LDs are used to generate the outputs from multiple MAC units; these are collected and multiplexed using an arrayed waveguide grating (AWG) based optical multiplexer (MUX). The output from the MUX, now embedded with parameters for the next layer, is passed back into the MAC units, through splitters. Devices such as electro-optic modulators (not depicted) may be used to implement non-linearities after the MAC operation. Unfortunately, the static nature of the hardware limits the size of the neural network model that can be accelerated using such a configuration. This configuration would also require a large number of splitters, which can cause increased optical losses and thus higher laser power requirement to compensate for the losses, as the size of an accelerator using this B&W configuration increases.

MRs and other on-chip optical resonators such as microdisks are crucial components in such non-coherent MAC configurations, as they impact the reliability and efficiency of the operation

performed. Figure 26(b) depicts an MR bank and its output spectrum along with the free spectral range (FSR). Factors such as fabrication-process variations (FPVs) and thermal variations which impact the MR critical dimensions and hence the effective refractive index (n_{eff}) of the device can cause a drift in the resonant wavelength ($\Delta\lambda_{MR}$) [139]. This drift can introduce errors into optical computation and is thus usually corrected with TO or EO tuning circuits. While EO offers faster tuning (~ns range) and consumes lesser power (~4 μm/nm), it also has a smaller tuning range [40]. TO tuning, on the other hand, consumes higher power (~27 mW/FSR) and has higher tuning latency (~μs range) [39], but offers a larger tuning range. Because of the larger correction capacity, TO is often preferred over EO despite its higher latency and power consumption. Therefore, as the number of MRs increases—when considering larger CNN or MLP models—the tuning power consumption also increases. This also creates increased wavelength requirements per waveguide and calls for longer waveguides to host the MRs, causing increased laser power consumption to supply the wavelengths and to compensate for the propagation losses in the longer waveguides. Also, more MRs and more wavelengths increase optical crosstalk, and also introduce thermal crosstalk due to the larger number of TO tuners employed. To counteract these challenges, and ensure better weight resolution, crosstalk mitigation strategies must also be considered.

To design an effective optical-domain BNN accelerator, all of these considerations must be taken into account. This highlights the need for (i) better device optimizations to tolerate variations; (ii) efficient and low-latency tuning mechanisms; (iii) a scalable architecture design, which is optimized for energy efficiency, area, and throughput. ROBIN, presented in this chapter, addressed all of these concerns for an efficient BNN accelerator implementation in the photonic domain.

4.3. BINARIZED NEURAL NETWORKS

BNNs [133] are types of DNNs (or CNNs) where both weights and activation parameters only use binary values, and the binary values are utilized during both inference and training using backpropagation. In the light of the discussion of various noises in photonic accelerator architectures, it is to be noted that the binary nature of weights in BNNs makes them resilient to small perturbations which can usually lead to gross classification errors in DNNs. Inspired by the seminal work on efficiently training BNNs [133], recent efforts either explore how BNN accuracy can be improved, apply BNNs to different application domains, or explore how BNNs can be implemented efficiently in hardware to leverage their low computation power and memory requirements in resource constrained environments.

BNNs utilize the sign function to convert real valued weights to +1 or -1. But this typically leads to complications in training as the gradient for the sign function always results in a zero. A heuristic called straight through estimator (STE), introduced in [140], can be used to circumvent this issue. STEs approximate the gradient by bypassing the gradient of the layer, by turning it into an identity function. The gradient thus obtained is used for updating real valued weights, using standard optimization strategies such as Adam or stochastic gradient descent (SGD). This process is utilized for activation parameters as well. Also, the use of batch normalization (BN) layers in BNNs has been shown to lead to several benefits [134]. The gain (γ) and bias (β) terms of the BN layer not only help condition the values during training, which speeds up BNN training, but also helps to improve accuracy in BNNs.

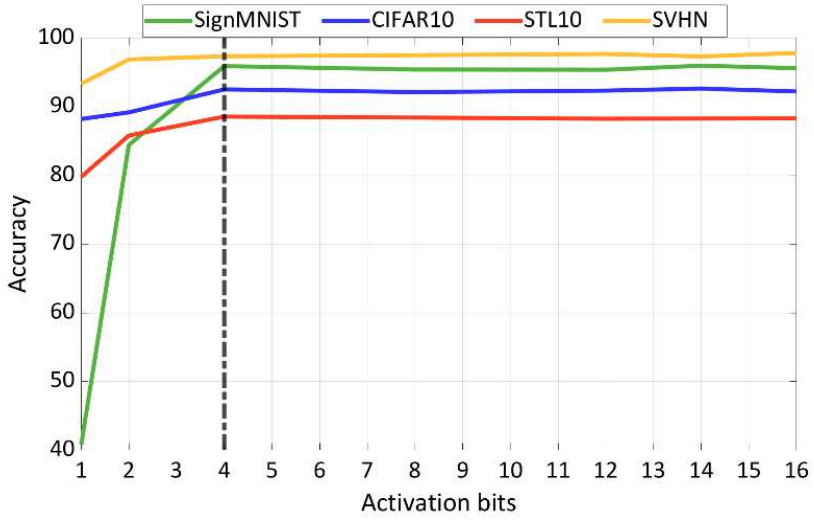


Figure 27. The accuracy sensitivity study conducted by varying activation parameter precision (number of bits). Weights are kept as binary values in all cases. The study was performed across four different models and their datasets (described later in Section 4.5).

Inference accuracy in BNNs can be increased by considering partially binarized BNNs, where selected layers have their parameters at higher precision. The last layer is usually not binarized to avoid severe loss in accuracy. With detailed analysis of the model, critical layers can be identified and can be kept at higher precision, for better accuracy, at the cost of increased resource (computation, memory) utilization. We conduct a BNN accuracy analysis to determine the appropriate activation parameter precision in considered models, which is required to determine the digital-to-analog converter (DAC) resolution in our accelerator architecture. In this analysis, weight paramters were restricted to binary (1-bit) values, but the bit precision level of the activations was altered from 1-bit to 16-bits. During BNN training, we ensured that we only binarize weights during the forward and backward propagations but not during the parameter update step, because keeping good precision weights during the updates is necessary for SGD to work at all (as parameter changes are usually tiny during gradient descent). After training, all weights were in binary format, while the precision of input activations was varied. Figure 27 shows

the results of varying activation precision across four different models and their datasets (described later in Section 4.5.1). We observed that the accuracy had notable change initially as activations bits were increased, but this gain in accuracy soon saturated. Based on the results, we consider binary (1-bit) weights with 4-bit activations, and thus use 4-bit DACs in our architecture.

4.4. *ROBIN* ARCHITECTURE

In this section, we describe the various optimization considerations at device, circuit, and architecture level used for designing the *ROBIN* architecture.

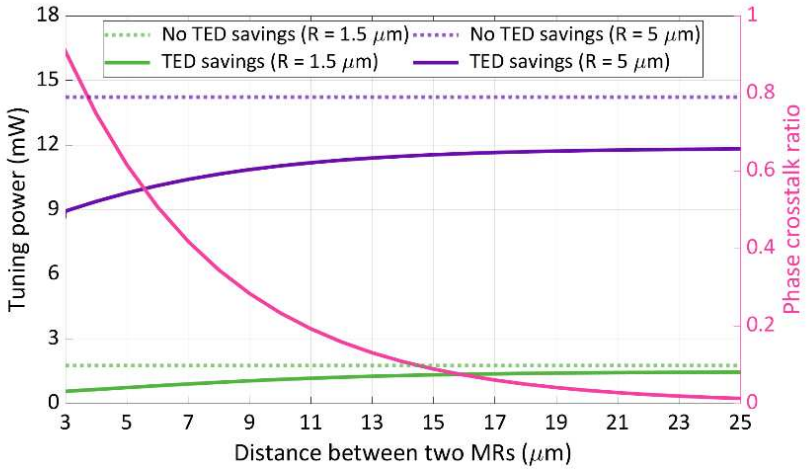


Figure 28. Tuning power compensation in a block of 10 MRs placed with and without considering thermal eigenmode decomposition (TED) for different MR radius. The orange line represents phase crosstalk ratio variation with distance between MRs.

4.4.1 TUNING CIRCUIT DESIGN

A tuning circuit design is essential for fast and accurate operation of MRs in our BNN accelerator. Presence of fabrication process variations (FPVs) can impact MR operations by altering their resonant wavelength (λ_{MR}) from the originally designed values. The errors caused by this shift can be significantly reduced by using an appropriate MR tuning circuit. The tuning

circuit employed can be either thermo-optic (TO) or electro-optic (EO) tuning circuits. Thermo-optic(TO)-based tuning mechanisms use microheaters to change the temperature in the proximity of a microring resonator (MR), which then alters the effective index (n_{eff}) of the MR. This in turn changes the λ_{MR} of the device. Such a change in resonant wavelength ($\Delta\lambda_{MR}$) can help compensate for fabrication-process and thermal variations in MRs. The electro-optic (EO)-based tuning mechanisms in an MR is based on the depletion and injection of carriers on a PN diode. However, only small shifts in an MR's resonant wavelength can be compensated using this mechanism (i.e., EO has a limited correction range). TO tuning is preferred to compensate for large shifts in MR's resonant wavelength. However, one has to compromise on latency (~µs range) and power consumption, which is higher than for EO tuning.

To reduce *ROBIN*'s reliance on TO tuning, which entails high overheads, the possibility of a hybrid tuning mechanism was explored. In this hybrid tuning mechanism, both TO and EO tuning are used to compensate for $\Delta\lambda_{MR}$. Such a tuning method has been proposed earlier [117] and can be easily transferred to an optimized MR (as discussed in Section 4.4.2) for hybrid tuning in our architecture. Such a mechanism would significantly reduce the overhead caused just by TO tuning.To reduce the power overhead of TO tuning in such a hybrid approach, we adapt a method called thermal eigenmode decomposition (TED), which was first proposed in [54] that involves collectively tuning all the MRs in an MR bank. By doing so we can cancel the effect of crosstalk (i.e., undesired phase shift) in MRs with much lower power consumption. The amount of phase crosstalk induced from one MR on another MR, placed adjacent to each other, can be modelled using the trend in Figure 28 (pink line). In this figure, as the distance between two devices (MRs) increases, the amount of phase crosstalk between them reduces. Correspondingly, as an example we calculate the tuning power compensation for an MR bank consisting of 10 MRs and different

radii placed at a distance (d) from each other. A few important trends to observe from Figure 28 are: (i) as the radius of an MR increases, tuning power compensation for $\Delta\lambda_{MR}$ increases;(ii) Without TED (collective tuning of MRs), the tuning power consumption is high, indicating that each MR would require more power to compensate for respective shifts in resonant wavelength ($\Delta\lambda_{MR}$); (iii) By employing TED, we see a significant reduction in tuning power consumption: 51% (radius of 1.5 μm) and 41% (radius of 5 μm) when MRs are placed at a distance of 5 μm and 7 μm apart from each other, respectively. Though placing MRs further close to each other would yield better compensation in power, one must take into account the placement and routing of tuning circuit for each MR in an MR bank. Additional power reduction can be obtained by performing device level optimizations, as designing MRs tolerant to FPVs would reduce the total power used to compensate for fabrication variations.

4.4.2 DEVICE-LEVEL OPTIMIZATION

We explore different MR designs to accommodate different needs in our *ROBIN* architecture such as multi-bit precision for activation values, single-bit precision for weight value representation, and batch normalization.

4.4.2.1 FABRICATION PROCESS VARIATION RESILIENCE

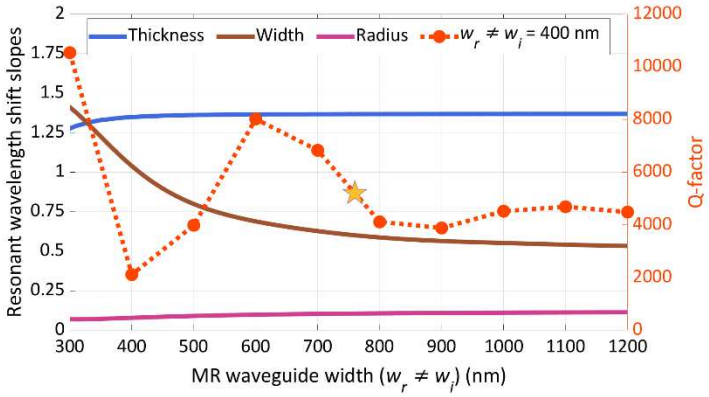
FPVs cause undesirable changes in device critical dimensions (e.g., width and thickness), which cause resonant wavelength shifts ($\Delta\lambda_{MR}$). To address $\Delta\lambda_{MR}$, we explore the impact of change in device parameters such as waveguide width, thickness, gap between input and ring waveguide, and radius using our in-house MR device-exploration tool. We map the behavior of different changes in the waveguide width, thickness, and radius in MRs due to FPVs. Figure 29(a) shows one of our design exploration results where we understand and observe the behavior of

resonant resonant-wavelength shift slopes due to change variations in the waveguide width, thickness and radius represented by orange, green and blue lines respectively. Resonant wavelength shift slope due to change in waveguide width ($\partial\lambda MR/\partial w$) can be given as:

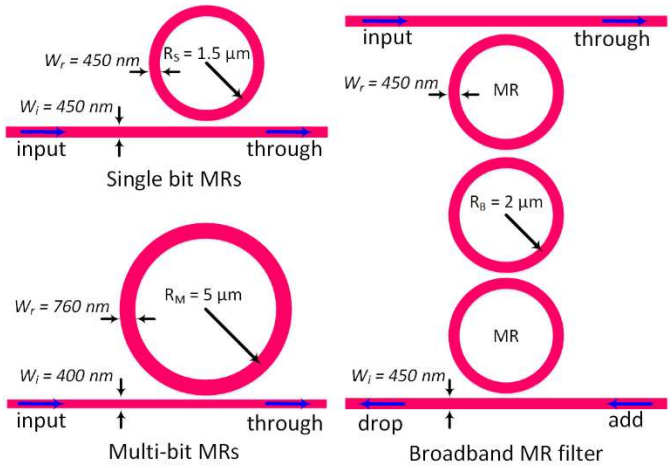
$$\frac{\partial \lambda_{MR}}{\partial w} = \left| \frac{\left(\Delta\lambda_{MR}(\lambda, w + \epsilon_W, t, R) - \Delta\lambda_{MR}(\lambda, w - \epsilon_W, t, R)\right)}{2\epsilon_W} \right| \quad (15)$$

In (15), ϵ_w denotes a small change in waveguide width and $\Delta\lambda_{MR}$ depends on changes in width (w), thickness (t), and radius (R). Similarly, $\partial\lambda MR/\partial(t, R)$ can also be approximated.

Figure 29(a) clearly shows that the impact of resonant-wavelength shift reduces as we increase the waveguide width, whereas the impact of thickness and radius variations remains constant. From the conducted experiemnts, $\Delta\lambda_{MR}$ is more sensitive to changes in waveguide width, hence the impact of $\Delta\lambda_{MR}$ reduces as the waveguide width is increased. We employ Lumerical MODE [141], an Eigen mode solver to calculate these shifts in resonant wavelengths. One can easily overcome higher-order mode excitation by employing adiabatic designs and waveguide tapers [142] in MRs with wider waveguides. Such a design translates to lesser tuning-power consumption due to FPVs.



(a)



(b)

Figure 29. (a) Resonant-wavelength shift slopes with respect to changes in waveguide width, thickness, and radius, and corresponding cross-over coupling(κ), when the input waveguide (w_i) is set to 400 nm the marked point represents our selected MR desig (b) The different MR designs considered in this work.

4.4.2.2 MULTI-BIT PRECISION MICRORING RESONATORS

As discussed in Section 4.3, increasing the number of bits used to capture activations in a model can boost the model accuracy in BNNs. However, we observed that there is not a significant accuracy boost beyond 4-bit activation values, hence, we explore MR designs which can achieve a resolution of 4 bits. To achieve a resolution of 4 bits, we have to take into consideration how the

optical signals from MRs impact each other due to crosstalk. We consider calculations from [131] to define the amount of noise from one MR on the other:

$$\phi(i,j) = \frac{\delta^2}{\left(\lambda_i - \lambda_j\right)^2 + \delta^2}, \tag{16}$$

where, $\phi(i-j)$ describes the noise content from the j^{th} MR present in the signal from the i^{th} MR, $(\lambda_i - \lambda_j)$ is the difference between the resonant wavelengths (λ_i, λ_j), and $\delta = \lambda/(2 \cdot Q - factor)$. Where, Quality factor or Q-factor is a measure of the sharpness of the resonance relative to the central frequency of a microring resonator (MR) that impacts the optical channel spacing, crosstalk, bandwidth, and other factors in the MR [116]. A sharper resonance (i.e., a higher Q-factor) can result in increased susceptibility to noise, as even a small change in the central frequency of the MR (due to perturbance) can lead to large losses. This limits the achievable resolution of the parameters being represented. Thus, smaller Q-factors are preferred. However, too small a Q-factor can also lead to larger device dimensions and higher optical crosstalk, which in turn can lead to larger losses and higher tuning power requirements. Q-factor in an MR is defined as follows:

$$Q - factor = \frac{\lambda_{MR}}{FWHM}, \tag{17}$$

where, FWHM is the full width at half maximum of a resonance spectrum which can be defined for an all-pass ring resonator (see Figure 29) as follows:

$$FWHM = \frac{(1 - ra)\lambda_{MR}^2}{\pi n_g L \sqrt{ra}}, \tag{18}$$

where, r is the self-coupling coefficient and a is the single-amplitude transmission, including both the propagation loss in the ring and the loss in the couplers; this can be written as $a = e^{-\alpha L}$, where α is power attenuation coefficient. L is round trip length or the circumference of the MR. In this chapter, we assume a lossless coupler in our designed MRs, hence $|\kappa|^2 + |r|^2 = 1$. Where κ is the cross-over coupling coefficient. For ideal cases with zero attenuation, $a \approx 1$. Based on the above equations, the noise power component can thus be calculated as:

$$P_{noise} = \sum_{j}^{(n-1)} \phi(i,j) P_{in}[i] \tag{19}$$

For power intensity (P_{in}) of 1, the resolution can be computed as:

$$Resolution = \frac{1}{max|P_{noise}|}, \tag{20}$$

To achieve a bit resolution of at least 4-bits, we need MRs with a Q-factor of ≈5000 (from (20)) while being tolerant to FPVs. Q-factor is highly sensitive to losses and change in dimensions of MR. In order to achieve the specific Q-factor value, we select the following MR dimesnions: input waveguide width of 400 nm and ring waveguide width of 760 nm, and radius (RM) of 5 μm. This MR design, as shown in Figure 29(a) (magenta line), provides improved tolerance to FPV, desirable Q-factor, and smaller area consumption. Such an MR design with Q-factor of 5000, allows enough levels of distinction between bits by slightly changing intensity, helps easily detect optical signal at the output port satisfying the requirement for multi-bit precision of activation values.

4.4.2.3 SINGLE-BIT MICRORING RESONATORS

In our architecture, we represent weight values with a single bit, and this requires just two levels of precision with the output signal from an MR. An MR of high Q factor may be used here, as we don't have to have high resolution here. Compact ring designs with high Q-factor have been proposed in [143], [144]. The work in [144] proposes an MR design with radius 1.5 μm to achieve a high Q-factor of 46,000 without the consideration of sidewall roughness while maintaining low bending loss ≈7 cm^{-1}. Similarly, an adiabatic MR structure of radius 3 μm is designed in [143] to avoid higher order mode excitation where a high Q-factor of 27,000 is achieved. These works indicate that such high Q-factor rings can be designed.

For one-bit weight representation in *ROBIN*, we design a ring of radius 1.5 μm, as shown in Figure 29(b), with input waveguide (w_i) and ring waveguide (w_r) width both set to 450 nm, to achieve a Q-factor of 25,000 that corresponds to a bit resolution of 1 from (6). These designs allow our architecture to save on area and tuning power consumption. We acknowledge that FPVs are an inevitable part of the fabrication process. However, since we just need to differentiate between two levels of operations, we do not explore for designs that are tolerant towards FPVs, for single-bit MRs.

4.4.2.4 BROADBAND MICRORING RESONATORS

Batch normalization (BN) layers can be considered essential in BNNs as they add complexity to the models, via the gain (γ) and bias (β) terms of the layer. These terms are learned during the training process along with the normalization parameters of the batch mean (μ) and standard deviation (σ). During the training phase, these terms are dynamic, but during inference they have

static values. This allows for a hardware implementation of a photonic version of batch normalization folding, where we may tune weights as per the following equation:

$$w_{fold} = \gamma \cdot \frac{W}{\sqrt{\sigma^2 + \epsilon}} = C_{fold} \cdot W \tag{21}$$

There is a similar equation for bias terms as well, but since BNN models benefit from batch normalization after every layer, these will be normalized out and hence can be ignored. The above constant, C_{fold} is applied to every weight term and hence is a participant in every matrix multiplication operation, i.e.:

$$Input_{l+1} = f\left(A_l \cdot \left(w_{fold}\right)_l\right) = C_{fold} \cdot f(A_l \cdot W_l) \tag{22}$$

In (22), $Input_{l+1}$ refers to the input to the $(l + 1)^{th}$ layer, $f()$ is non-linear activation function, A_l is the activation of l^{th} layer and W_l is the weights from l^{th} layer. This operation can be applied to partial sums as well, and can be implemented using a broadband photonic device with its gain tuned to reflect C_{fold}.

For implementing the photonic batch normalization, a broadband device is preferred as this allows simultaneous gain tuning of all the wavelengths in the waveguide efficiently, both area and energy wise. Hence, the last type of MRs we consider are broadband MRs that are needed for batch normalization (BN) layers due to their relevance in BNNs. A large passband can be achieved by cascading several MRs and properly selecting the design parameters of MRs [145]. We explore such a higher order MR, or cascaded MR filter, to achieve a wide passband. The work presented in [146] explores a possibility for passband widths ranging from 6.25 Ghz to a maximum of 3 Thz. This work explores different design parameters of a higher-order filter while evaluating different losses such as insertion, propagation and coupling loss in higher order MRs. A 0.5 nm resonant

wavelength shift of MR was reported for a fabrication error of 10 nm showing that such a design is tolerant to FPVs.

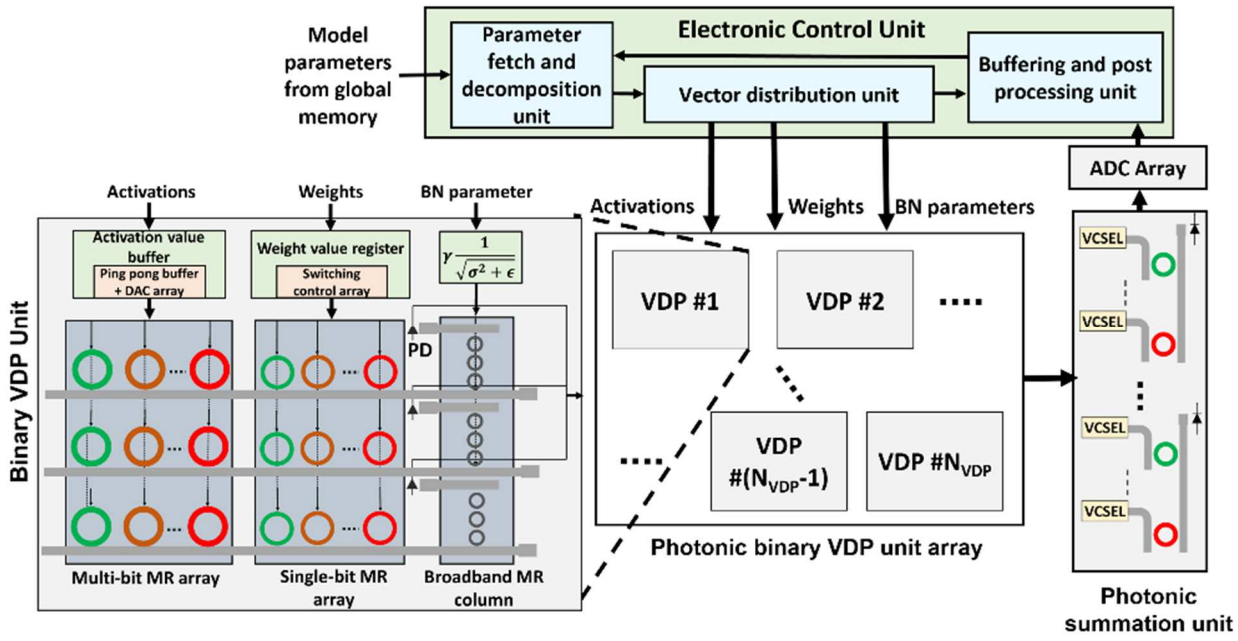


Figure 30. An overview of the ROBIN architecture, showing the electronic control unit, the photonic vector dot product (VDP) unit array, and the photonic summation unit, along with a detailed view of the VDP unit internal structure.

A 3rd-order MR based switching device with radius of 2 μm shown in Figure 29(b), fits the requirement for broadband MR. The coupling coefficients at the input (κ_i^2) is 0.53 and coupling at higher order rings is 0.2. The propagation loss of 25 dB/cm has been reported and insertion loss of the two elements in higher order filter are 4.35 dB and 0.36 dB, respectively [145]. Having such a design, one can achieve a flat-top passband with bandwidth width of at least 3 THz. Employing this broadband MR can help us apply the batch normalization parameter C_{fold} on all the available resonant wavelengths in the bank. Having a large bandwidth such as 2.5 Thz allows us to conveniently tune up to 20 different wavelengths.

4.4.3 ARCHITECTURE DESIGN

An overview of the *ROBIN* accelerator architecture is shown in Figure 30. The optical device and tuning circuit optimizations from the previous subsections are utilized within the optical binary vector dot product (VDP) units. We use banks of heterogeneous MRs (described in sections 4.4.2) to imprint activation parameters, weights, and the BN layer constants onto optical signals. Multiple such VDP units are composed together to form the overall architecture, as shown in the figure, which is then used to accelerate a given BNN model. We utilize a photonic summation unit for summing the partial sum outputs from our VDPs, before passing the partial sums on to the electronic control unit (ECU), as shown in Figure 30. We also rely on the ECU for fetching parameters from the global memory, decomposing them to lower dimensional vectors, distributing these vectors among the VDP units, and for implementing non-linear activations functions and pooling layers. We describe the working of the *ROBIN* architecture in more detail in the following subsections.

4.4.3.1 DECOMPOSING VECTOR OPERATIONS

To map convolution (CONV) and fully connected (FC) layers from BNN models to our accelerator, we first need to decompose large vector sizes into smaller ones, so they can be mapped to the VDP array in our architecture. This decomposition approach can be explained as follows.

In CONV layers, a filter performs convolution on a patch (e.g., 2×2 elements) of the activation matrix in a channel to generate an element of the output matrix. The operation can be represented as:

$$K \otimes A = Y. \tag{23}$$

Assuming a 2×2 filter kernel and weight matrices, (23) can be rewritten as:

$$\begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix} \otimes \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} = k_1 a_1 + k_2 a_2 + k_3 a_3 + k_4 a_4, \tag{24}$$

Rewriting (24) as a vector dot product, we have:

$$[k_1 \quad k_2 \quad k_3 \quad k_4] \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = k_1 a_1 + k_2 a_2 + k_3 a_3 + k_4 a_4, \tag{25}$$

Once we can represent the operation as a vector dot product, it is easy to see how it can be decomposed into partial sums. For example:

$$[k_1 \quad k_2] \cdot \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = k_1 a_1 + k_2 a_2 = PS_1, \tag{26a}$$

$$[k_3 \quad k_4] \cdot \begin{bmatrix} a_3 \\ a_4 \end{bmatrix} = k_3 a_3 + k_4 a_4 = PS_2, \tag{26b}$$

$$PS_1 + PS_2 = Y. \tag{26c}$$

In FC layers, typically much larger dimension matrix-vector multiplication operations are performed between input activation vectors and weight matrices. Therefore, we have:

$$A \cdot W = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \cdot [w_1 \quad w_2 \quad \cdots \quad w_n], \tag{27}$$

$$A \cdot W = \begin{bmatrix} a_1 \cdot w_1 + a_1 \cdot w_2 + \cdots + a_1 \cdot w_n \\ a_2 \cdot w_1 + a_2 \cdot w_2 + \cdots + a_2 \cdot w_n \\ \vdots \\ a_n \cdot w_1 + a_n \cdot w_2 + \cdots + a_n \cdot w_n \end{bmatrix}. \tag{28}$$

In (27), a_1 to a_n represent column vectors of activations (A) and w_1 to w_n represent row vectors of weight matrix (W). The resulting vector is a summation of dot products of vector elements (28). Similar to the decomposition of CONV operation, these can then be decomposed into lower dimensional dot products.

4.4.3.2 VECTOR DOT PRODUCT (VDP) UNIT DESIGN

As discussed in subsection 4.4.3.1, we decompose matrix operations to lower dimensional vector dot product operations. These vector dot product operations are executed optically within our VDP units. The heterogeneous MR designs combined with optical circuit-level optimizations for area and power consumption are utilized to design VDP units (Figure 30) suited for accelerating both CONV and FC layers without compromising on accelerator throughput. For representing weight values, we use high Q-factor, small radius single-bit MRs described in Section 4.4.2.2. The smaller radius contributes to lower tuning power and helps reduce propagation loss along the VDP waveguide. This is possible due to the binarized nature of weight matrices in BNNs. For activation values we consider MRs with slightly lower Q-factor, for better resolution, as discussed in Section 4.4.2.1. Optical BN layer implementation require simultaneously tuning all the wavelengths in the waveguide to the batch normalization constant, and for this we use third order MR filters, as described in Section 4.4.2.3. The combination of these heterogeneous designs allows the VDP units to be highly energy efficient. We also make use of electronic buffering in the VDP units to reduce the digital to analog converter (DAC) usage. In particular, we make use of ping-pong buffers, which allow us to use a single DAC array to feed the activation devices in all the waveguides in a VDP unit. As weight values are single-bit values, we can use simple switching circuits to essentially turn the MR tuning circuits on or off depending on the value of the weight parameters.

In designing a VDP unit, there are several important parameters that must be carefully considered: number of higher resolution MRs for activation representation (N_A), number of single-bit MRs for weight representation (N_W), and number of broadband MRs (N_B) for batch normalization folding implementation. Thus, the total number of MRs per waveguide NMR = $N_A + N_W + N_B$. The number of required DACs is equal to N_A. By the mathematical property of the dot product operation, N_A must be equal to N_W. The number of waveguides to which we distribute the MRs is denoted as N_{WG}. The maximum size of the vector that can be represented in a VDP unit is given by $N_{WG}*N_A$. We divide this vector across multiple waveguides to reduce power consumption, as this allows us to reuse wavelengths and reduce the overall laser power consumption, as discussed next, in subsection 4.4.3.3. Multiple VDP units work concurrently on parameters from the same layer and generate partial sums simultaneously, for efficient parallelization and to increase the throughput of the accelerator. The total VDP unit count used in *ROBIN* is N_{VDP}. Thus, the VDP and architecture design process can be considered as an optimization problem where we try to explore N_{VDP}, N_{WG}, N_A (= N_W), and NB values while trying to maximize throughput and minimize area and power consumption. We present results of this architecture exploration analysis in Section 4.5.3.

4.4.3.3 OPTICAL WAVELENGTH REUSE IN VDP UNITS

Prior works on optical accelerator design typically considers a separate wavelength to represent each individual element of a vector. As the size of the vectors being mapped increase, this approach leads to an increase in the total number of lasers needed in the laser bank, which in turn increases power consumption. Beyond employing the decomposition approach discussed above, we also consider wavelength reuse per VDP unit to minimize laser power. In this approach, within VDP units, the vectors assigned from the electronic control unit (ECU) are further decomposed into

smaller sized vectors for which dot products can be performed using MRs in parallel, in each arm of the VDP unit. By decomposing the mapped vectors further, same wavelengths can be reused across arms within a VDP to reduce the number of unique wavelengths required from the laser. Photodetectors (PDs) perform summation of the element-wise products to generate partial sums from decomposed vector dot products. The partial sums from the decomposed operations are then converted back to the optical domain by VCSELs (bottom right of Figure 30), multiplexed into a single waveguide, and accumulated using another PD, before being sent for buffering. Thus, our approach leads to an increase in the number of PDs and splitters compared to other accelerators but significantly reduces both the number of MRs per waveguide and the overall laser power consumption. The reduction in overall power consumption is also assisted by the fact that PDs do not consume significant power.

In each arm within a VDP unit, we can use a maximum of 15 MRs per bank for a total of 30 MRs per arm. The choice of MRs per arm considers not only the thermal crosstalk and layout spacing issues and the benefits of wavelength reuse (as discussed earlier), but also the fact that optical splitter losses become non-negligible as the number of MRs per arm increase, which in turn increases laser power requirements. Thus, the selection of MRs per arm within a VDP unit must be carefully adjusted to balance parallelism within/across arms, and laser power overheads.

4.4.3.4 *ROBIN* PIPELINING AND SCHEDULING

The pipeline and schedule of operations during BNN model execution on the *ROBIN* accelerator is shown in Figure 31. The electronic control unit (ECU) for the accelerator communicates with the global memory and retrieves the trained weights for the model being accelerated. The weights are stored in SRAM-based buffers. Considering the vector granularity of the VDP units, latency of operation of the photonic core, and the parameter sizes (4-bit activation

bits and binary weight parameters), we can calculate the memory bandwidth necessary. From our analyses (presented in Section 4.5.3), we found that our architecture needs a maximum bandwidth of 93.75 GB/s at the ECU to photonic core interface. This is a reasonable bandwidth assumption for an SRAM-based memory with operating frequency $\geq$ 2.5 GHz and a read width of 250 bits. Previous works, such as [147], have explored similar SRAM systems, but for a much higher bandwidth requirement at 250 GB/s. The lower bandwidth requirement for our system can be attributed to the smaller parameter sizes, while the work in [147] considered 16-bit precision for the neural network parameters. Memory interfaces which exceed the necessary bandwidth are already available commercially: e.g., NVIDIA Tesla K20M GPUs have 320-bit memory interfaces at 2.6 GHz which can operate every half clock cycle to provide a bandwidth of 208 GB/s.

These weight matrices are decomposed to lower dimensional vectors and are distributed to the VDPs by the ECU's vector decomposition unit. The decomposition operation is described by the left-hand side of equations 10 to 12. As described in the equations, the vector decomposition unit converts matrices to vectors (row-wise conversion for weight matrices and column-wise conversion for activation matrices), and then those vectors into sub vectors. The size of the sub vectors depends on the granularity of the VDP units. The received vectors are buffered in the VDP units and are fed into the DAC array through a ping-pong buffer so that they can keep the MAC operation running continuously. The partial sums generated are passed on to the photonic summation unit, the output from which is passed on to the ECU. The ECU buffers the sums and calculates inputs that are then passed on to the next layer by subjecting the parameters to non-linearities (activation functions) and performing other layer specific operations, like pooling.

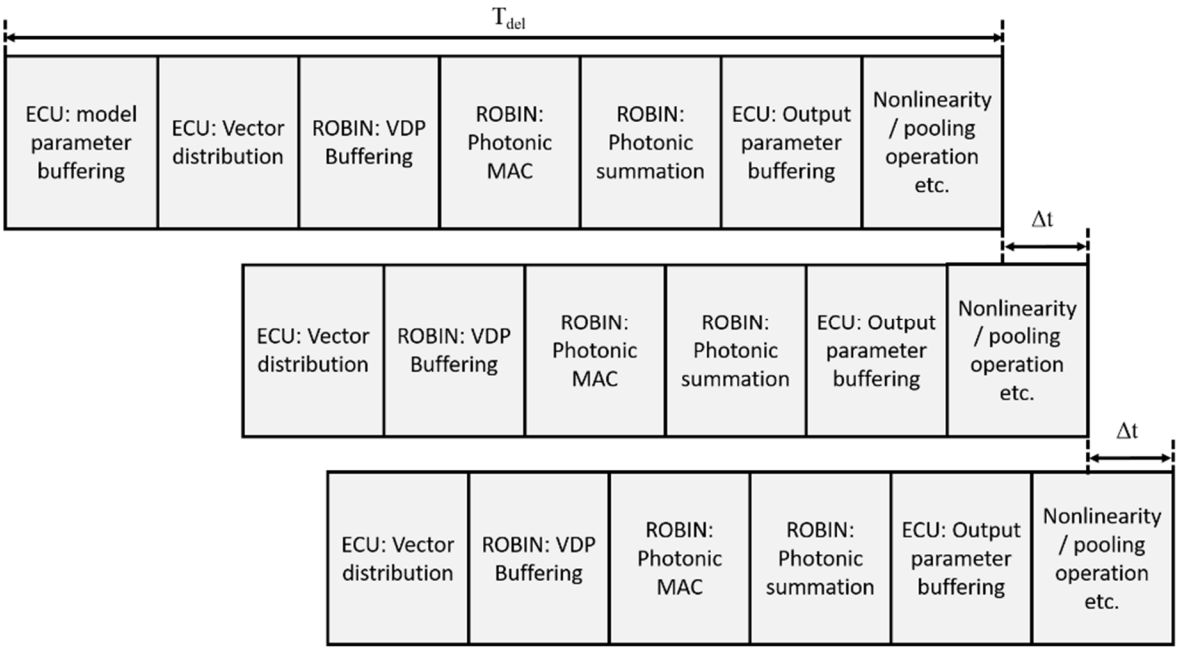


Figure 31. Pipelined scheduling of operations during BNN execution on the ROBIN accelerator.

The model parameter buffering stage is not repeated every pipeline operation, but must be repeated as the parameters buffered in the buffers in ECU are depleted (i.e., distributed to VDP units). As such, the total time required by *ROBIN* to perform inference acceleration for a given model can be given as:

$$Total\ time\ of\ operation = T_{del} + \Delta t \times X + (ECU\ parameter\ buffering\ delay) \times x, \quad (29)$$

where,

$$\Delta t = local\ buffer\ operation\ delay + vector\ distribution\ delay, \quad (30)$$

$$X = \frac{Total\ number\ of\ parameters\ in\ the\ model}{N_w \times N_{VDP}}, \quad (31)$$

$$x = \frac{(Parameters\ buffered\ in\ ECU)}{N_w \times N_{VDP}}. \quad (32)$$

Table 9 Models and datasets used for evaluations

Model no.	CONV layers	FC Layers	BN layers	Parameters	Datasets
1	2	2	3	60,642	Sign MNIST
2	6	3	6	1,546,570	CIFAR10
3	6	3	7	13,570,186	STL10
4	6	2	6	552,362	SVHN

Comparing our pipeline to the pipeline presented in the previous work on photonic BNN acceleration, [138], we can observe the following differences: (i) *ROBIN*'s pipeline takes into consideration model parameter retrieval from global memory, buffering in the ECU, and how these parameters are utilized in the photonic core. The pipeline in [138] does not include these operations in its pipeline; *ROBIN*'s pipeline considers both ECU and photonic core operation, whereas the pipeline in [138] is photonic system centric; *ROBIN* utilizes photonic batch normalization folding which does not require an extra step, whereas in [138] this operation is performed electronically and requires a separate stage in their pipeline.

4.5 EXPERIMENTS AND RESULTS

4.5.1 SIMULATION SETUP

Several simulation studies were conducted to evaluate the effectiveness of the *ROBIN* BNN accelerator. The optimized heterogeneous MR designs, the tuning circuit optimizations, and architectural level considerations discussed so far were included in our simulation considerations.

The operation of the *ROBIN* architecture was simulated using a custom Python simulator to estimate its performance in terms of power, frames per second (FPS) performance, and energy consumption. For analyzing the inference accuracy across different activation precision and the

impact of FPV noise on the inference accuracy, we used Tensorflow 2.3 along with Qkeras [121]. Figure 32 shows the training accuracy versus epoch graph of the models described in Table 9, to illustrate the accuracy and loss across the epochs.

We compare *ROBIN* with DEAP-CNN [110] and HolyLight [111], two recent optical DNN accelerators from prior work, along with LightBulb [138], which is an optical BNN accelerator, as well as numbers reported from several electronic DNN and BNN accelerators. For simulating the operation of optical accelerators, we considered optical signal losses due to various factors: signal propagation loss (1 dB/cm [135]), splitter loss (0.13 dB [122]), combiner loss (0.9 dB [123]), MR through loss (0.02 dB [124]), MR modulation loss (0.72 dB [125]), microdisk loss (1.22 dB [126]), EO tuning loss (6 dB/cm [40]), and TO tuning loss (1 dB/cm [39]). We also considered the ADC design from [148] and the 4-bit DAC from [149] in our analyses. The analysis of the optical accelerators (DEAP-CNN [110], HolyLight [111], and LightBulb [138]) follows the modeling methodology we have adopted for *ROBIN*, where we factor in power consumption and delays associated with photonic devices used in these accelerators. A summary of the power and latency considerations for our analyses is given in Table 10. These power and latency values were used in our simulations and latency of operation of our architecture. In order to give better perspective on the architecture's performance, a comparison for inference time on *ROBIN* and a conventional CPU is presented in Section 4.5.5.

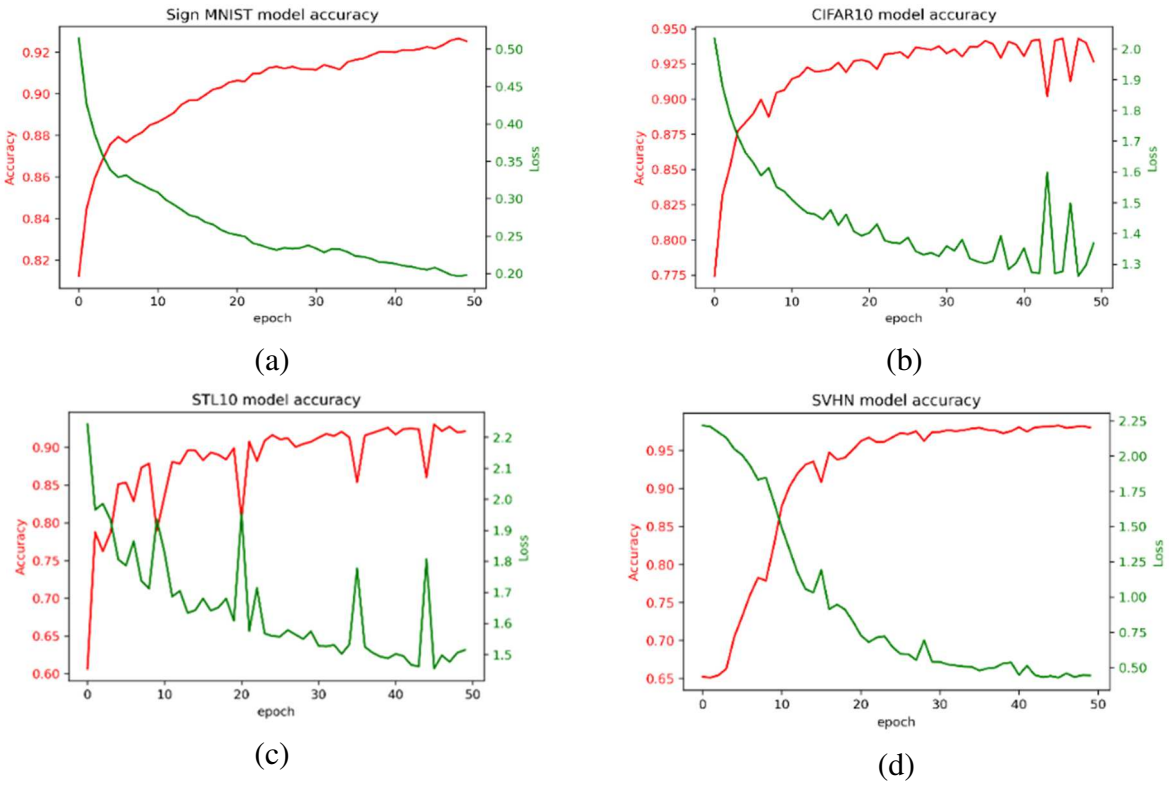


Figure 32. The training accuracy vs epoch for the BNN models considered for (a) Sign MNIST, (b) CIFAR10, (c) STL10, and (d) SVHN datasets. (a) shows top-1 accuracy, while (b)-(d) show top-5 accuracy

To calculate laser power consumption, we use the following power model:

$$P_{laser} - S_{detector} \geq P_{photo-loss} + 10 \times \log_{10} N_{\lambda}\,, \tag{33}$$

where P_{laser} is the laser power in dBm, $S_{detector}$ is the PD sensitivity in dBm, and $P_{photo-loss}$ is the total loss encountered by the optical signal, due to all of the factors discussed above.

Table 10 Parameters considered for analysis of photonic accelerators

Devices	Latency	Power
EO Tuning [40]	20 ns	4 μW/nm
TO Tuning [39]	4 μs	27.5 mW/FSR
VCSEL [128]	10 ns	0.66 mW
TIA [129]	0.15 ns	7.2 mW
Photodetector [130]	5.8 ps	2.8 mW
DAC [149]	0.33 ns	59.7 mW
ADC [148]	24 ns	62 mW

4.5.2 FABRICATION-PROCESS VARIATION ANALYSIS

FPV in optical devices is corrected using TED tuning in our architecture, as discussed in Section 4.4. At the system level, this tuning leads to significant power consumption overhead, and any avenue to further reduce tuning power consumption becomes important. We conduct an FPV noise injection analysis, where we inject noise, modeled using FPV data, into the MR devices into our *ROBIN* accelerator, during the inference phase. This experiment was conducted to: *(i)* study the impact of FPV induced noise on BNN models mapped to our accelerator; *(ii)* determine how effective TED tuning is in such scenarios; and *(iii)* uncover any opportunities for further power minimization.

To analyze the impact of FPV on the model and how TED tuning compensates for it, we first consider the effect of FPV on the shift in resonant wavelength ($\Delta\lambda_{MR}$) in MRs. Resonant-wavelength shift in an MR can be modeled from [150] as:

$$\Delta\lambda_{MR} = \frac{\partial\lambda_{MR}}{\partial w}\sigma_w + \frac{\partial\lambda_{MR}}{\partial t}\sigma_t + \frac{\partial\lambda_{MR}}{\partial R}\sigma_R, \quad (34)$$

where, $\sigma_{w,t,R}$ are the associated standard deviations for waveguide width, thickness, and radius variations; and $\frac{\partial\lambda_{MR}}{\partial(w,t,r)}$ is the rate of change in the MR resonant wavelength considering the variations in the waveguide width, thickness, and radius represented in (34). We generate virtual FPV maps for the accelerator layout with a mean (μ) of 0 and standard deviation ($\sigma_{(w,t,R)}$) of 4.9 nm, 1.5 nm, and 0.75 nm for waveguide width, thickness, and radius, respectively. These standard deviation values are experimentally obtained based on real fabricated MR devices through our collaboration with CEA-Leti. Using these values, we are able to derive $\Delta\lambda_{MR}$ using (34). So, the current resonant wavelength (λ'_{MR}) of the FPV affected MR becomes:

$$\lambda'_{MR} = \lambda_{MR} + \Delta\lambda_{MR}, \quad (35)$$

Due to a shift in λ_{MR}, the transmission of the wavelength through the MR is impacted. The intensity of the wavelength at the through port is given by the following equation from [143].

$$T = \frac{I_{out}}{I_{in}} = \frac{a^2 - 2racos\phi + r^2}{1 - 2arcos\phi + ra^2}, \quad (36)$$

In (36), $\phi = \beta L$, with L being the roundtrip length and β the propagation constant $\beta = 2\pi/\lambda$ of the circulating mode; and r^2 is the self-coupling coefficient of an MR. A detailed analyses for the calculation of r using super mode theory is presented in [151]. The output intensity from the MR is important, as for non-coherent MAC units, the parameter values are encoded onto the signal intensity, and a change in expected output can be seen as perturbation or noise source.

The noise injection was modeled using equations (34) and (36), where we consider the resonant-wavelength shift ($\Delta\lambda_{MR}$) in MRs due to FPV and its impact on the parameters imprinted on the MRs. From our analysis using the FPV data from our device fabrications with CEA-Leti,

and equation (18), we are able to obtain the mean and standard deviation values for $\Delta\lambda_{MR}$ in a wafer. The values calculated are μ = -0.1461 nm and σ = 24.417 nm. Using these values, 50 $\Delta\lambda_{MR}$ maps for the accelerator were generated and then using equation (36) the perturbation to the parameters imprinted on to the devices were modeled. Noise injection to the models was performed at inference time using Tensorflow.

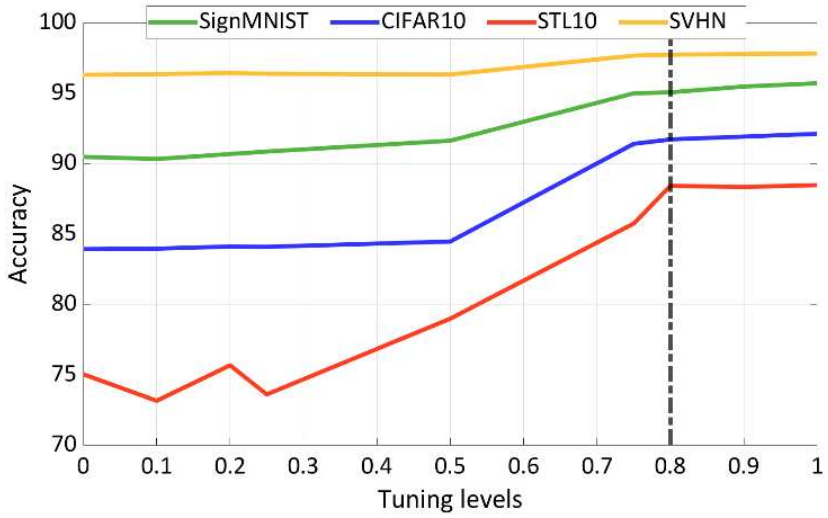


Figure 33. Inference accuracy versus level of tuning applied. At 80% tuning, the inference accuracy saturates, rendering further tuning unnecessary, and providing an opportunity to save tuning power.

Figure 33 shows the results of this experiment, where we explored the impact of FPV-induced noise in the four BNN models, and the effect of TED tuning for FPV compensation. We expected that the better the devices were tuned, the better the accuracy that would be exhibited by the accelerator. But it was observed that the model's accuracy can be sustained without perfectly tuning the devices. Figure 33 shows that at 80% FPV correction through tuning, the BNN retains appreciable inference accuracy. Thus, there is not a significant accuracy benefit to tune beyond the 80% level, this allows for a 20% reduction in tuning power requirement. This reduction in tuning power is factored into our architecture level analysis, which is presented next.

4.5.3 ROBIN ARCHITECTURE OPTIMIZATION ANALYSIS

In this section, we show results of our exploration of the parameters discussed in Section 4.4.3.2. As mentioned in Section 4.4.3.2, we try to optimize N_{VDP}, N_{WG}, N_A, and N_B to reduce area and power consumption while trying to obtain the best throughput (frames per second or FPS) possible. N_B was fixed to be 1 per waveguide, allowing us to have up to 20 wavelengths in the same waveguide with a channel spacing of 1 nm, which in turn allows us to tune all the MRs simultaneously to the BN layer parameters. We then explored N_{VDP}, N_{WG}, and N_A, with the goal of optimizing power, area, and FPS. The result of this exploration analysis is shown in Figure 34 in the form of a scatter plot. From this analysis we identified two configurations for *ROBIN*, where one is optimized for FPS/Watt, with lowest area and power consumption (energy optimized *ROBIN* or *ROBIN-EO*), and another with the best FPS but with higher area and power consumption (performance optimized *ROBIN* or *ROBIN-PO*). In terms of (N_A, N_{VDP}, N_{WG}), these configurations can be represented as (10, 50, 10) for *ROBIN-EO* and (50, 200, 10) for *ROBIN-PO*. These configurations were compared against other optical and electronic DNN/BNN accelerator platforms, to showcase their efficiency of operation. Results for these comparisons with other accelerators are presented in the following section.

4.5.4 COMPARISON WITH STATE-OF-THE-ART OPTICAL AND ELECTRONIC DNN/BNN ACCELERATORS

We compared *ROBIN-EO* and *ROBIN-PO* against various electronic and optical neural network acceleration platforms. For optical DNN accelerator platforms, we selected DEAP-CNN [110] and HolyLight [111]. The electronic accelerator platforms considered are: GPU (Nvidia Tesla P100), SIGMA [152], Edge TPU [153], DaDianNao [154], and FPGA implementation of

Null Hop [155]. We also compare *ROBIN* against the best-known previous photonic BNN accelerator, LightBulb [138].

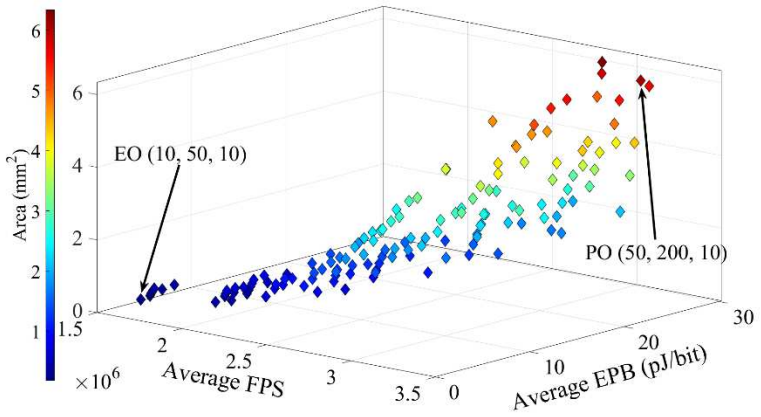


Figure 34. Scatterplot of average FPS vs. average EPB vs. area of various ROBIN configurations. The configuration with highest FPS/Watt (energy optimized or EO) and the one with best FPS (performance optimized or PO) are specified.

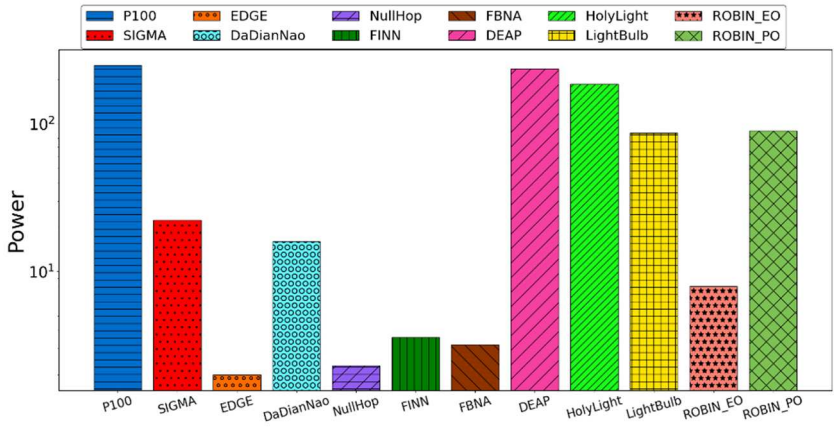


Figure 35. Power consumption comparison among variants of ROBIN versus other optical accelerators (DEAP-CNN, Holylight, LightBulb), and electronic accelerator platforms (P100, SIGMA, EdgeTPU, DaDianNao, Null Hop, FINN, and FBNA).

When compared to LightBulb, *ROBIN* has the following differences:

(i) *ROBIN* is designed to accelerate partially binarized neural networks, as opposed to fully binarized neural networks as in LightBulb, for obtaining better accuracies;

(ii) *ROBIN* utilizes photonic batch normalization folding for faster, energy efficient batch normalization layer operation whereas LightBulb relies on an electronic implementation of the batch normalization operation;

(iii) *ROBIN* has various circuit- and device-level optimizations in place to counteract thermal and process variations, which also ensure high throughput and energy efficient operation; whereas LightBulb does not take into account thermal and process variations and the necessary tuning latency and energy consumption overheads needed to counter them;

(iv) architecture-level optimizations in *ROBIN* ensure lower power consumption in terms of tuning and laser power; these considerations are not part of the architecture proposed in LightBulb.

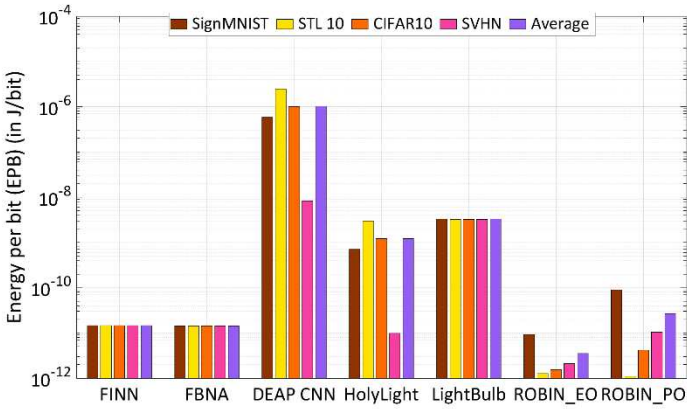


Figure 36. EPB comparison between electrical BNN accelerators, optical accelerators, and the ROBIN variants.

We also compare against electronic BNN accelerators FBNA [156] and FINN [157].

Figure 35 shows the power comparison across the accelerators from prior work and the two *ROBIN* variants. It can be observed that *ROBIN-PO* has substantially higher power consumption than *ROBIN-EO*, as *ROBIN-PO* is focused on FPS performance rather than energy conservation. *ROBIN-PO* has a much larger vector granularity per VDP unit along with substantially higher VDP unit count to maximize parallelism, when compared to *ROBIN-EO*. The larger unit count and the waveguide count in *ROBIN-PO* drives its power requirements higher. On the other hand, it can be observed that the energy and area efficient *ROBIN-EO* has comparable power consumption to that of edge and mobile electronic neural network accelerators.

In Figure 36, we compare the energy-per-bit values (EPB) across the various BNN accelerators considered in this work. We can observe that both the *ROBIN* variants perform significantly better than the optical accelerators in comparison. This lower EPB is owing to the meticulous device, circuit, and architecture level optimizations we have considered in our architecture, which takes into account various losses and delays at the architecture level and counteracts them. The heterogeneous MRs used in *ROBIN* provide energy and area benefits, and the utilization of TED for collectively tuning MRs provides further energy benefits on top of the 20% reduction we obtained from the analysis in section 4.5.3. TED also allows for closer placement of MRs, which in turn helps reduce propagation delays. This reduction is also impacted by the faster inputs to DAC arrays enabled by local buffering and ping-pong buffers in the VDP units.

Lastly, in Figure 37 we present the average FPS/Watt comparison between the various accelerator platforms. Both the *ROBIN* variants perform well against the accelerator platforms to which they were compared against. *ROBIN-EO* outperforms all other platforms other than FBNA and FINN. This is owing to the extremely low power consumption reported by these BNN

accelerators. However, the *ROBIN* variants display superior FPS performance with respect to these electronic accelerators, as can be seen in Figure 38.

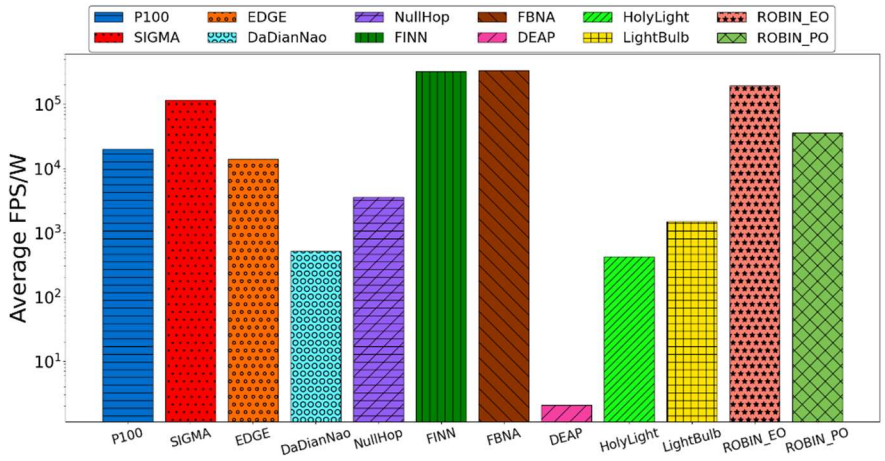


Figure 37. Average FPS/Watt among different accelerator platforms, visualized.

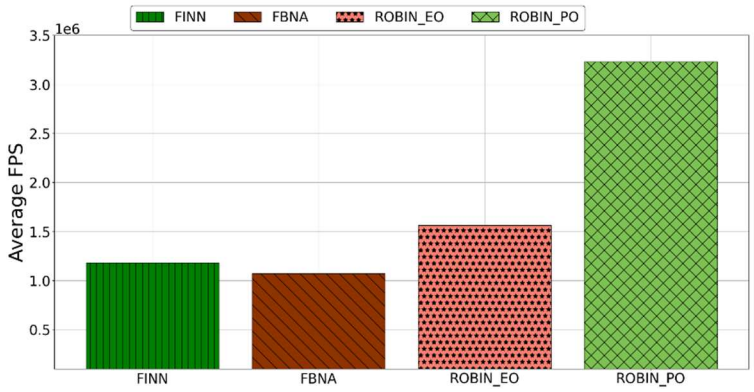


Figure 38. FPS comparison between the ROBIN variants and the electronic BNN accelerators.

Table 11 Inference time on ROBIN-PO and Intel i7 desktop for the four models considered in evaluations

Model no.	Parameters	Datasets	Inference time (for one image)	
			ROBIN-PO	i7-4790
1	60,642	Sign MNIST	0.0218 μs	0.16 ms
2	1,546,570	CIFAR10	0.28 μs	1.75 ms
3	13,570,186	STL10	2.3 μs	2.5 ms
4	552,362	SVHN	0.11 μs	1.25 ms

In summary, this chapter showcases the effectiveness of cross-layer design of BNN accelerators with the emerging silicon photonics technology for energy/area efficient implementations and for performance-oriented designs. Overall, we can see that our energy efficient design (*ROBIN-EO*) exhibits EPB values ~4× lower than electronic BNN accelerators and ~933× lower than the photonic BNN accelerator, while the performance-oriented design (*ROBIN-PO*) shows ~3x and ~25x better FPS than the electronic and photonic BNN accelerators respectively. With the growing maturity of silicon photonic device fabrication in CMOS-compatible processes, it is expected that the energy costs of device tuning, losses, and laser power overheads will go further down, making an even stronger case for considering optical-domain accelerators for deep learning inference.

4.5.5 COMPARISON TO CPU BASED INFERENCE

To highlight the advantage of dedicated inference acceleration, we have compared the performance of our *ROBIN* architecture against a standard desktop CPU performing inference on these models. The CPU we have considered is an Intel i7-4790, and we have used Tensorflow to

analyse the latency for inference. The CPU, i7-4790, is reported to have an average power consumption of approximately 103W. This power consumption is comparable to the ~90 W we report for the *ROBIN*-PO variant. The summary of observations for inference time are shown in Table 11. The *ROBIN* accelerator is observed to provide several orders of magnitude reduction in inference time for all the four models and datasets, compared to the Intel i7 system.

4.6 CONCLUSION

In this chapter we proposed *ROBIN,* an optical-domain BNN accelerator which utilizes device-level, circuit-level and architecture-level optimizations to save on energy and area while improving overall throughput. Through our optimization efforts, we identified two variants of *ROBIN*: *ROBIN-EO*, which is optimized for energy and area efficiency, and *ROBIN-PO*, which exhibits higher FPS performance, at the expense of greater power consumption. Our simulation analysis showed that *ROBIN* exhibits significantly better EPB performance than the various state-of-the-art optical neural network accelerators. Owing to significantly lower power consumption reported by the electronic BNN accelerators considered, *ROBIN* variants are not able to obtain better FPS/Watt than them, but upon closer examination both *ROBIN* variants can be seen to have better throughput than the electronic BNN accelerators. These results highlight the promise of our proposed *ROBIN* accelerator for accelerating BNN model execution for resource-constrained platforms.

The work described in this chapter is focused on BNN acceleration using photonic systems. In this work, we considered how photonic systems can be used to accelerate the partially binarized networks, with weights remaining binary, while activations being multi-bit parameters. To improve on this work, one may consider employing mixed quantization in the models considered,

where different layers have different levels of quantization for their activation parameters. This can enable better accuracy for the considered models. The photonic system- and device-level optimizations discussed in this chapter are not limited to BNN inference accelerators. These techniques may also be considered for other non-BNN accelerators for DNN/CNNs as well.

CHAPTER 5: NON-COHERENT PHOTONIC ACCELERATOR DESIGN FOR UNSTRUCTURED SPARSITY IN CONVOLUTION NEURAL NETWORKS

Over the past decade, convolutional neural networks (CNNs) have exhibited success in many application domains, such as image/video classification, object detection, and even sequence learning. As CNNs continue to be used for solving more complex problems, they have become increasingly compute and memory intensive. This is reflected in the increase in operations needed from ~4.5 million for LeNet-5 [120], proposed in 1998, to ~30 billion for VGG16 [158], proposed in 2014. To keep pace with the continuous increase in CNN resource requirements, several accelerator platforms have been proposed, including graphical processing units (GPUs) with tensor units, Google's tensor processing units (TPUs), and custom application-specific integrated circuits (ASICs). However, these platforms still have low performance and energy-efficiency for most CNN applications. Sparse neural networks (SpNNs) [159] enable a reduced number of neurons and synapses while maintaining the original model accuracy. Therefore, they represent a promising optimization to reduce the overall resource requirements for CNNs in resource-constrained environments.

Unfortunately, simply deploying a SpNN on an accelerator does not necessarily ensure model performance and energy-efficiency improvements. This is because the strategies for dense neural network acceleration, for which most accelerators today are optimized, are not be able to take advantage of the sparsity available in neural networks. Dense neural network accelerators orchestrate dataflow and operations for parameters that have been sparsified (i.e., zeroed out). By having to process sparse parameters, conventional accelerators incur high latency and energy consumption that should be avoided. Therefore, carefully devised strategies for taking advantage of sparsity and reducing the number of operations becomes essential.

There have been a few recent efforts to design accelerators that provide support for SpNNs [155], [160], [161]. But these electronic accelerators face fundamental limitations in the post-Moore era, where processing capabilities are no longer improving as they once did, and metallic wires create new dataflow bottlenecks [106]. Neural network accelerator architectures that leverage silicon photonics for computing and data transfer can enable low latency and energy-efficient computation solutions [43], [44], [111], [162], [163]. However, they are not impervious to the high latency and energy wastage problem when accelerating SpNNs.

In this chapter, we present a novel neural network accelerator designed with silicon photonics that is optimized for exploiting sparsity, to enable energy-efficient and low-latency SpNN acceleration. To the best of our knowledge, this chapter presents the first non-coherent photonic SpNN accelerator. Our novel contributions in this chapter include:

- The design of a novel photonic-domain SpNN hardware accelerator architecture that utilizes a modular, vector-granularity-aware structure to enable high throughput and energy-efficient execution across different CNN models;
- Sparsity-aware data compression and dataflow techniques for fully connected and convolution layers, which are tuned for the high throughput operation of our photonic accelerator;
- A comprehensive comparison with state-of-the-art sparse electronic and dense photonic CNN accelerator platforms, to demonstrate the potential of our accelerator platform.

5.1. RELATED WORK

To efficiently accelerate SpNNs, there is a need for specialized hardware architectures. In recent years, a few such architectures have been proposed by the electronic machine learning (ML) acceleration research community, e.g., [160], [161], [155]. The framework presented in [160] leveraged a custom instruction-set architecture (ISA) for SpNNs. Specialized buffer controller architectures were used, which involved indexing to keep track of sparse elements, thus preventing them from being fed to the processing elements. In [161], a software-hardware co-optimized reconfigurable sparse CNN accelerator design was proposed for FPGAs. The architecture exploited both inter- and intra-output feature map parallelism. Kernel merging along with structured sparsity were considered to further improve the overall efficiency. The work in [155] described an FPGA-based implementation of a sparse CNN accelerator. The accelerator made use of an output feature-map compression algorithm, which allowed the accelerator to operate directly on compressed data.

To obtain lower latency and better energy efficiency, there has been growing interest in using silicon photonics for ML acceleration [37], and many-core computing in general [32]. Silicon photonic neural network accelerators can be broadly classified into two types: *coherent* and *non-coherent*. Coherent architectures use a single wavelength to operate and imprint weight and activation parameters onto the electrical field amplitude of optical signals, e.g., [162]. In contrast, non-coherent architectures use multiple wavelengths, where each wavelength can be used to perform an individual neuron operation in parallel with other wavelengths, e.g., [43], [44], [111], [162], [163]. In these architectures, parameters are imprinted directly onto signal amplitude. The recent work in [162] was the first to consider sparsity in the design of coherent photonic neural network accelerators. Structured sparsity techniques were used along with fast Fourier transform

(FFT) based optical convolution, with the goal of reducing the area consumption of coherent architectures. The singular value decomposition (SVD)-based approach for phase matrix representation, which is crucial to reduce the overall area of the coherent architecture, also makes these architectures susceptible to accuracy loss, as the experiments in [162] showed. Due to phase encoding noise, phase error accumulation, and scalability limitations of coherent accelerators [164], there has been a growing interest in non-coherent photonic accelerators. Non-coherent dense neural network accelerators were proposed in [43] and [111], where the basic device for multiply and accumulate units relies on microring resonators [43] and microdisks [111]. The work in [43] also utilized cross-layer device-level and circuit-level optimizations to enable lower power consumption in the optical domain. The SONIC architecture proposed in this work represents the first non-coherent photonic SpNN accelerator, where multiple software optimizations for sparsity, clustering, and dataflow are integrated closely with the hardware architecture design for improved energy-efficiency and latency, without compromising on inference accuracy.

The rest of the chapter is organized as follows. Section 5.2 provides an overview of our proposed software and dataflow optimizations for convolution and fully connected layers in CNNs. Section 5.3 describes the hardware design of our non-coherent photonic accelerator that is tuned for these model optimizations. Section 5.4 presents the experiments conducted and results. Lastly, we draw conclusions in Section 5.5.

5.2. SOFTWARE AND DATAFLOW OPTIMIZATIONS

5.2.1. MODEL SPARSIFICATION

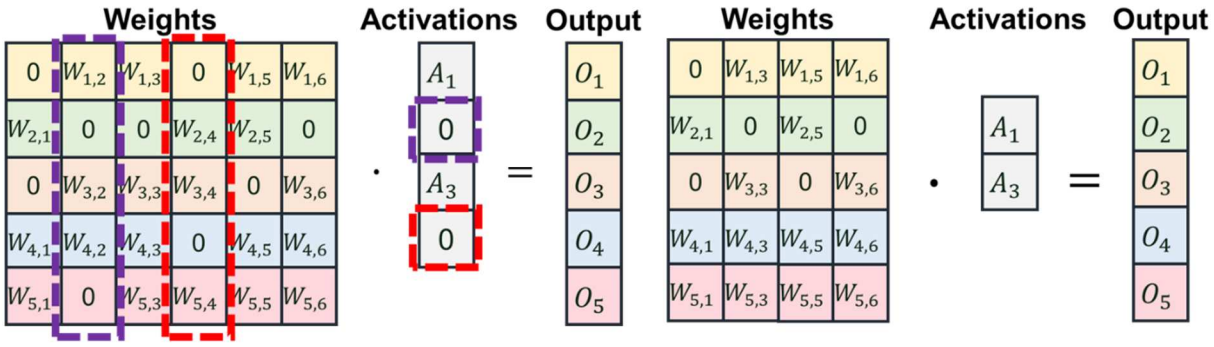
To generate SpNNs, we adapt a layer-wise, sparsity-aware training approach from [165]. We opt for layer-wise sparsity instead of sparsifying the entire model, to have more control over the

process, and to avoid overly sparsifying sensitive layers which notably contribute to the overall model accuracy. In our approach, for every layer selected to be sparsified, a binary mask variable is added, which is of the same size and shape as the layer's weight tensor. The algorithm also determines which of the weights participate in the forward execution of the graph. The weights in the chosen layer are then sorted by their absolute values and the smallest magnitude weights are masked to zero until the user-specified sparsity levels are reached. Also, note that we opt for sparsity-aware training instead of post-training sparsification, as the latter approach can indiscriminately remove neurons, thus adversely affecting the inference accuracy. We also utilize an L2 regularization term during training, to encourage smaller weight values and avoid overfitting, which further helps improve the overall accuracy post-deployment.

5.2.2. WEIGHT CLUSTERING

The electrical-optical interface in photonic accelerators, such as in [43], can be highly power consuming. This is because digital-to-analog converters (DACs), which have high power overheads, are used in these interfaces to tune the optical devices in the multiply-and-accumulate (MAC) units. Moreover, higher resolution (i.e., the number of bits used to represent each weight and activation parameter) requirements for a DAC translate into higher power and latency overhead in the DAC. Thus, to reduce the DAC overhead, we perform post-training quantization of the models, in the form of weight clustering. We opt for density-based centroid initialization of the weights, for the clustering operation, as described in [166]. For this clustering approach, a cumulative distribution function is built for the weights. The distribution is evenly divided into regions, based on the user specified number of clusters. The centroid weight values of the evenly distributed regions are then deduced, and these values are used to initialize clustering. This process effectively reduces the variations in weight values and confines the values to the centroids.

Therefore, if there are C centroids, and thus C clusters, the model will end up with C unique weights. This implies that the weights can be represented with a resolution of $\log_2 C$, thus reducing the required DAC resolution and enabling power and latency savings. Section 5.4.1 describes our weight clustering (and sparsification) explorations and parameters in more detail.



(a) (b)

Figure 39. FC layer operation, where the product of the weight matrix and activation vector is calculated. (a) Zero element identification in the activation vector and corresponding columns in weight matrix (marked in dotted outlines); (b) Compressed ma matrix and vector, but weight matrix still exhibits parameter sparsity.

5.2.3. DATAFLOW OPTIMIZATIONS

Beyond sparsification and weight clustering optimizations, we also perform enhancements to improve dataflow efficiency in our hardware platform. Fully Connected (FC) layers are computationally intensive layers in CNNs where all neurons in the layer are connected to all the other neurons in the following FC layer. The baseline FC-layer operation is a matrix-vector product, which generates the output vector to be passed on to the next FC layer, as represented in Figure 39(a). As the figure shows, there can be many parameters in the weight matrix and the activation vector which are zeroes. These zero parameters can be prevented from being passed on to the processing elements to reduce model latency and energy consumption.

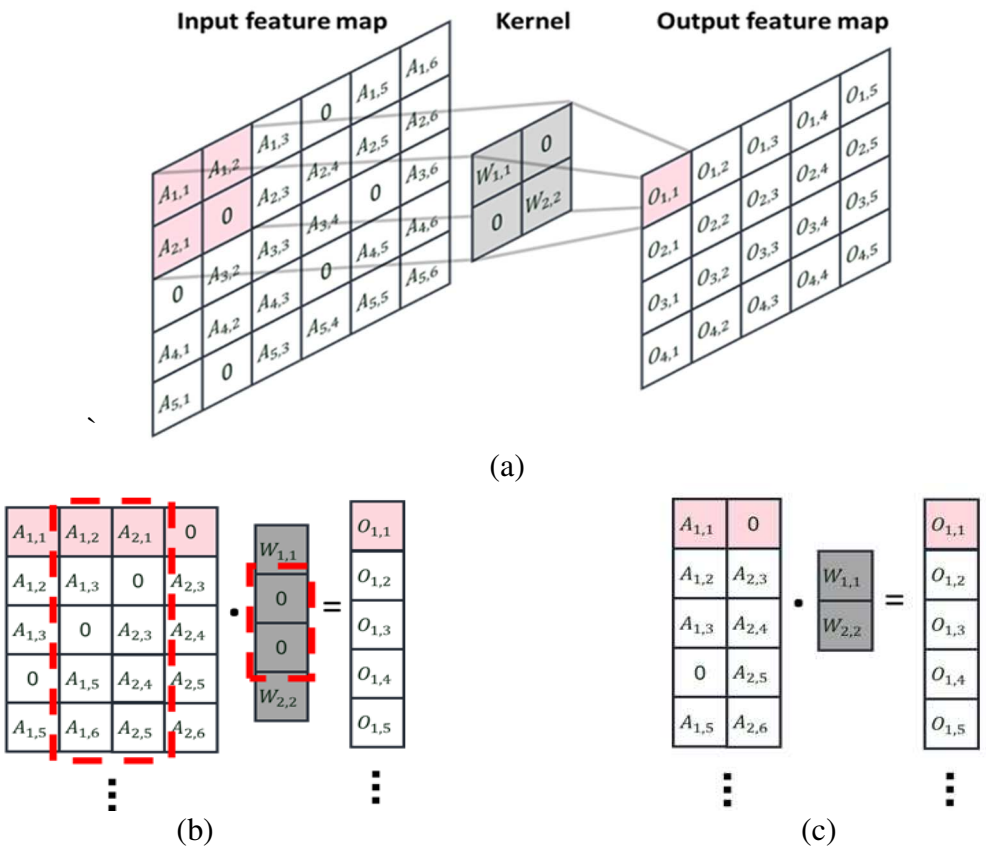


Figure 40. (a) Convolution operation between kernel (weight) matrix and input feature map (activations). A patch of the input feature map is convolved with the kernel matrix at a time to generate an output feature-map element (a patch and the corresponding output element are shown in red boxes). (b) Convolution operation unfurled into a vector-matrix-dot-product operation; avenues for compression are indicated by dotted-red outlines. (c) The result of the compression approach, with input feature map still exhibiting parameter sparsity.

To achieve this goal, a compression approach as depicted in Figures 39(a) and 39(b) is utilized. In this approach, we identify the zero parameters in the activation vector, and remove the corresponding columns in the weight matrix which will be operated upon by these parameters during the dot-product operation. This approach generates dense activation vectors, but the weight vectors can still be sparse, as depicted in the weight matrix in Figure 39(b). This process also does not impact the output vector calculation accuracy or output vector dimension.

For convolution (CONV) layers, the main difference with FC layers is the convolution operation performed in CONV layers. We unroll the CONV layer kernels and their associated patch of the input feature (IF) map matrix, to form vector-dot-product operations from the convolution operations (see Figure 40(a)). The compression approach for FC layers can be repeated for these unrolled matrix-vector multiplication operations (see Figure 40(b)). The compression approach in CONV layers helps generate dense kernel vectors to be passed to the vector-dot-product units (VDUs). Note that the IF vectors (activations) being passed for processing may still have sparsity present, as shown in Figure 40(c). The residual sparsity in the FC layer weight matrices and the CONV layer IF maps is handled at the vector-dot-product unit (VDU) level, as discussed in more detail in Section 5.3.2.

5.3. *SONIC* HARDWARE ACCELERATOR OVERVIEW

Figure 41 shows a high-level overview of the proposed non-coherent *SONIC* architecture for SpNN inference acceleration. *SONIC* comprises of an optical processing core, which uses vector-dot-product units (VDUs)—described in Section 5.3.2—to perform multiply and accumulate operations for FC and CONV layers in the photonic domain during inference. Several peripheral electronic modules are also integrated, to interface with the main memory, map the dense and sparse vectors to the photonic VDUs, and perform post-processing operations, such as applying non-linearities and accumulating partial sums generated by the photonic core. DAC arrays within VDUs convert buffered signals into analog tuning signals for MRs, and vertical-cavity surface-emitting lasers (VCSELs) are used to generate different wavelengths. Analog-to-digital converter (ADC) arrays are used to map the output analog signals generated by photonic summation to digital values that are sent back for post-processing and buffering. The devices, VDU, and architecture are discussed further in the following subsections.

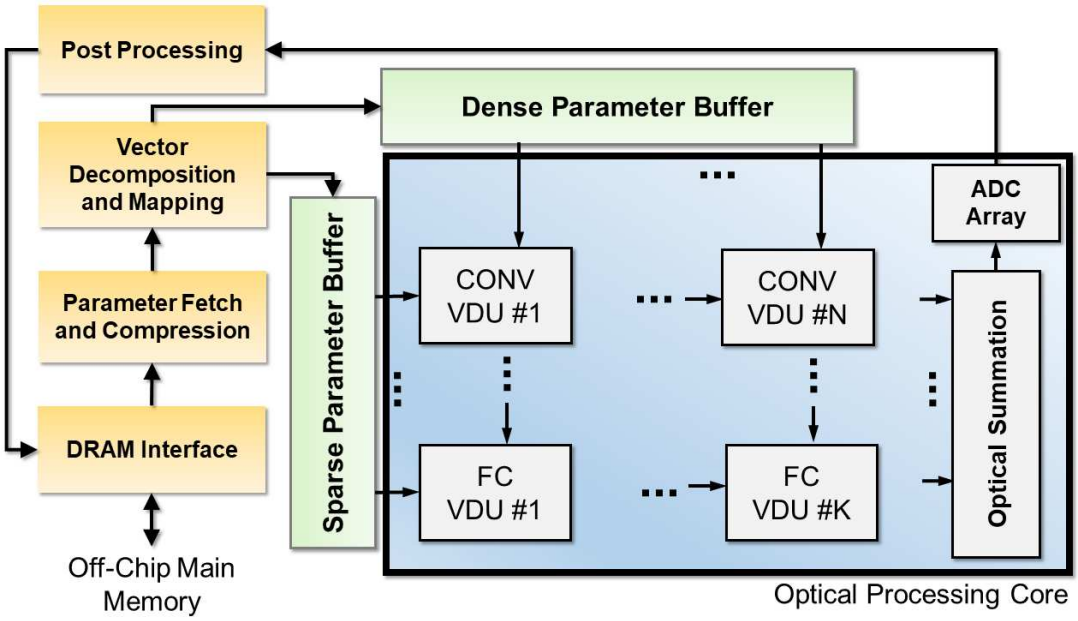


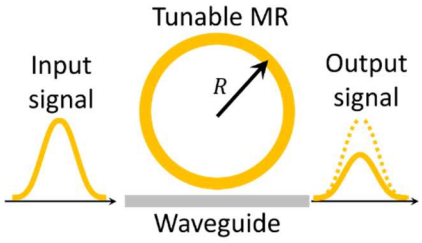
Figure 41. An overview of the *SONIC* architecture, with *N* CONV layer-specific VDUs and *K* FC layer-specific VDUs.

5.3.1. MICRORING RESONATORS (MRS) AND ROBUST TUNING

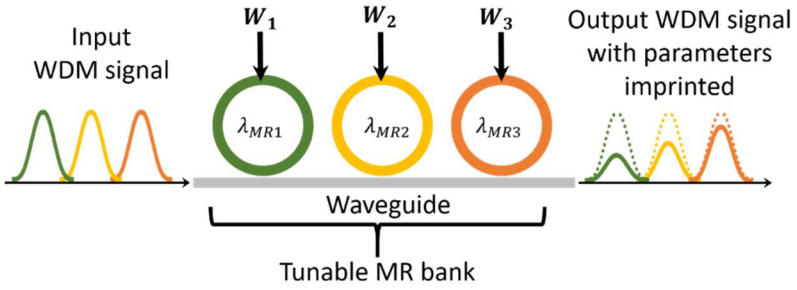
MRs are the primary devices used within our VDUs to implement matrix-vector multiplication operations. MRs are wavelength-selective silicon photonic devices, which are usually designed to be responsive to a specific 'resonant' wavelength (λ_{MR}). Such MRs are used to modulate and filter their resonant wavelengths in a carefully controlled manner, via a tuning circuit, to realize multiplications in the optical domain.

An MR tuning mechanism can be used to induce a resonance shift ($\Delta\lambda_{MR}$), and to change the output wavelength amplitude (Figure 42(a)) to realize a scalar multiplication operation. The tuning mechanism in MRs operates by heating (thermo-optic (TO) tuning [39]) or carrier injection (electro-optic (EO) tuning [40]), thereby inducing a change in the effective index (n_{eff}), which in turn impacts λ_{MR}. The induced $\Delta\lambda_{MR}$ increases the loss a wavelength experiences as it passes the MR, modifying the amplitude and imprinting the desired parameter (W_1–W_3 for MR_1–MR_3 in Fig

4(b)). To improve throughput, Wavelength Division Multiplexing (WDM) signals are used with a group of MRs (i.e., MR bank, Figure 42(b)), where each MR is sensitive to a specific λ_{MR}. A large passband in MRs can be achieved by cascading several of them, as in [146], which can be used to simultaneously tune multiple wavelengths.



(a)



(b)

Figure 42. (a) An all-pass microring resonator (MR) filter (R is the radius of the MR and determines the resonant wavelength). (b) An MR bank where multiple MR filters, each sensitive to a particular wavelength, are arranged to perform vector-matrix multiplication.

In *SONIC,* we make use of a TO+EO hybrid tuning circuit to induce $\Delta\lambda_{MR}$. Such a tuning approach has previously been proposed in [117] for silicon photonic devices with low insertion loss. This approach can be easily transferred to MR banks for hybrid tuning in our architecture. The hybrid tuning approach supports faster operation of MRs with fast EO tuning to induce small $\Delta\lambda_{MR}$ and using TO tuning for large $\Delta\lambda_{MR}$. To further reduce the power overhead of TO tuning in our hybrid approach, we adapt a method called thermal eigenmode decomposition (TED), which

was first proposed in [54]. Using TED, we can collectively tune the MR bank with lower power consumption.

5.3.2. VECTOR-DOT-PRODUCT UNIT (VDU) DESIGN

As we decompose the operations in FC and CONV layers to vector-dot-product operations, our processing units are effectively vector-dot-product units (VDUs). Figure 43 depicts the VDU design in *SONIC*. As Figures 39(b) and 40(c) showed, the granularity of the vectors involved in FC and CONV operations can be different. In real models, the CONV kernel sizes are relatively small when compared to FC layers. Also, in our dataflow for CONV layers, the dense vectors are generated by kernel matrices (weights), and for FC layers, the dense vectors are generated by activation vectors. However, for CONV layer dense vectors, we only need low resolution digital-to-analog converters (DACs), because of the clustering approach we utilize (see Section 5.2.2). Moreover, for FC layers, the sparse vectors may utilize the low-resolution DACs, due to the same reason. Therefore, considering these differences, we separate the VDU implementations for CONV and FC layers. However, both the VDU implementations follow the layout illustrated in Figure 43.

As shown in Figure 43, VDUs use separate local buffers to store the sparse and dense vector parameter values. The parameters are fed into DAC arrays for driving the optical devices (MRs or VCSELs). Each VDU has a local VCSEL array, which is driven using a DAC array. A DAC drives its corresponding VCSEL to generate optical signals with amplitude tuned to reflect its corresponding vector parameter.

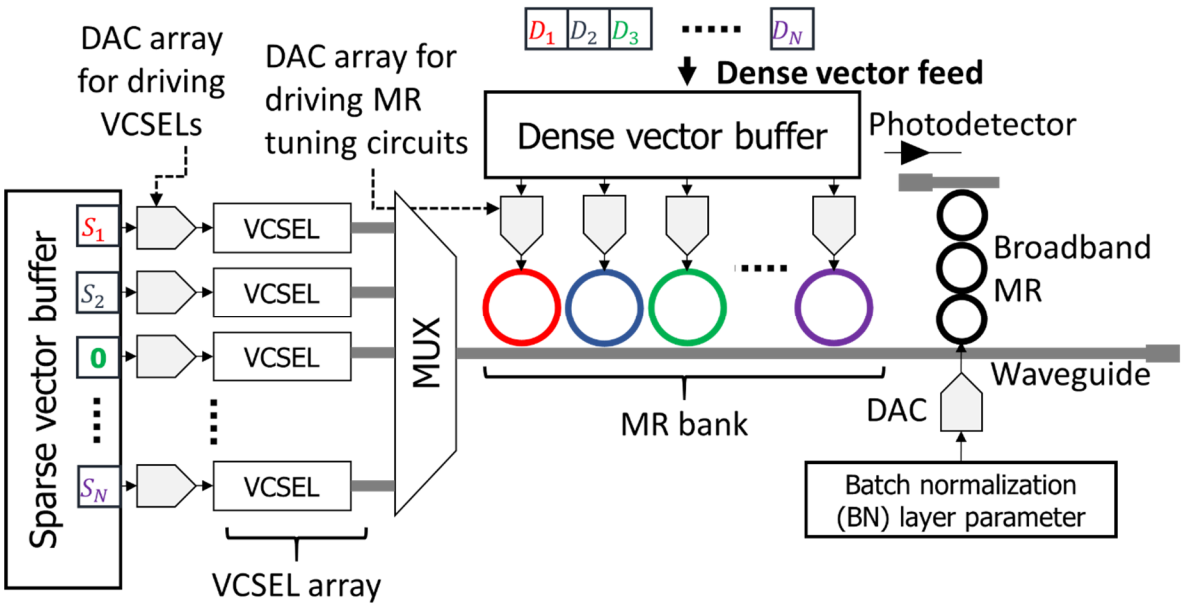


Figure 43. Vector-dot-product unit in the *SONIC* architecture.

We enhance the VDU design for sparsity by preventing a VCSEL from being driven if a zero element is encountered in the sparse vector (recall that after the compression approach described in Section 5.2.3, there may be residual sparsity in the IF map or weight matrix). This involves power gating the VCSEL, and hence subsequent operations for the dot product will not occur. The power gating thus helps avoid the wasteful operations with the zero parameters, which were not eliminated by our data compression approach (see Section 5.2), in the vectors fed to the VDU. The signals from the VCSEL array are fed into an optical multiplexer (MUX in Figure 43) to generate a WDM signal, which is transmitted to the MR bank via a waveguide. The MR bank is comprised of several tunable MRs, each of which can be tuned to alter the optical signal amplitude of a specific input wavelength, so that the intensity of the wavelength reflects a specific value, as discussed earlier.

We also make use of a broadband MR to tune all wavelengths simultaneously to reflect batch normalization (BN) parameters for a layer (Figure 43). Once the multiplication between parameters and BN parameters have been performed, a photodetector is used to convert the optical signal back to an electrical signal, to obtain a single, accumulated value from the VDU.

5.3.3. SONIC ARCHITECTURE

The VDU design discussed above is integrated in the *SONIC* architecture shown in Figure 41. As mentioned earlier, we separate the VDU designs for the FC and CONV layer operations. The separate VDU designs account for the vector granularity differences between FC and CONV layer operations, and the differences in DAC requirement for driving the VCSEL and MR arrays. The architecture relies on an electronic-control unit for interfacing with the main memory, retrieving the parameters, mapping the compressed parameters, and post-processing the partial sums generated by the VDUs. The optical processing core (see Figure 41) focuses on CONV, FC, and batch normalization acceleration during the inference phase. Other operations, such as activation and pooling are implemented electronically, as done in prior works on optical computation. The *SONIC* architecture design in Figure 41 arranges VDUs in an array. For CONV layers, we consider N VDU units, with each unit supporting an $n \times n$ dot product. For FC layer acceleration, we consider K VDU units, with each unit supporting a $m \times m$ dot product. Here, $m > n$ and $N > K$, as per the requirements of each of the distinct layers. In each VDP unit, the original vector dimensions are decomposed into n or m dimensional vectors. Here, n and m are dependent on the dense vector granularity we obtain through the compression approach for the CONV and FC layers (Section 5.2.3).

Table 12 CNN models considered for experiments.

Datasets	Conv layers	FC layers	No. of parameters	Baseline accuracy
MNIST	2	2	1,498,730	93.2%
CIFAR10	6	1	552,874	86.05%
STL10	6	1	77,787,738	74.6%
SVHN	4	3	552,362	94.6%

Table 13 Parameters considered for analysis of accelerators.

Devices	Latency	Power
EO Tuning [40]	20 ns	4 μW/ nm
TO Tuning [39]	4 μs	27.5 mW/FSR
VCSEL [167]	0.07 ns	1.3 mW
Photodetector [130]	5.8 ps	2.8 mW
DAC (16 bit) [149]	0.33 ns	40 mW
DAC (6 bit) [168]	0.25 ns	3 mW
ADC (16 bit) [148]	14 ns	62 mW

5.4. EXPERIMENTS AND RESULTS

For our experiments, we consider four custom CNN models with both CONV and FC layers, for the well-known CIFAR10, STL10, SVHN, and MNIST datasets. Details on the baseline models are shown in Table 12. For evaluating the performance of the *SONIC* architecture, we compare it against two state-of-the-art SpNN accelerators: *RSNN* [161] and *NullHop* [155], along with dense photonic accelerators *CrossLight* [43] and *HolyLight* [111], and a photonic binary neural network accelerator *LightBulb* [138]. Furthermore, we attempted to implement the coherent SpNN photonic accelerator from [162]; however, the work does not provide details or results for latency, power, and energy, which prevented us from comparing against it. We also show comparative results against the NVIDIA Tesla P100 GPU and Intel Xeon Platinum 9282 CPU. We compared all these

architectures in terms of throughput (i.e., frame per second (FPS)), energy per bit (EPB), and power consumption efficiency (FPS/W). We devised a custom Python simulator, integrated with Tensorflow v2.5, to evaluate *SONIC* and other accelerators. The parameters summarized in Table 13 were used to configure the accelerators to obtain performance and power/energy results.

Table 14 Summary of the sparsification and clustering results.

Datasets	Layers pruned	No. of weight clusters	No. of parameters	Final accuracy
MNIST	4	64	749,365	92.89%
CIFAR10	7	16	276,437	86.86%
STL10	5	64	46,672,643	75.2%
SVHN	5	64	331,417	95%

5.4.1. MODEL SPARSIFICATION AND CLUSTERING RESULTS

In the first experiment, we focus on software model optimization in *SONIC*. To obtain the best accuracy possible, we performed layer-wise sparsification in the models considered, as described in Section 5.2.1. We also use this experiment to partially explore the design space of *SONIC* hardware implementations. As depicted in Figure 43, we use DACs for driving MRs and VCSELs in our accelerator. To decide on the required DAC resolution (and corresponding power and latency costs), we perform post-training weight clustering, as described in Section 5.2.2. Our goal was to generate models with as much per-layer sparsity as possible, and minimal DAC resolution, while exhibiting comparable accuracy to the baseline model.

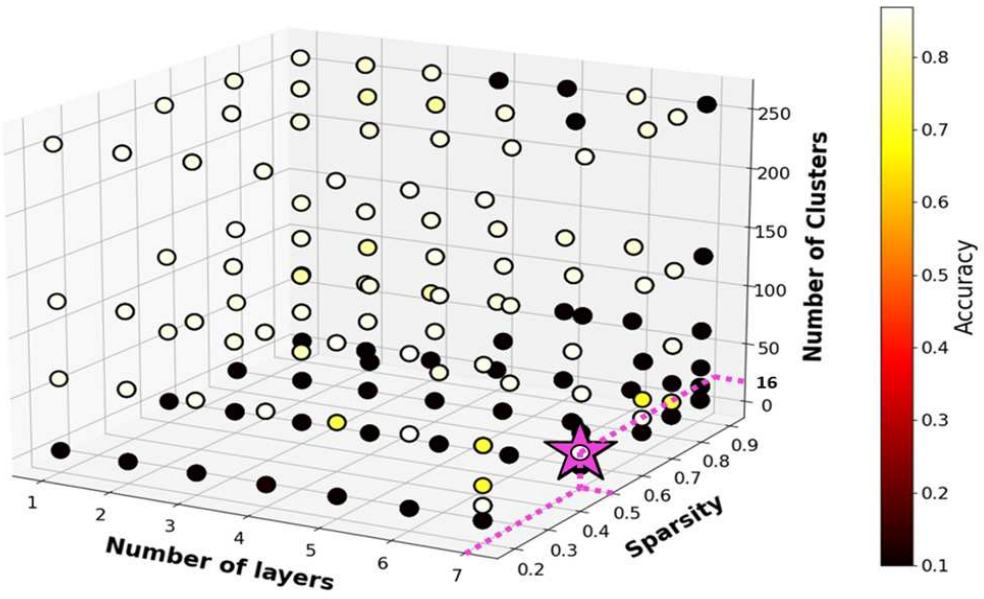


Figure 44. Visualization of sparsity and clustering exploration on the CIFAR10 model. Number of layers is the total layers sparsified, sparsity is the average pruning aggressiveness, and number of clusters refers to the total weights clusters. The best (highest accuracy) configuration is indicated by the star.

A summary of the optimized models and the final accuracy achieved after sparsification and weight clustering is shown in Table 14. Note that the final accuracy of the optimized models in Table 14 is comparable or slightly better than the baseline accuracy shown in Table 12, which is consistent with the trend in prior works. To arrive at these numbers for each model, we performed a detailed exploration. Figure 44 shows the design space considered during sparsity and clustering exploration for the CIFAR10 model (figures for the other three models are omitted for brevity).

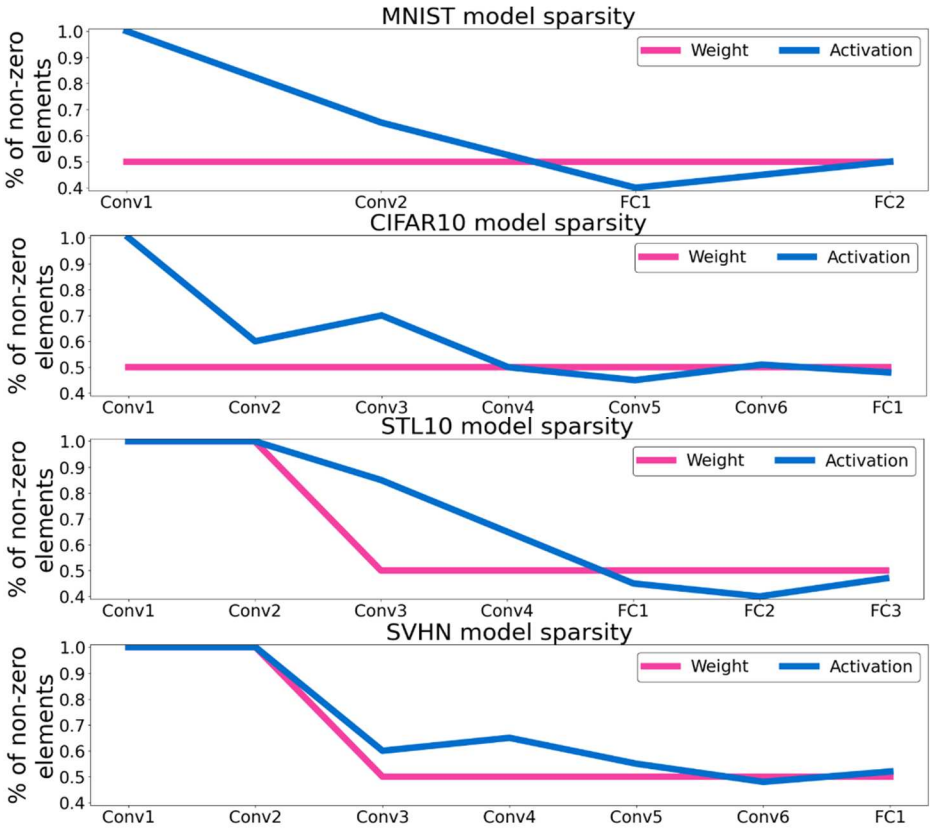


Figure 45. Sparsity across various layers in the four models considered.

Figure 45 further shows the layer-wise breakdown of sparsification for all four models, where the plots show the layer-specific sparsity level for weight parameters (in the best solution for each model from Table 14) and the resulting sparsity in activations as they traverse the sparse layers. Our exploration was able to identify the need for a maximum of 16 clusters for the best CIFAR10 solution and a maximum of 64 clusters across the four models (Table 14). Based on these results, we consider 6-bit DACs (to support up to 64 levels) for weight parameters. We kept activation granularity at 16-bits, which provided us with sufficient accuracy (Table 14) and thus used 16-bit DACs for activations, in the *SONIC* accelerator.

5.4.2. COMPARISON WITH STATE-OF-THE-ART ACCELERATORS

We explored various *(n, m, N, K)* configurations for the *SONIC* architecture (see Section 5.3.3) and found the best configuration in terms of FPS/W, EPB, and power consumption to be (5, 50, 50, 10). We found that the value of *n* is heavily dependent on CONV layer kernel values, which was fixed after our model sparsification experiments. Increasing *n* beyond five did not provide any benefits, as the dense kernel vectors do not exceed five-parameter granularity for the considered models.

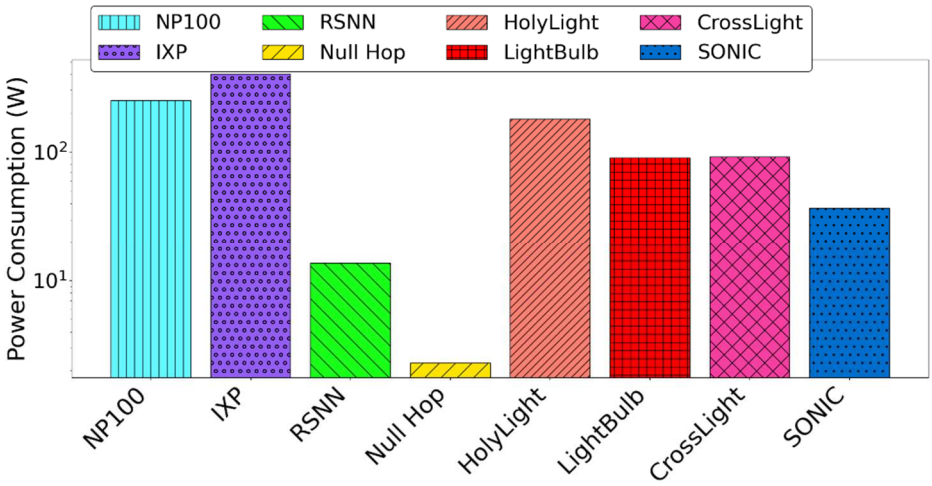


Figure 46. Power comparison across the accelerator platforms.

Figure 46 shows power consumption and Figure 47 shows the power efficiency (in terms of frames-per-second/watt or FPS/W) across the accelerators considered. In these figures, NP100 is the GPU and IXP is the CPU. We can observe that due to its sparsity-aware, clustering-aware, and dataflow-optimized hardware architecture design, *SONIC* exhibits substantially higher power efficiency, even though it has higher power consumption than the electronic SpNN accelerators. *SONIC*, on average, exhibits 5.81× and 4.02× better FPS/W than the *NullHop* and *RSNN* electronic SpNN accelerators. *SONIC* also exhibits 3.08×, 2.94×, and 13.8× better power efficiency on average than the *LightBulb*, *CrossLight*, and *HolyLight* photonic accelerators, respectively. This is

because none of these photonic accelerators are optimized to take advantage of sparsity and clustering.

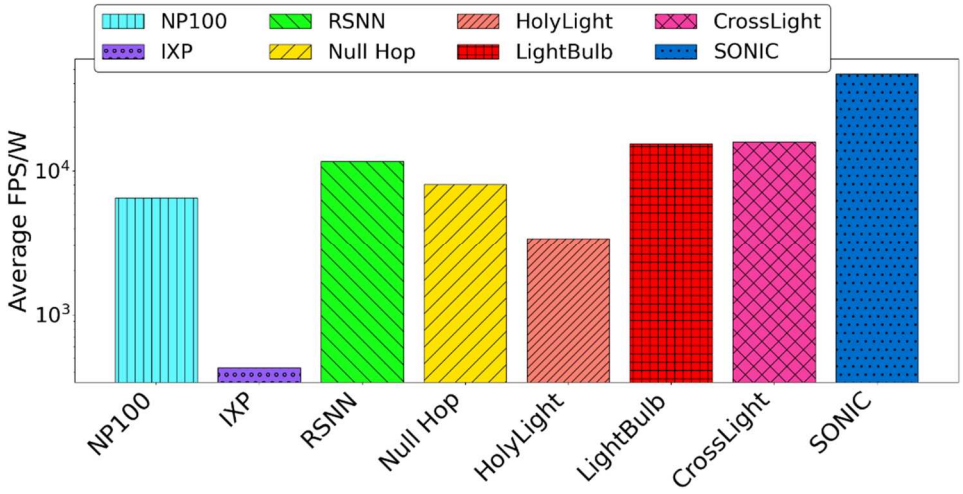


Figure 47. FPS/W comparison across the accelerator platforms.

When comparing the energy-per-bit (EPB) across the accelerators, as shown in Figure 48, we can again observe that the co-design of the software and dataflow optimizations along with the hardware architecture in *SONIC* allow it to outperform the photonic and electronic SpNN accelerators. *SONIC* exhibits, on average, 19.4×, 18.4×, and 27.6× lower EPB than *LightBulb*, *CrossLight*, and *HolyLight*, respectively. *SONIC* also exhibits 8.4× and 5.78× lower EPB than *NullHop* and *RSNN*. These results highlight the promise of *SONIC* for optimized SpNN implementations on resource-constrained platforms.

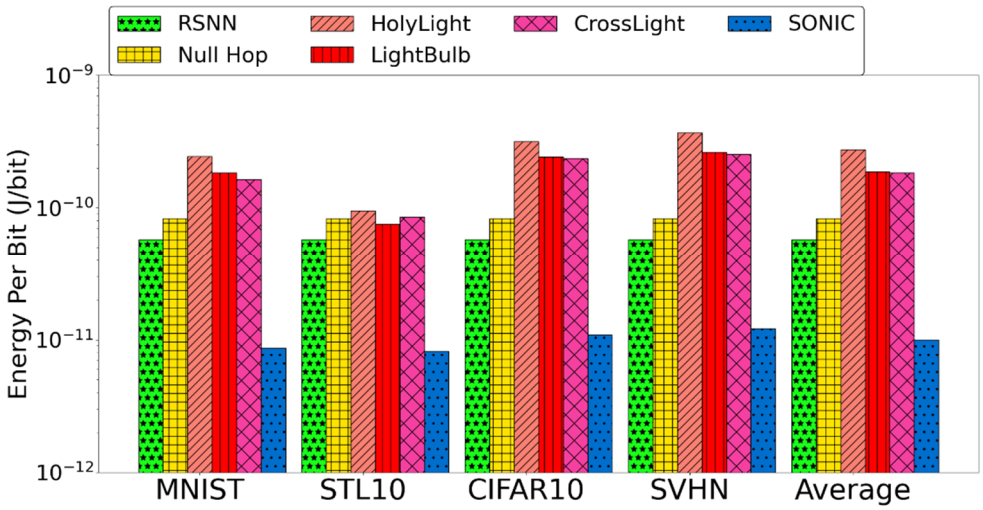


Figure 48. EPB comparison across the accelerator platforms.

5.5. CONCLUSIONS

In this chapter, we presented a novel non-coherent photonic sparse neural network accelerator, called *SONIC*, that integrates several hardware and software optimizations. *SONIC* exhibits up to 5.8× better power efficiency, and 8.4× lower EPB than state-of-the-art sparse electronic neural network accelerators; and up to 13.8× better power efficiency and 27.6× lower EPB than state-of-the-art dense photonic neural network accelerators. These results demonstrate the promising low-energy and low-latency inference acceleration capabilities of our *SONIC* architecture.

CHAPTER 6: EXPLORING NON-COHERENT PHOTONICS FOR ENERGY EFFICIENT, HIGH THROUGHPUT ACCELERATOR FOR HETEROGENEOUSLY QUANTIZED CONVOLUTION NEURAL NETWORKS

In the past decade, artificial neural networks (ANNs) have gained popularity for their ability to serve as a universal approximator for any computational function. ANNs have been applied to many complex problems and have replaced older machine learning (ML) algorithms in many domains. Convolutional neural networks (CNNs) are one of the most well-known ANNs that have exhibited success in many application domains such as image/video classification, object detection, and even sequence learning. As CNNs continue to be used for solving increasingly complex problems, they have in turn become more and more compute and memory intensive. Research into exploring how to reduce the memory footprint of the CNN model while retaining inference accuracy has been an active area of research in recent years. Some examples of such research areas include exploiting sparsity in CNNs [165], [17], where unnecessary model (weight and activation) parameters are pruned, and inducing quantization in CNN models [169, 169], [170], [171], where the parameter sizes (bitwidths) are reduced. By reducing parameter bitwidths with quantization, both memory usage and computational latency (and energy) can be reduced. Conventional quantization approaches quantize CNN models so that all layers in the model have the same bit length for their parameters. However, the varying functional redundancies and diverse behavior across model layers necessitates the use of mixed precision (i.e., heterogeneous) quantization, for better accuracy.

The increasing CNN model complexity in emerging applications (e.g., autonomous drones, self-driving vehicles) also necessitates that the underlying hardware platforms consistently deliver better performance while satisfying strict energy requirements. Emerging accelerator platforms are

therefore being designed while considering CNN model optimizations, such as sparsity and quantization. For example, newer Nvidia GPUs can take advantage of sparsity in CNN models to increase throughput and energy efficiency [172].

Even with various optimizations at the hardware and software level, electronic CNN accelerators are still prone to diminishing energy and throughput efficiencies, due to the slow-down of Dennard scaling. A potential solution to obtain better energy and throughput efficiency for CNN applications is to consider more efficient hardware technologies, such as silicon photonics, during the design of CNN accelerators. Silicon photonics not only enables low-latency and high bandwidth communication but can also be used for low-latency, and energy-efficient computations, e.g., matrix-vector multiplication in the photonic domain [34], [36]. However, there are various challenges associated with designing a power and energy-efficient silicon photonic CNN accelerator, including high laser power consumption; high power utilization at electrical-photonic and photonic-electrical interfaces; and high latencies associated with photonic device tuning mechanisms. Moreover, none of the photonic CNN accelerators proposed to date support the execution of heterogeneously quantized CNN models.

In this work, we propose, *HQNNA*, a silicon photonic CNN accelerator designed for optimizing both homogeneous and heterogeneous quantization in CNN models for energy- and throughput-efficient high accuracy inference acceleration. Our novel contributions in this work include:

- The design of a novel non-coherent photonic accelerator which utilizes wavelength division multiplexing (WDM) along with time-division multiplexing (TDM) for bit-slicing based operation for heterogeneously quantized CNN acceleration;
- The design of modular vector granularity aware matrix-vector multiplication units for energy- and throughput- efficient accelerator operation;

- A comprehensive comparison with state-of-the-art photonic CNN accelerators to demonstrate the potential of our proposed CNN accelerator.

The rest of this chapter is organized as follows: Section 6.1 goes over related works and our motivation for this work. Section 6.2 discusses our architecture in detail. Section 6.3 presents details on experiments conducted and the results observed. Finally, Section 6.4 gives concluding remarks and directions for future work.

6.1. BACKGROUND AND MOTIVATION

6.1.1. PHOTONIC CNN ACCELERATION

Silicon photonics based ML accelerator architectures represent an emerging paradigm. These accelerators can be broadly divided into two major categories: coherent and non-coherent photonic accelerators. Non-coherent architectures use multiple wavelengths, where each wavelength can be used to perform an individual neuron operation. In these architectures, parameters are imprinted onto the optical signal amplitude directly, and to manipulate individual wavelengths, wavelength-selective devices such as microring resonators (MRs) or microdisks are used. The architecture we propose is a non-coherent photonic accelerator architecture.

MRs are designed to be wavelength-selective and respond to a specific 'resonant' wavelength (λ_{MR}). An MR can modulate and filter its λ_{MR} in a carefully controlled manner, via a tuning circuit (Figure 49(a)). This mechanism can be used to realize multiplications in the analog optical domain. An MR tuning mechanism can be used to induce a resonant shift ($\Delta\lambda_{MR}$), and to change the output wavelength amplitude to realize a scalar multiplication. Such a tuning mechanism can operate by carrier injection (electro-optic (EO) tuning [40]) or heating (thermo-optic (TO) tuning [39]), thereby inducing a change in the effective refractive index (n_{eff}) of the device, which introduces a

$\Delta\lambda_{MR}$. Such tuning can be considered for imprinting the desired weight or activation parameter, by varying the loss a wavelength experiences as it passes the MR. The interaction of the tuned wavelength with another MR with the same λ_{MR}, tuned to represent another CNN parameter, results in multiplication between the parameters. To implement a dot product operation, multiple MRs, each representing a different CNN parameter can be arranged in an MR bank (Figure 49(b)) and can be fed a WDM input. These products can be then summed using a photodetector (PD) which converts wavelength intensity to a representative electric current.

Several previous works have discussed CNN acceleration using non-coherent photonics principles. In [43], an MR based CNN accelerator architecture, which utilized modular vector dot product units, with separate accelerator units for convolutional and fully connected layers, was proposed. The work also used optimized MR designs and tuning circuit optimizations, for energy and throughput efficiency. The work in [111] utilized microdisks instead of MRs due to the lower area and power consumption they offer. Another microdisk based photonic accelerator, for fully binarized CNNs, was proposed in [138]. Fully binarized CNNs use single-bit weight and activation parameters. Because of this simplification, [138] was able to utilize energy-efficient photonic XOR and population count operations instead of multiply and accumulate operations. The work in [44] on the other hand, proposed an MR based partially binarized CNN accelerator. The partially binarized CNNs considered in the work used single-bit weight parameters and 4-bit activation parameters, which allowed for increased inference accuracy over fully binarized CNNs. To realize batch normalization operations, which are utilized in many binarized CNNs to boost performance, in the photonic domain, [44] used broadband MRs as well.

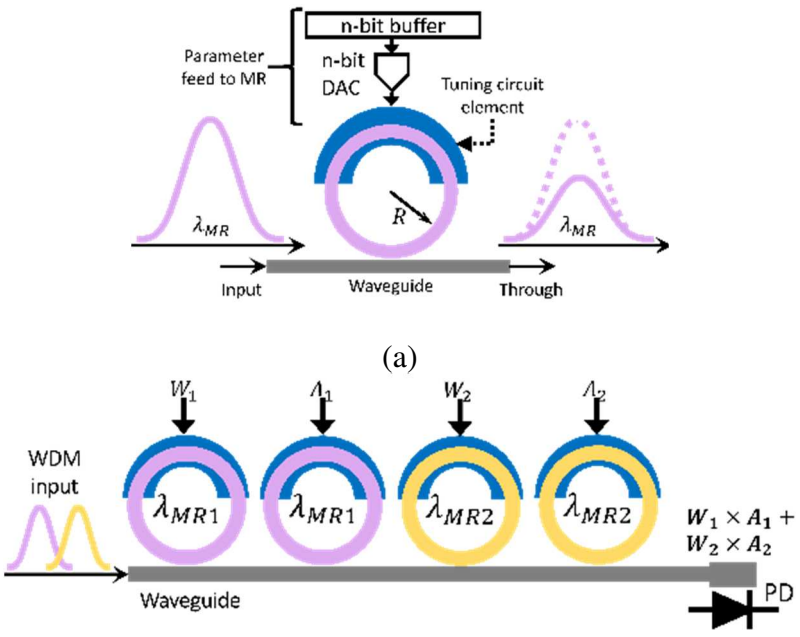


Figure 49. (a) Non-coherent parameter imprinting via MR tuning; (b) MR banks operating on a WDM signal to implement vector dot product.

6.1.2. HETEROGENEOUS QUANTIZATION IN CNNS

Quantizing weight and activation parameters in CNNs is an effective way to design efficient models, which in principle can help achieve better hardware efficiency. Conventional quantization approaches use the same precision for all the weight and activation parameters across layers (e.g., a bitwidth of 16 for parameters in all layers). This is referred to as homogeneous quantization. However, this approach leads to sub-optimal model accuracies. Heterogeneous or mixed precision quantization addresses this issue by allowing different layers to have different levels of quantization. But finding the optimum values for parameter quantization, per layer of a network, can be challenging. Utilizing exhaustive search is a naïve approach to obtain these quantization levels which is impractical for even very small model sizes. Another option is to utilize quantization exploration tools such as AutoQKeras. However, the lengthy search process in these tools quickly becomes taxing in terms of processing requirements as the number of layers grow.

Several efforts propose intelligent neural network architecture search strategies for optimizing the quantization levels across layers in a CNN. A differentiable neural architecture search (DNAS) framework was proposed in [169] to explore the search space with gradient-based optimization. The framework creates a stochastic supernet with weights and a trainable architecture parameter. The supernet is constructed with the same structure as the target CNN that needs to be heterogeneously quantized. The architecture parameter is used to identify the edges of the supernet that possess better accuracy. The supernet is trained with a custom loss function, with the Gumbel Softmax trick used to stabilize the training process. The technique presented in [170] is similar to the one in [169], with the key difference being the loss function, which penalizes a higher weighted average of the bitwidths of the weights across layers. The training method produces non-integer bitwidths, which are meaningless for practical hardware. Hence, the learned non-integer bitwidths are adjusted to the nearest greater integer. A multi-layer perceptron (MLP) based framework for generating the bitwidths of the parameters across layers of a CNN, for a given KL-divergence (Ω) from the baseline network's softmax output was proposed in [171]. Ω is calculated as the average of N images from the training set. The network to be heterogeneously quantized is randomly quantized initially and Ω is obtained, for S times. The bitwidths across the layers and corresponding Ω becomes the dataset to train the MLP. Once trained, the MLP is tasked with generating an output vector representing the optimum bitwidths across layers of the CNN for a given Ω value.

6.1.3. MOTIVATION

Given the prominence of quantized models to facilitate efficient CNN deployment on resource-limited embedded and IoT platforms, the ability to accelerate heterogeneously quantized models is essential for modern CNN accelerator architectures. The photonic architectures discussed in Subsection 6.1.1, have fixed parameter resolution and hence are not able to accelerate CNN models

with different quantization levels. The parameter resolution achievable in non-coherent photonic accelerators is limited by the noise-induced due to fabrication process variation (FPV), and crosstalk among MRs [37]. The maximum resolution of parameters is also dependent on the electrical-optical interface i.e. the digital to analog converters (DACs) used to pass the parameter values to the MR tuning circuitry. The architectures discussed in prior work are either designed for a particular resolution (e.g., [43] is designed for 16-bit resolution and uses 16-bit DACs) or target a very specific type of CNN model only (e.g., the XOR and pop-count mechanism of [138] limits it to only fully binarized CNNs). This prevents them from accelerating heterogeneously quantized models altogether or from effectively accelerating heterogeneously quantized models. An architecture such as the one presented in [43] may be able to represent any quantized model with parameter bitwidths up to 16-bits, but will not benefit in energy or latency from the lower bitwidths. To fully exploit quantization for latency and energy benefits, we propose the *HQNNA* accelerator in this work, which utilizes WDM and TDM, along with bit-slicing of parameters to achieve efficient inference performance. To the best of our knowledge, this is the first work that proposes a silicon photonic CNN accelerator optimized to efficiently execute CNN models with heterogeneous quantization across layers.

6.2. HQNNA HARDWARE ACCELERATOR

As mentioned earlier, *HQNNA* is a non-coherent photonic architecture that aims to address the challenge of accelerating heterogeneously quantized CNNs, while obtaining energy benefits from the quantization. The proposed architecture has a photonic accelerator substrate, which is comprised of an array of photonic matrix-vector multiply units (MVUs). An electronic control unit is used to interface with memory and to map CNN model parameters to MVUs. The outputs from the MVUs are processed locally before being passed on for non-linearity operations and then

storage back to memory. The following subsections go into the details of the architecture operation and optimizations.

6.2.1. TDM-BASED OPERATION AND ENERGY BENEFITS

In non-coherent photonic accelerator architectures, the parameters are imprinted on the wavelength amplitude. The parameters being imprinted are restricted in their resolution by two main factors: the DAC resolution and the presence of thermal or heterodyne crosstalk noise. Thermal crosstalk arises when the operation of adjacent TO tuning mechanisms perturb each other, causing tuning errors which in turn cause errors in parameter values. Heterodyne crosstalk is leakage of power from spectrally close wavelengths into each other, which can also cause errors.

DAC resolution is a limiting factor as, DACs have substantially higher power and latency requirements, as the DAC resolution increases. Due to these reasons, most photonic architectures opt to support low-resolution parameter resolution in CNNs. For example, the photonic accelerator discussed in [111] is designed for a 4-bit resolution, while those in [138] and [44] target binarized neural networks (1-bit weights). However, without sufficient optimizations of the CNN model, such as those discussed in Subsection 6.1.2, the quantized CNN may exhibit poor inference accuracy at low resolutions, as observed in results from the low-resolution models in [111], [138], [44]. The photonic accelerator in [43] proposed various optimizations in terms of tuning and WDM management to achieve a high resolution of 16-bits, which ensures better inference accuracy than [111], [138], [44]. However, such an architecture is at a disadvantage in terms of energy efficiency when accelerating a quantized model, where quantization levels vary across model layers.

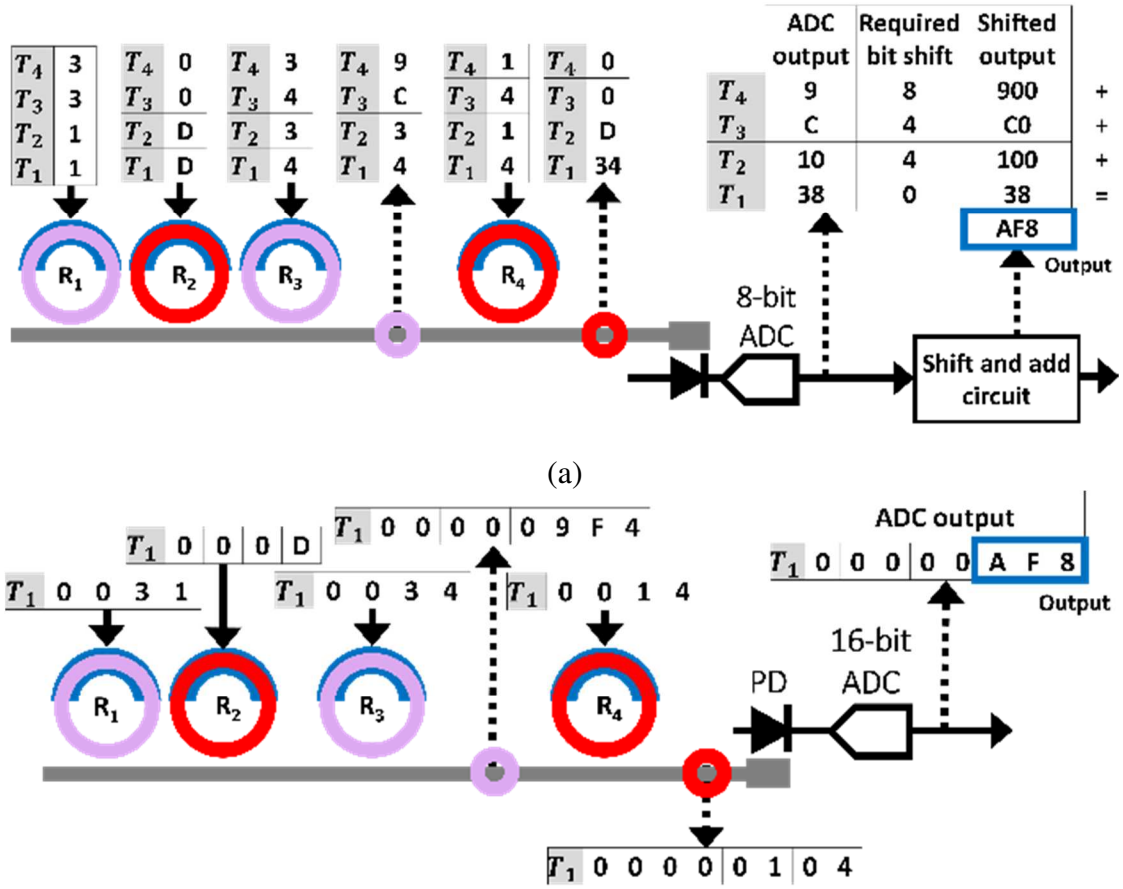


Figure 50. (a) TDM based operation for a vector dot product operation between two 2-element vectors in our proposed HQNNA architecture; (b) the same dot product operation performed with accelerator in [44].

To support heterogeneous quantization, and obtain the energy and power benefits it offers, we propose a novel bit-slicing and TDM based approach in our non-coherent architecture. This architecture, also makes use of WDM-based operations, but utilizes TDM and bit-slicing with it to more aggressively reduce power and energy consumption. Our approach distributes bit-slices across time steps onto the MVU to perform the multiplication and accumulation operation photonically, and then makes use of digital shift and adder circuits to obtain the correct output from the MVU operation. The number of time slices required to complete an operation depends on the bit-slice size (b) and the parameter size (p).

An overview of our operation, making use of a simple example is shown in Figure 50 involving multiplication of two 2-element vectors: A = [0×31, 0×0D] and B = [0×34, 0×14]. We assume that these two vectors represent the input activations and weights of a CNN layer, respectively, both of which are quantized to 8-bits (p). For *HQNNA* the four time steps may be explained as follows: At *T1* the least significant nibbles (4-bits) of elements of A and B are introduced into the multiplication unit. Elements, which have to interact with each other during the dot product operation, are assigned the same λ. So A_1 and B_1 are represented using λ_1 (purple in Figure 50(a)), and A_2 and B_2 are represented using λ_2 (red in Figure 50(a)). The interaction between the nibbles generates intermediate products at each time step, indicated by the purple and red circles and corresponding callout tables. The intermediate sums (generated using photodetectors) are converted to digital signals using an analog to digital converter (ADC), shifted appropriately, and are stored in a local buffer. The ADC data from each time step has a fixed shift value associated with it. Now, to generate the dot product the nibbles of A needs to interact with, i.e., be multiplied by, both nibbles of the corresponding elements in B. So, in *T2*, the second nibbles from the B-elements are imprinted, while A data does not change. After *T2*, all the data from B has been introduced to the least significant nibbles of A and corresponding sums are obtained. Thus on *T3*, the second nibble of A-elements can be introduced, and B needs to be fed again as in *T1* and *T2*. Thus, the proposed architecture, utilizing b of 4-bit will take up to four time steps to complete the operation (Figure 50(a)), while [44] will accomplish the same operation in a single time step (Figure 50(b)).

However, the energy consumption difference is staggering as our proposed architecture (Figure 50(a)), in this example, will consume approximately 6 *mJ* of energy while the architecture in [44] (Figure 50(b)) will consume up to 240 *mJ* of energy for this operation. Even for p of 16-bit (not shown), the proposed architecture, at b of 4-bit, will only have an approximate energy

consumption of 24 *mJ*, over 16 time steps while the [44] analog will still consume 240 *mJ* of energy. One limitation of our architecture is lower throughput than [44] (see results in Section 6.3); however our primary optimization goal is improved energy-efficiency. Moreover, the lowered throughput in our architecture is somewhat mitigated by the fact that lower resolution DACs used in our architecture have lower latencies associated with them. This is significant as one of the primary sources of latency other than TO tuning in a non-coherent architecture is DAC latency.

6.2.2. TUNING CIRCUITS

The thermo-optic (TO) tuning approach is widely used for FPV correction in MR-based systems and non-coherent architectures use them for imprinting CNN parameters. The main advantage of using TO tuning is its large tuning range (4-5 *nm*), but this comes with a higher latency (~μs range) and higher power overheads (~27 *mW/FSR*) [39] (FSR = free spectral range). Furthermore, the operation of TO tuning circuits can affect the fidelity of operation of neighboring MRs in the form of thermal noise [54]. Therefore, solely relying on microheater-based TO tuning can impair the operation of the non-coherent CNN accelerator. As an alternative, the electro-optic (EO) tuning mechanism operates through carrier injection into the MR body with a PN-junction across the MR. This in turn changes the effective refractive index (n_{eff}) of the device and hence the resonant wavelength (λ_{MR}), thereby tuning it. EO tuning is faster (~*ns* range) and consumes lower power (~4 $\mu W/FSR$), but it has a significantly lower tuning range than TO tuning (~1 *nm*) [40]. The lower tuning range means EO tuning alone is inadequate to address the large $\Delta\lambda_{MR}$s induced by FPV in MRs, but is sufficient for CNN parameter imprinting onto the resonant wavelength.

To overcome FPVs and for accurate parameter imprinting (required for photonic multiplication), in our architecture, we make use of a hybrid tuning circuit, which combines EO

and TO tuning. The hybrid tuning approach considers the advantages each tuning mechanism offers while covering for their disadvantages. The approach supports the efficient operation of MRs with fast EO tuning to quickly induce small $\Delta\lambda_{MR}$, such as in the case of parameter imprinting, and using the slower TO tuning infrequently to induce large $\Delta\lambda_{MR}$, for FPV correction in our architecture. To address the thermal noise generation from TO tuning, we adapt a method called thermal Eigenmode decomposition (TED), which was first proposed in [54]. TED also comes with the added advantage of significantly reducing TO power consumption and frequency of TO operation.

6.2.3. MVU DESIGN

To accelerate ANNs in general, the most time-consuming operations, matrix-vector multiplications, have to be accelerated. Inference acceleration in particular deals with fixed weight matrices and input-dependent activations. For CNNs, two main types of layers have to be considered: convolution (CONV) layers and fully connected (FC) layers. CONV layers perform convolution operations between smaller weight matrices or kernels and input feature maps (activations), to generate output feature maps for the next layer. The convolution operation can be decomposed into vector dot product operations as discussed in [44]. FC layers on the other hand perform matrix-vector multiplication operations between significantly larger weight matrices and activation vectors.

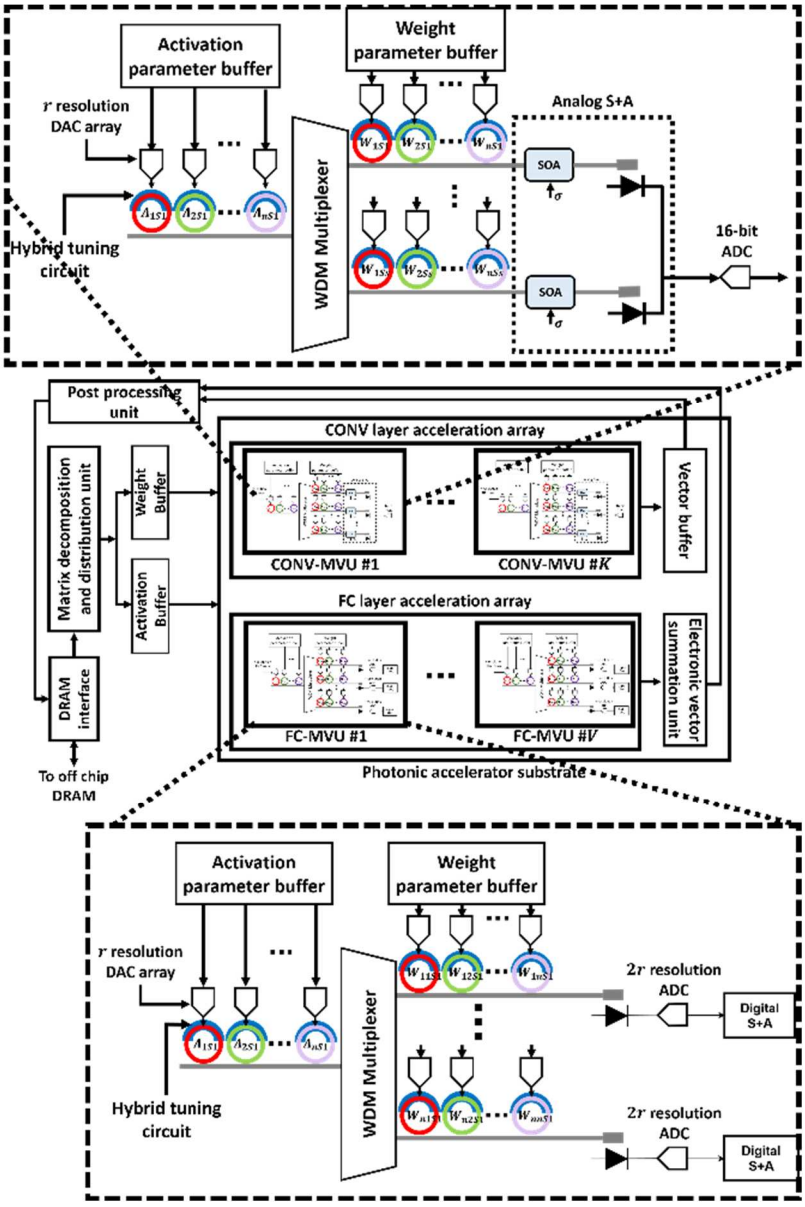


Figure 51. Architectural overview of HQNNA with the internal architecture of CONV-MVU and FC-MVU highlighted.

The basic compute unit in our architecture, to support both CONV and FC layer operations, is a matrix-vector multiplication unit (MVU). The MVU accepts a WDM signal through an input waveguide, which is imprinted with the vector parameters using an MR bank. For imprinting the parameters, we make use of DAC-based EO tuning in the hybrid tuning circuit (Figure 51). The

tuned signal from the MR bank is distributed across the matrix rows, again distributed across waveguides, using a splitter-based photonic multiplexer. The contents of the matrix depend on the layer being accelerated.

Table 15 The best models found through heterogeneous quantization techniques considered, compared to the quantized versions from other photonic accelerator works, in terms of inference accuracy and memory footprint

Model	No. of layers	Parameter count	Quantization type	Weight size across layers	Activation size across layers	Inference accuracy	Memory footprint
AlexNet	7	38,413,156	*HQNNA*	[6, 6, 4, 4, 4, 4, 4]	[6, 6, 4, 4, 4, 4, 4]	76.4%	169 MB
			CrossLight [44]	16	16	79.3%	650 MB
			HolyLight [111]	4	4	76.1%	162 MB
			LightBulb [138]	1	1	56.1%	41 MB
			ROBIN [44]	1	4	62.5%	48 MB
ResNet20	20	271,786	*HQNNA*	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4]	[4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 8, 6, 8, 10, 10, 10, 10, 10, 8]	79.7%	8.75 MB
			CrossLight	16	16	81.9%	70 MB
			HolyLight	4	4	77.6%	17.5 MB
			LightBulb	1	1	56.1%	4.4 MB
			ROBIN	1	4	64.2%	5.8 MB
CNN (SVHN)	7	552,362	*HQNNA*	[8,8, 4, 4, 4, 4, 4]	[8, 8, 4, 4, 4, 8, 4]	87.9%	34.4 MB
			CrossLight	16	16	86.2%	134 MB
			HolyLight	4	4	82.1%	32.4 MB
			LightBulb	1	1	29.4%	8.4 MB
			ROBIN	1	4	49.4%	9.8 MB

For FC layers, the matrix is comprised of bit-slices of individual weight values, which need to change with time steps so that the vector-matrix multiplication operation can happen in its entirety. For CONV layers the architecture performs a vector dot product operation, so the MVU can be used to represent all the bit-slices of one of the vectors simultaneously, across waveguides, to reduce the number of time slices needed for vector dot product operations. In both cases, the results per time slice need to be shifted and added. For FC layer operation, this can be done after the

summation operation as each waveguide generates partial sums for separate elements. For the CONV layer, the entire MVU generates a single convolution output. Because of this difference, we use separate MVU implementation for CONV and FC layers. In FC layers the shift and accumulate operation is done electronically, but for CONV layers, this can be done photonically.

For photonic shifting, we make use of Semiconductor Optical Amplifiers (SOAs) along with addition via Kirchhoff's Current Law (KCL) from the photodiode outputs. The SOAs can be fed a shift signal (σ), the gain-tuning signal, to reflect the shift operation. A 1-bit left shit can be performed photonically by multiplying with 2, which can be done through SOA-based gain tuning for 100%.

6.2.4. HQNNA ARCHITECTURE

The accelerator architecture, as shown in Figure 51, is composed of an array of MVUs, with input data routed through an electronic control unit. The MVU array is reused for CONV and FC layer activation. The vectors and matrices are mapped across the MVU array and the resulting partial sum vectors are summed digitally to obtain the sum vectors. For FC layers, each MVU considers an activation vector of size v and a $v \times v$ weight matrix simultaneously. Larger vectors and matrices are split up across different FC-MVUs for obtaining the vector to be passed to the next FC layer. The weight parameters have to be fed across *ceil(p/b)* time steps and a single activation vector slice has to operate on all the weight slices. This process has to be repeated *ceil(p/b)* times to obtain the final output vector. Thus an output vector of v size is generated every $(ceil(p/b))^2$ time steps. For CONV layer acceleration, the kernel, unfurled to a vector of size k, and its different bit-slices can be presented simultaneously to a k-element, activation vector slice. The activation vector slices, in turn, have to be presented to the kernel across *ceil(p/b)* time steps to obtain a single output vector element. The value of k is decided by the kernel sizes present in the

models and may be further decomposed across MVUs as dictated by laser power consumption constraints. As the value of k increases, the MR count, the waveguide length, and hence the laser power needed increases, which can be modelled using equation (37).

$$P_{laser} - S_{detector} \geq P_{photoloss} + 10 \times \log_{10} N_{\lambda} \quad (37)$$

where, P_{laser} is laser power in dBm, $S_{detector}$ is the PD sensitivity in dBm, N_{λ} is the number of laser sources, and $P_{photoloss}$ is the total photonic loss encountered by the signal. We considered photonic signal losses due to various factors: waveguide propagation loss (1 dB/cm), splitter loss (0.05 dB [173]), MR through loss (0.02 dB [124]), MR modulation loss (0.72 dB [125]), EO tuning loss (6 dB/cm [40]), and TO tuning loss (27.5 mW/FSR [39]). The value of v is more open-ended and needs to be optimized depending on the throughput analysis for FC layers across models. The DAC resolution, and corresponding power and latency, will be dependent on the b value being used. The best b value across models will also have to be optimized for throughput and energy efficiency (see Section 6.3). For CONV layer operations, we consider K Conv-MVUs and for FC layer operation, V FC-MVUs are considered.

Table 16 Parameters considered for architecture analysis

Devices	Latency	Power
EO tuning [40]	20 *ns*	4 μW/*nm*
TO tuning [39]	4 μs	27.5 mW/*FSR*
VCSEL [167]	0.07 *ns*	1.3 mW
Photodetector [130]	5.8 *ps*	2.8 mW
SOA [174]	0.3 *ns*	2.2 mW
DAC (16-bit) [149]	0.33 *ns*	40 mW
ADC (16-bit) [148]	14 *ns*	62 mW
DAC (8-bit) [168]	0.29 *ns*	3 mW
ADC (8-bit) [175]	0.82 *ns*	3.1 mW

6.3. EXPERIMENTS AND RESULTS

To evaluate the effectiveness of *HQNNA* we conducted several simulation-based analyses. For the CNN models, we consider the well-known models AlexNet [176] and ResNet20 [177] for CIFAR 10 dataset classification, along with a custom model for SVHN dataset classification. For power, energy and latency simulations, we designed a simulator for this architecture, using Python. For analyzing the model accuracy, we used Tensorflow v2.8 along with QKeras [121]. We compare the performance of our architecture in terms of energy-efficiency (energy-per bit, or EPB), throughput (giga-operations per second or GOPS), and throughput-energy efficiency (GOPS/EPB) against *CrossLight* [43], *HolyLight* [111], *LightBulb* [138], and *ROBIN* [44]. For obtaining optimal heterogeneous quantization for these models, we explored different algorithms as discussed in Subsection 6.2.2. The best configuration found using [169] was used for AlexNet, and ResNet20. For the SVHN CNN model, an exhaustive quantization search using AutoQKeras was performed. The resulting best heterogeneously quantized models and their parameters are presented in Table 15. This quantization exploration among the models is essentially a search for optimal p value in terms of accuracy and memory footprint. We also perform the various quantization techniques adapted in the works we compare *HQNNA* against. All these models are presented together in Table 15 to contrast the accuracy and memory footprint benefits heterogeneous quantization can provide.

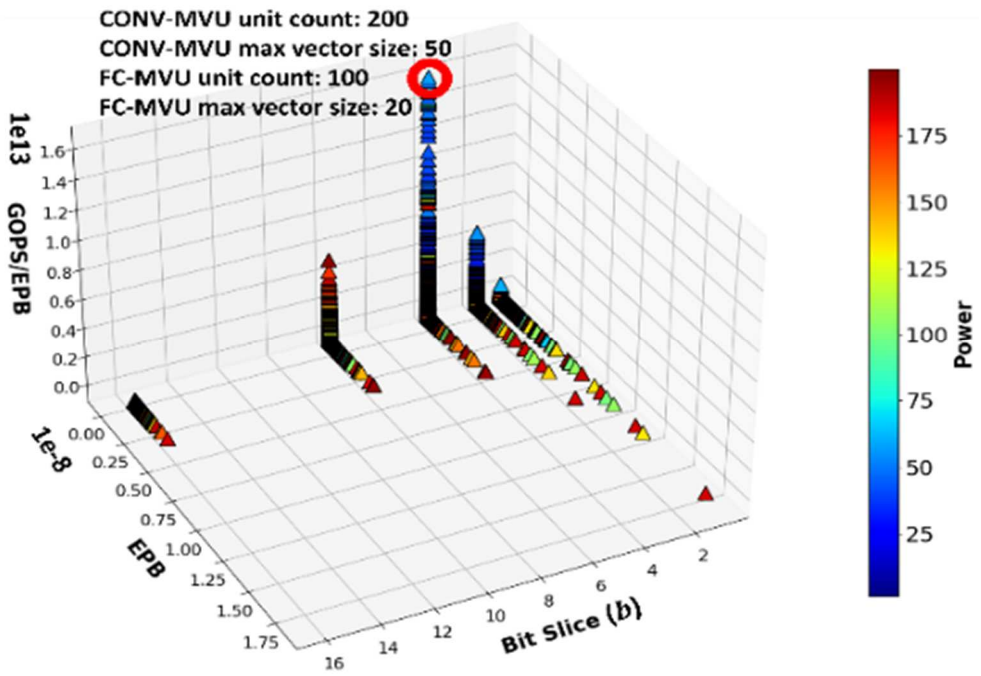


Figure 52. Architecture exploration for different *b* values, with the optimal architecture in terms of best GOPS/EPB identified.

The power and latency parameters used to model the architectures are shown in table 16. DACs with lower resolutions (1-bit, 2-bit, 4-bit) are not widely researched, possibly due to niche application spaces. For them, we have assumed the same latency as the design from [175]. We have also scaled DAC power for these lower resolution devices, with respect to resolution (N) using the following proportionality:

$$P_{DAC} \propto \left(\frac{2^N}{N} + 1\right) \tag{38}$$

In our first experiment, we optimize the *HQNNA* architecture, in terms of (*v, k, b, V, K*), where: *v* is the maximum vector granularity per FC-MVU; *k* is the maximum vector granularity per CONV-MVU; *b* is he bit-slice length and hence the DAC resolution; *V* is the total number of FC-MVUs; *K* is the total number of CONV-MVUs. We performed an exploration, for obtaining

optimal *v, k, b, V,* and *K* values, by performing and exhaustive sweep across possible values for these parameters, the results of which are shown in Figure 52.

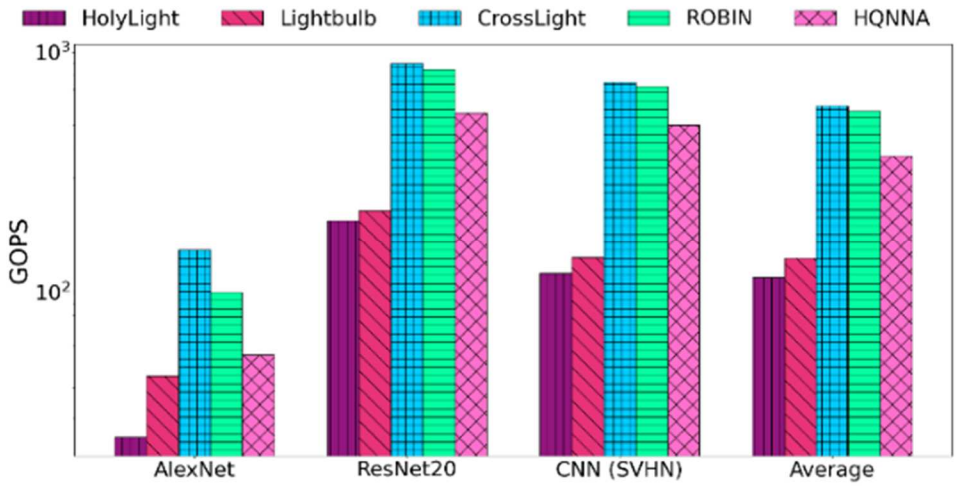


Figure 53. GOPS across models compared across architectures.

The best configuration was found in terms of throughput energy efficiency in terms of GOPS/EPB. The best (*v, k, b, V, K*) was found to be (50, 20, 4, 200, 100) for the models being considered. This configuration of *HQNNA* exhibits low maximum power consumption (57.5 W) due to the lower tuning power consumption (Subsection 6.2.2) and the lower DAC power consumption. However, as the GOPS results in Figure 53 shows, the TDM based approach causes a reduction in throughput as expected when compared to *CrossLight* and *ROBIN*. *HQNNA* has 2.33× and 2.25× lower GOPS than *CrossLIght* and *ROBIN* respectively. Due to the hybrid tuning approach with faster EO tuning being utilized for the MVU operations; *HQNNA* exhibits 2.2× and 1.3× higher GOPS than *HolyLight* and *LightBulb*, respectively.

Figure 54 shows the EPB comparison cross all architectures. The lower power consumption of *HQNNA* along with lower latencies of the lower resolution DACs being used enables this architecture to obtain lower EPB values as well. On average, *HQNNA* achieves 73.8×, 52.2×,

12.2× , and 3.59× lower EPBs than *HolyLight*, *LightBulb*, *CrossLight*, and *ROBIN* respectively as shown in Figure 54.

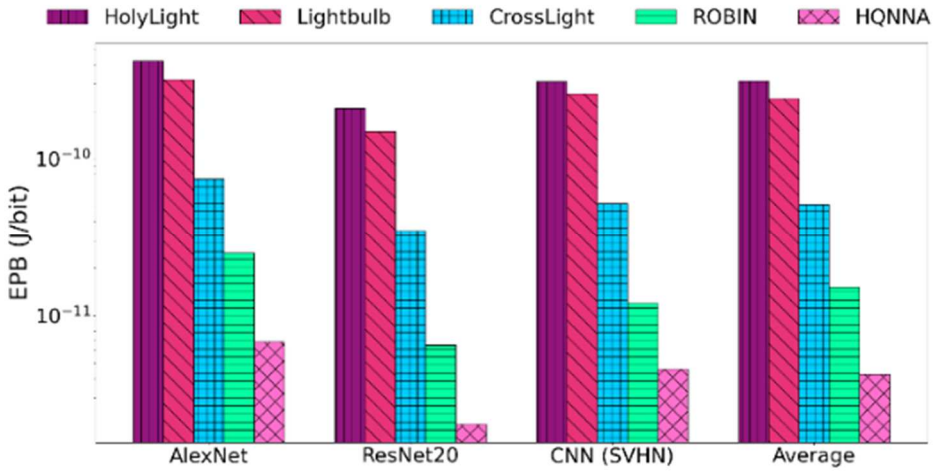


Figure 54. EPB across models, compared between this architecture and the considered optical architectures

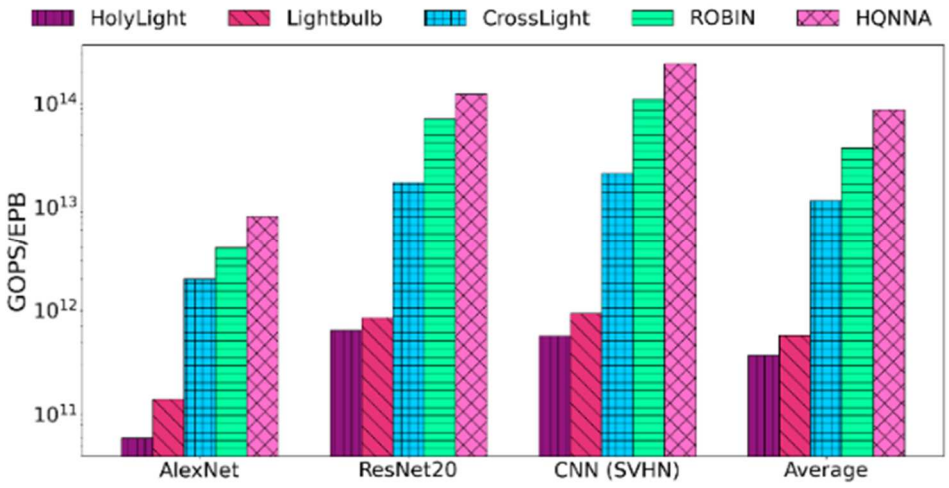


Figure 55. GOPS/EPB across models compared across architectures.

Finally, we compare our architecture against the other photonic accelerators in terms of GOPS/EPB. Despite having lower throughput than *CrossLight* and *ROBIN*, due to the significantly lower EPB, *HQNNA* exhibits significantly higher GOPS/EPB. Our architecture achieves 159.5× ,

103.1× , 28.6× , and 3.37× better GOPS/EPB than *HolyLight*, *LightBulb*, *CrossLight*, and *ROBIN* respectively (Figure 55).

6.4. CONCLUSION

In this chapter, we presented a novel non-coherent photonic CNN accelerator, *HQNNA*, which uses WDM and TDM simultaneously to efficiently accelerate heterogeneously quantized CNN models. Through identifying optimal quantization profiles for the CNNs and corresponding optimizations for hardware, *HQNNA* was able to achieve better performance in terms of energy- and throughput- efficiency: up to 73.8× better energy per bit and 159.5× better throughput-energy efficiency than conventional photonic CNN accelerators. Thus *HQNNA* represents a promising new substrate for energy-efficient acceleration of quantized CNN models.

CHAPTER 7: RECURRENT NEURAL NETWORK ACCELERATION USING NON-COHERENT PHOTONICS

Recurrent Neural Networks (RNNs) are a class of Artificial Neural Networks (ANNs) where connections among neurons form a directed graph along a temporal sequence. Such models have internal memory and feedback connections that make them well suited for learning trends and patterns inherent in sequences where the data elements are correlated. As a result, RNNs have been found to perform well for sequence learning tasks, such as speech recognition, human activity recognition, etc. [178]. While recent developments with Transformer models for sequential learning are promising, such models have large parameter counts that are not suited for resource-limited platforms [178].

When learning large sequences of data, simple RNNs [179] face the problem of vanishing gradients, which limits their usability. To alleviate the vanishing-gradients issue, more advanced RNN models have been developed based on Gated Recurrent Units (GRUs) [180] and Long Short-Term Memory (LSTM) [181]. These models are often employed in real-time scenarios, such as in IoT devices with virtual voice assistants and natural language processing abilities. Therefore, there is a critical need for efficiently accelerating such models for edge/IoT environments. However, inference acceleration in RNNs is a challenging task because of the recursive nature of these models and the compute-intensive operations required for large-dimensional sequence data. Moreover, RNNs are very reliant on the activation functions they employ, particularly the *sigmoid* and *tanh* functions. Thus, accelerating RNNs requires unique strategies that differ from those for accelerating other ANN models, such as MLPs and CNNs.

In recent years, several accelerators for RNNs have been proposed [182], [183], [184], [185], [7], [186]. Most of these efforts aim at accelerating a single RNN variant: LSTMs. However, other RNN models with simple RNNs and GRUs can be useful in resource-constrained scenarios. In particular, GRUs can offer comparable performance as LSTMs while offering faster execution and using less memory. In this chapter, we present the design of a novel RNN accelerator called RecLight which can accelerate ANNs that consist of any combination of simple RNNs, GRUs, and LSTMs. Unlike any prior RNN accelerator, we leverage noncoherent integrated silicon photonics. Silicon photonics is already a proven solution for high-throughput communication in the telecom, datacom, and rack-level computing domains, but in recent years it has also shown immense promise to accelerate computations [186]. The use of CMOS-compatible silicon photonic devices and circuits can overcome the energy and performance bottlenecks in conventional electronic accelerators. The novel contributions of this work are as follow:

- The design of a novel noncoherent silicon photonic accelerator targeting accelerating RNN variants;
- A detailed analysis of achievable resolution for RNNs with silicon-photonic microring resonator devices;
- A novel photonic multiply-and-accumulate (MAC) unit design that minimizes power dissipation and energy consumption while maximizing the overall throughput;
- A comprehensive comparison with state-of-the-art electronic RNN accelerators, for sequence learning.

The rest of the chapter is organized as follows. Section 7.1 presents a background on RNNs and their acceleration with photonic devices. Section 7.2 gives an overview of the RecLight

architecture. Section 7.3 discusses experimental setup and results, followed by the conclusions in Section 7.4.

7.1. BACKGROUND AND RELATED WORK

7.1.1. RNN ACCELERATION

RNN is a term used to denote any ANN model with feedback connections to the neurons in a layer. Such models are used for learning temporal dependencies between elements in a sequence, such as time series data. Due to the simplistic nature of the fundamental block in a simple RNN model, it is prone to exploding/vanishing gradients during training, which prevents the model from learning long-term dependencies in the input data [7]. To learn longer term dependencies, more complex RNN cells such as GRUs and LSTMs can be useful. Compared to simple RNNs, the gates and states used in GRUs and LSTMs make them effective for learning long-term dependencies. The individual cells are typically chained together within a layer, and multiple layers are often stacked together to realize powerful deep RNN models for sequence learning problems.

RNN accelerator-design efforts have mostly focused on LSTM acceleration, possibly owing to the increased popularity of the LSTM models over the other two RNN model variants. The work in [182] presented an FPGA implementation for LSTM acceleration using a software-hardware co-optimization approach. In [183], a similar FPGA implementation approach is used with a compression technique to accelerate LSTM inference. This approach employs block-circulant, instead of sparse matrices, to compress weight matrices. ASIC implementations of LSTM accelerators are proposed in [184] and [185]. In [184], approximate multiplication was employed along with synchronization of the proposed elastic pipeline to maximize the accelerator throughput. The architecture in [185] utilized systolic arrays for acceleration and to reduce

memory transfer overhead. Some recent FPGA-based implementations of GRU accelerators have been presented in [7] and [186]. *Unlike these efforts, RecLight supports accelerating all three major RNN variants.*

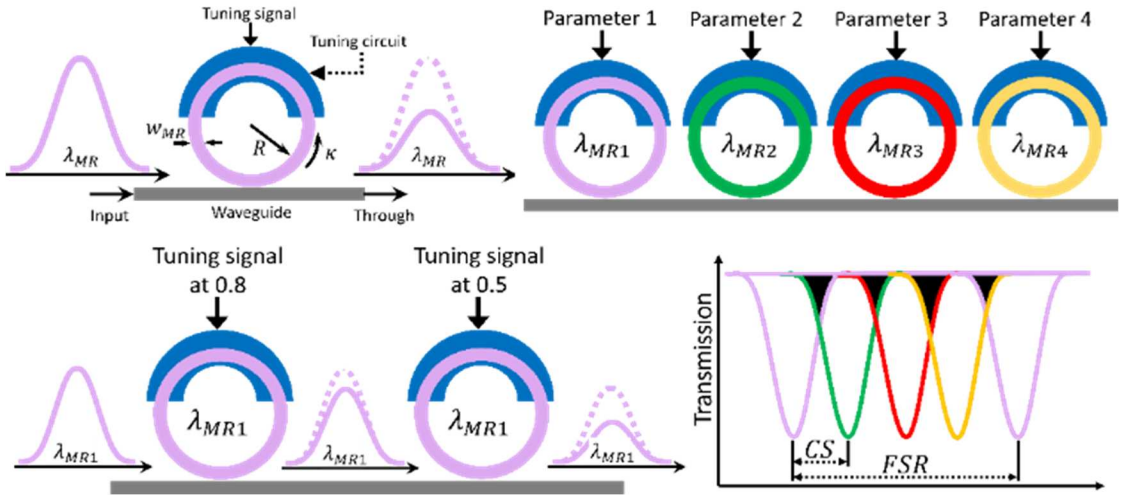


Figure 56. An MR with tuning circuit (top left) used for tuning wavelengths to reflect parameter values. Such MRs can be placed together to form an MR bank (top right). MRs of the same wavelength can be used to perform multiplication operations (bottom left). The transmission spectrum of an MR bank is shown on bottom right, depicting free-spectral range (*FSR*), channel spacing (*CS*), and inter-channel crosstalk (regions shaded black).

7.1.2 SILICON PHOTONICS FOR ANN ACCELERATION

Silicon photonics has already been established in literature as energy-efficient, high throughput solution for on-chip communication [42], [99]. Silicon-photonic ANN accelerators have received significant interest in recent years [37]. Optical ANN accelerators can be broadly classified into two types: coherent and noncoherent architectures. Coherent architectures use a single wavelength to operate and imprint weight/activation parameters onto the electrical field amplitude, phase, or polarization of an optical signal [113]. Here, the term *coherent* refers to the physical property of the wave with which it can interfere constructively or destructively on the same wavelength. Noncoherent architectures, such as [43], [44], [45], use multiple wavelengths, where each

wavelength can be used to perform computations in parallel. In these architectures, parameters are imprinted onto the signal amplitude and wavelength-selective devices, such as microring resonators (MRs; see Figure 56, top left), are used to manipulate individual wavelengths. Existing noncoherent photonic ANN accelerators primarily focus on accelerating CNNs and MLPs (see survey in [37]). *To the best of our knowledge, RecLight is the first RNN accelerator that leverages noncoherent silicon photonics.*

7.1.3. COMPUTATIONS WITH NONCOHERENT PHOTONIC DEVICES

Microring Resonators (MRs) are used as the primary optoelectronic device for computation in noncoherent architectures. As *RecLight* utilizes these devices, we provide a brief background on their operation. An MR is designed to be sensitive to a particular wavelength, called its resonant wavelength (λ_{MR}), which depends on multiple factors based on:

$$\lambda_{MR} = \frac{2\pi R}{m} n_{eff}, \tag{39}$$

where R is the radius of the MR, m is the order of the resonance, and n_{eff} is the effective refractive index of the device. An MR can modulate (transmit) electronic data over an optical signal λ_{MR} with the help of a tuning circuit that can alter n_{eff} in a carefully controlled manner. The MR tuning mechanism can induce an appropriate resonant shift ($\Delta\lambda_{MR}$), to change the output wavelength amplitude (Figure 56, top left) and realize a scalar multiplication operation. Such tuning is also used to imprint the desired parameters on an optical signal by adjusting an MR's tuning signal (corresponding to the parameter value), and hence varying the signal magnitude through the loss a wavelength experiences as it passes the MR. The tuning mechanism in MRs can be implemented via either microheaters (thermo-optic (TO) tuning [39]) or carrier injection (electro-optic (EO)

tuning [40]), thereby inducing a change in n_{eff}, which impacts λ_{MR}, and introduces the appropriate $\Delta\lambda_{MR}$.

he behavior of a large number of neurons can be emulated in noncoherent architectures by using wavelength-division multiplexing (WDM). To process multiple wavelengths simultaneously, several MRs can be placed together on the same waveguide to form an MR bank (Figure 56; top right). The number of wavelengths that can be accommodated with WDM depends on the free-spectral range (*FSR*) of the MRs. *FSR* is the spectral distance between two consecutive resonant peaks/modes of the *same* MR. To accommodate a large number of wavelengths, a large *FSR* is required. Moreover, to ensure reliable operation, channel spacing (*CS*), which is the spectral distance between two adjacent (different) MR resonances, must be sufficiently large (see Figure 56; bottom right). Low *CS* can cause power from adjoining resonances to leak into each other causing inter-channel or heterodyne crosstalk [53] (indicated by the regions shaded black in Figure 56, bottom right). The next section describes the *RecLight* architecture that addresses these challenges for reliable and high-performance photonic RNN acceleration.

7.2. RECLIGHT ARCHITECTURE

RecLight is a noncoherent photonic architecture that can accelerate inference with simple RNN, GRU, and LSTM based models. An overview of the *RecLight* architecture is shown in Figure 57 In the following subsections, we describe the *RecLight* architecture and the hardware optimizations we have considered to efficiently accelerate RNNs with *RecLight*.

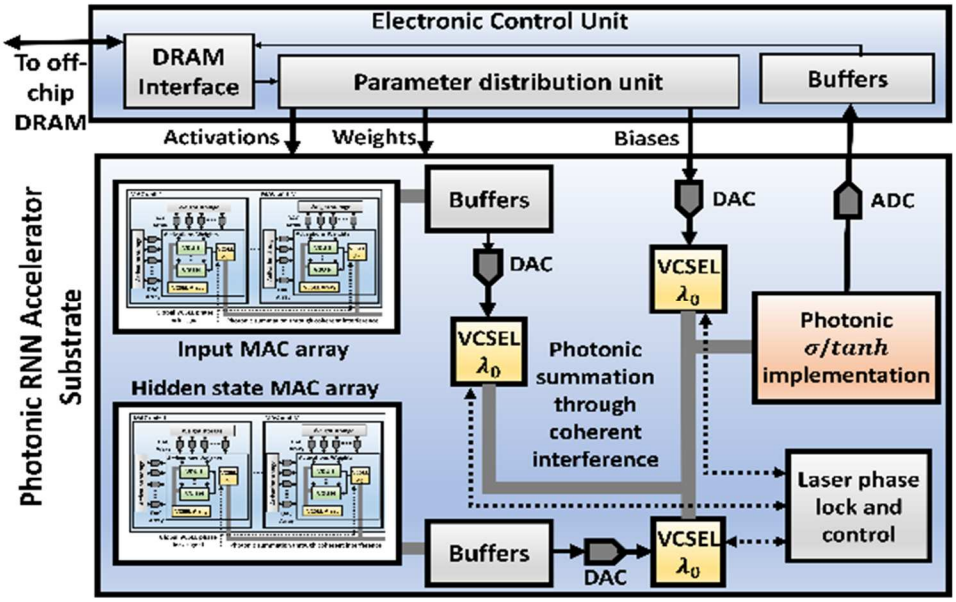


Figure 57. An overview of the proposed *RecLight* architecture.

7.2.1. MR TUNING CIRCUIT DESIGN

RecLight makes use of a hybrid tuning circuit where both TO and EO tuning are used to induce $\Delta\lambda_{MR}$. EO tuning is faster (≈ns range) and consumes lower power (≈4 μW/nm), but with a smaller tuning range [40]. In contrast, TO tuning has a larger tunability range, but consumes higher power (≈27 mW/*FSR*) and has higher (≈μs range) latency [39]. The hybrid tuning approach considers the advantages that each tuning mechanism offers while covering for its disadvantages. The feasibility of such a hybrid tuning approach has previously been shown in [117] for silicon photonic devices with low insertion loss. We use this approach for hybrid tuning of MR banks in our architecture. The approach supports efficient operation of MRs with fast EO tuning to quickly induce small $\Delta\lambda_{MR}$ and using the slower TO tuning infrequently for large $\Delta\lambda_{MR}$. To further reduce the power overhead of TO tuning in the hybrid approach, we adapt a method called thermal Eigenmode decomposition (TED), which was first proposed in [54]. Using TED, we can

collectively tune all the MRs in an MR bank with lower power consumption. TED also comes with the advantage of alleviating thermal crosstalk noise generated by heat dissipated from adjoining TO circuitries which use microheaters to induce thermal tuning.

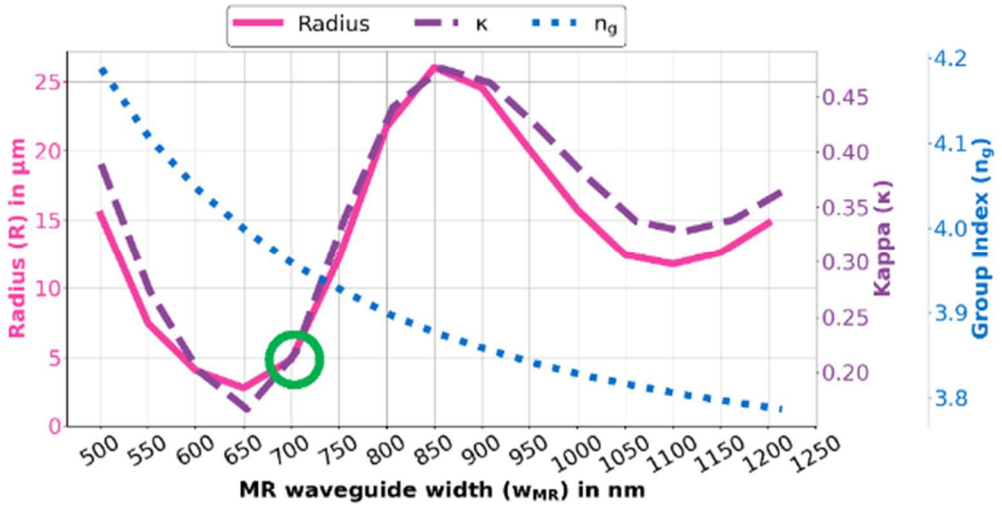


Figure 58. MR design exploration with the selected MR design (R=5 μm) highlighted by the green circle.

7.2.2. MR DEVICE DESIGN AND RESOLUTION ANALYSIS

By using TED and alleviating thermal crosstalk, which was pointed out to be the main constraint in parameter resolution achievable in noncoherent photonic computation in [187], we can achieve better resolution in *RecLight*. In addition, we consider the inter-channel crosstalk in an MR bank, using the analytical models from [187] (see Figure 56 bottom right).

As MR count increases, the resulting inter-channel crosstalk prevents good resolution from being achieved, at lower Q-factor values. At sufficiently high Q-factor values (9000 to 10000), even large MR banks can achieve 32-bit resolution, due to the sharper resonance (i.e., higher Q-factor) reducing crosstalk. But high-resolution support comes with the overhead of high-resolution digital-to-analog converters (DACs) and analog-to-digital converters (ADCs), which are power

hungry devices. Hence, we consider 16-bit resolution for our parameters, as 16-bit quantized models can achieve comparable performance to full-precision models [188]. Also, the large channel spacing and MR count in banks comes with the need for large *FSR* values, which are difficult to achieve. A larger *FSR* requires smaller radii which introduce higher optical losses in MRs. From our analysis, we found that with *CS* = 2.5 nm and Q-factor = 5000 in MRs, our MR banks can achieve a resolution of 16 bits with up to 15 MRs per bank. Using the models in [116], MR radius (*R*) can be described as:

$$R = \frac{Q\lambda_{MR}\kappa^2}{2\pi^2 n_g\sqrt{1-\kappa^2}}, \tag{40}$$

where κ is the coupling coefficient and n_g is the group index of the MR. We set Q at 5000, for our exploration presented in Figure 58. For λ_{MR} = 1550 nm, waveguide thickness of 220 nm, input waveguide width of 400 nm, and a gap of 100 nm, we performed an exploration for *R* and MR waveguide width (w_{MR}) while satisfying the Q-factor requirement of 5000. To obtain the corresponding κ and n_g values, we performed detailed device-level simulations with the ANSYS Lumerical tool [141].

The results from this experiment are presented in Figure 58. To avoid strong higher-order mode excitation when increasing w_{MR}, we selected w_{MR} and *R* to be 700 nm and 5 μm (green circle in Figure 58), respectively. Note that a smaller *R* will impose higher optical losses. The resulting FSR is 19.3 nm, which is sufficient for achieving our 16-bit parameter resolution goal in *RecLight*. The parameters obtained from this analysis are used to guide our architectural analysis, presented in Section 7.3.

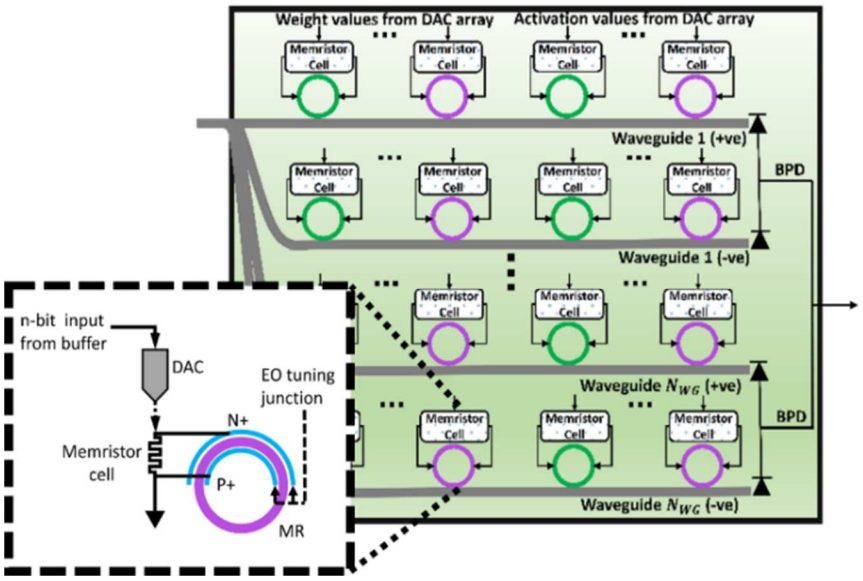
7.2.3. VDU AND MAC UNIT DESIGN

Effective ANN inference acceleration requires accelerating the most time-consuming operations during inference, which happen to be matrix-multiplication operations. This also holds true for RNNs, as most operations in RNN cells involve multiplication between matrices (of weights, inputs, etc.). These operations can be decomposed into vector-dot-product operations, as discussed for CNNs in [187]. The Vector-Dot-product Units (VDUs) in *RecLight,* as shown in Figure 59(a), are photonic computation units designed to perform vector-dot-product operations. RNN weights and activations are routed to individual MR tuning circuits using 16-bit DACs (to support 16-bit parameter resolution). To reduce the power consumption in the DACs, which can be substantial, we use a local parameter storage mechanism within the VDU that relies on memristors. A memristor cell is integrated into the EO tuning mechanism of an MR (see Figure 59(a)). The conductance of the memristor alters the biasing voltage being applied across the EO tuning junction in the MR. This conductance can in turn be tuned with an appropriate signal from the DAC. As the memristor can hold this conductance value once the voltage across it is removed, we can use the same DAC array to tune multiple MR banks. For this, we consider splitting the MR banks in a VDU across multiple waveguides (N_{WG}). If the VDU handles a vector granularity of v, this split allows us to use only $2v/N_{WG}$ DACs instead of the initial $2v$ DACs required. While this approach does incur some penalty in the form of slightly increased latency, the power benefits it brings far outweighs this penalty. The stored conductance in a memristor cell allows EO tuning to leverage the stored parameter to set the junction voltage across the tuning junction in the MR. Banks of such MRs within the VDUs perform the dot-product operations within the RNN cells mapped to them. These banks can also be tasked with accelerating fully connected (FC) layers which usually come after the RNN layers in deep RNN models used in many sequence-learning

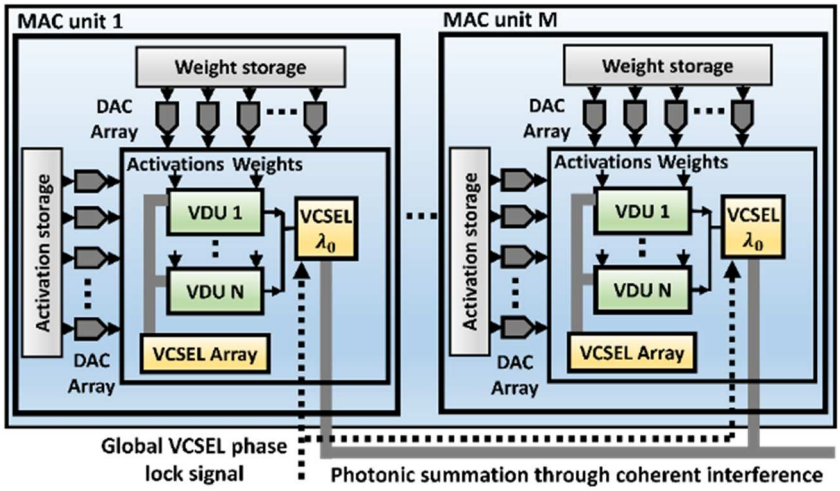
applications. To support both positive and negative values of parameters involved, we use separate positive and negative parameter arms in a VDU, for the same waveguide. The sum obtained from the negative arm is subtracted from the sum from the positive arm using a balanced photodetector (PD) arrangement, shown as BPD in Figure 59(a).

Multiple VDUs are combined to form a photonic Multiply and Accumulate (MAC) unit as shown in Figure 59(b). The VDUs in a MAC unit share the laser source and the DAC array between them. The laser sources we use in *RecLight* are vertical cavity surface-emission laser (VCSEL) [167] arrays. The shared VCSEL array allows for reusing the same wavelengths across multiple (N) VDUs, thereby reducing the VCSEL requirement and laser power consumption. This VCSEL-reuse also allows our architecture to attain the large channel spacing requirement (see Section 7.2.2) to attain 16-bit resolution. Splitting the MR banks across N_{WG} waveguides also helps further reduce the laser power consumption and possible inter-channel crosstalk. This split does incur a splitter loss (considered in our analysis), but the advantages it brings in terms of power consumption and robustness in operation are considerable.

To combine the partial sums generated by the MAC units, we employ coherent photonic summation. For this, we use an electrical signal from the VDU array to drive a VCSEL. Across MAC units, these driven VCSELs all generate the same wavelength λ_0 that, when introduced into the same waveguide, undergoes interference to generate the sum from a MAC unit array. To ensure coherent summation, we use a laser phase locking mechanism [189]. It ensures that VCSELs' output signals are in phase and hence constructive interference can occur. The output from the MAC unit array is added to the corresponding bias value optically, depending on which gate matrices were deployed. The bias value is fed directly to a λ_0 VCSEL, through a 16-bit DAC, for driving it, and photonic coherent summation is performed to obtain the summed output.



(a)



(b)

Figure 59. (a) VDU showing an MR bank with memristor cells for local parameter storage (BPD: Balanced photodetector). Inset: EO tuning control for memristor cell in VDU. (b) A MAC array comprised of *M* MAC units, each with *N* VDUs. Each MAC unit has a vertical cavity surface-emission laser array (VCSEL array) driven using the output from the VDU array.

7.2.4. IMPLEMENTATION OF THE NON-LINEAR UNIT

RNN cells require specific non-linear activation functions (*sigmoid* and *tanh*). While most photonic ANN accelerators assume that activation functions are implemented electronically [186],

this can lead to high overhead due to frequent opto-electronic conversions that would be needed for each RNN cell. To reduce such overhead, we consider an optoelectronic implementation of the activation functions.

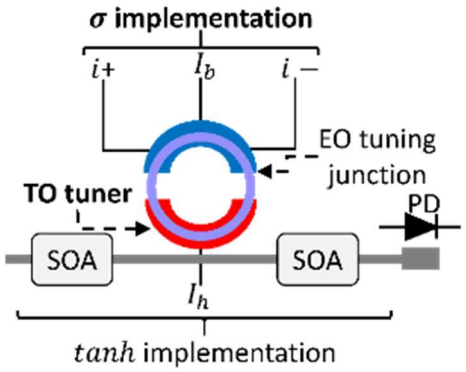


Figure 60. *Sigmoid* (σ) [190] and *tanh* implementation for *RecLight*.

The work in [190] implemented non-linear functions such as *sigmoid* (σ) using silicon photonic components (see Figure 60). In [190], a photonic control unit is used to drive $i+$ and $i-$ signals that are fed to the EO tuning circuitry of the MR. I_b and I_h are applied to, respectively, the EO tuning and TO tuning in the MR. But in our architecture, the required saw-tooth waveform signals can be generated by a more efficient electronic circuit as we only need to generate σ. Note that *tanh* can be also implemented based on σ (for input signal x) as:

$$tanh(x) = 2\sigma(2x) - 1. \quad (3)$$

To implement these activation functions, we use two Semiconductor-Optical Amplifiers (SOAs) [191], each providing a 100% gain to the input signal (Figure 60). The stored result is fed to a power gated electronic subtractor circuit to obtain the *tanh* value. The circuit in Figure 60 can be reconfigured to implement both σ and *tanh*, as enabling SOAs and the subtractor circuit will generate the *tanh* function and σ otherwise.

7.2.5. RECLIGHT ARCHITECTURE

As shown in Figure 57, the architecture of *RecLight* is designed to accelerate all the three RNN variants, including simple RNNs, GRUs, and LSTMs. Each VDU (see Figure 59(a)) in the architecture is assigned vectors with vector granularity of v to operate on. N VDUs along with their respective shared VCSEL array and BPDs, and λ_0 VCSELs (for summation) form a single MAC unit (see Figure 59(b)). Each MAC unit has its own local weight and activation parameter storage and associated DAC array. Each DAC array holds v 16-bit DACs to feed the parameters to one VDU at a time. M MAC units form a MAC array. Each MAC array is tasked with an RNN cell gate-level matrix multiplication. Each type of RNN is composed of temporal iterations of fundamental cells, each of which has gates associated with it. To accelerate an RNN, this fundamental cell operation and the associated gate operation must be accelerated. Our MAC units are designed to take into account the sequential nature in which the gate level RNN operations are performed.

Specifically, each gate operation requires an input state MAC operation and a hidden state MAC operation. Our architecture has MAC arrays specifically assigned for input and hidden state MAC-operation acceleration. In an RNN cell, two weight matrices, i.e., the hidden state vector and the input vector, along with the corresponding gate's bias vector are involved in a gate-level operation. To reflect this, two MAC arrays, each handling one of the two matrices, are designed with M MAC units each. The outputs from the two MAC arrays are photonically summed, to which the bias parameter can be added photonically without any electrical-to-optical conversion. The coherent photonic summation also allows us to subject the overall sum to the photonic non-linearity implementation. The non-linearity being used depends on the gate being operated on (Section 7.2.4). The result is collected in a storage unit where minor post-processing is performed

if needed. In this manner, layers of RNNs can be processed in *RecLight*. Moreover, fully connected (FC) layers (found in some deep RNN models) can also be accelerated by decomposing and mapping them to the VDUs in the architecture.

7.3. EXPERIMENTS AND RESULTS

To evaluate the effectiveness of *RecLight*, we performed several simulation-based analyses. We consider three datasets to build RNN models: a time series analysis based on the weather dataset from [192], the IMDB sentiment analysis dataset, and the Penn Treebank (PTB) dataset for language modeling. We designed an RNN, GRU, and LSTM based ANN model each for these datasets, details of which are provided in Table 17.

We designed a *RecLight* simulator in Python to estimate performance and energy costs, by modeling the microarchitecture of the MAC units as described in Section 7.2.3. The simulator performs layer-wise decomposition of RNN parameters into vectors, mapping them onto the modeled MAC units, and analyzes latency and energy consumption for the mapped operations. We parameterized the energy and latency requirements of the devices, as per the parameters presented in Table 18, which are based on fabricated silicon photonic devices.

We used Tensorflow 2.3 with Qkeras [121] for analyzing model accuracy across different parameter resolutions. From our analysis, the 16-bit quantized RNN models, as they are deployed in our architecture, perform with comparable accuracies to models with full precision (32-bit) parameters, as can be seen from Table 17. Table 18 shows the optoelectronic parameters considered for the simulation-based analysis with *RecLight.*

Table 17 RNN models considered for analysis.

Weather data time series prediction			
Model	Total parameters	MAE (32-bit)	MAE (*RecLight*)
RNN	152,976	0.4820	0.489
GRU	170,880	0.5782	0.5844
LSTM	217,696	0.5621	0.5650
IMDB sentiment analysis			
Model	Total parameters	Accuracy (32-bit)	Accuracy (*RecLight*)
RNN	2,216,137	73.8%	72.75%
GRU	2,691,713	75.3%	74.7%
LSTM	3,156,236	77.3%	76.8%
PTB dataset for language modelling			
Model	Total parameters	Perplexity (32-bit)	Perplexity (*RecLight*)
RNN	11,015,000	131.45	131.63
GRU	13,952,000	97.7	98.5
LSTM	14,615,000	66.02	65.78

Table 18 Parameters considered for analysis of *RecLight*.

Devices	Latency	Power
EO Tuning [40]	20 ns	4 μW/ nm
TO Tuning [39]	4 μs	27.5 mW/*FSR*
VCSEL [167]	0.07 ns	1.3 mW
Photodetector [130]	5.8 ps	2.8 mW
DAC (16 bit) [193]	0.33 ns	40 mW
ADC (16 bit) [148]	14 ns	62 mW
Memristor cell [163]	0.1 ns	0.07 μW

As discussed in Section 7.2.5, *RecLight* design involves parameters *v* (vector granularity), *N* (number of VDUs per MAC unit), *M* (number of MAC units), and N_{WG} (number of waveguides in a VDU). We performed an analysis to determine the best [*v, N, M,* N_{WG}] configuration possible for *RecLight* in terms of throughput (giga-operations-per-second (GOPS)) and energy-efficiency (energy-per-bit (EPB)). The result of this exploration is presented in a scatterplot in Figure 61. From this exploration, we can identify the *RecLight* architecture configuration with the best EPB/GOPS ratio, across all the models considered, with the configuration [15, 15, 40, 10] shown by the pink star in Figure 61. This *RecLight* configuration is used for further analyses.

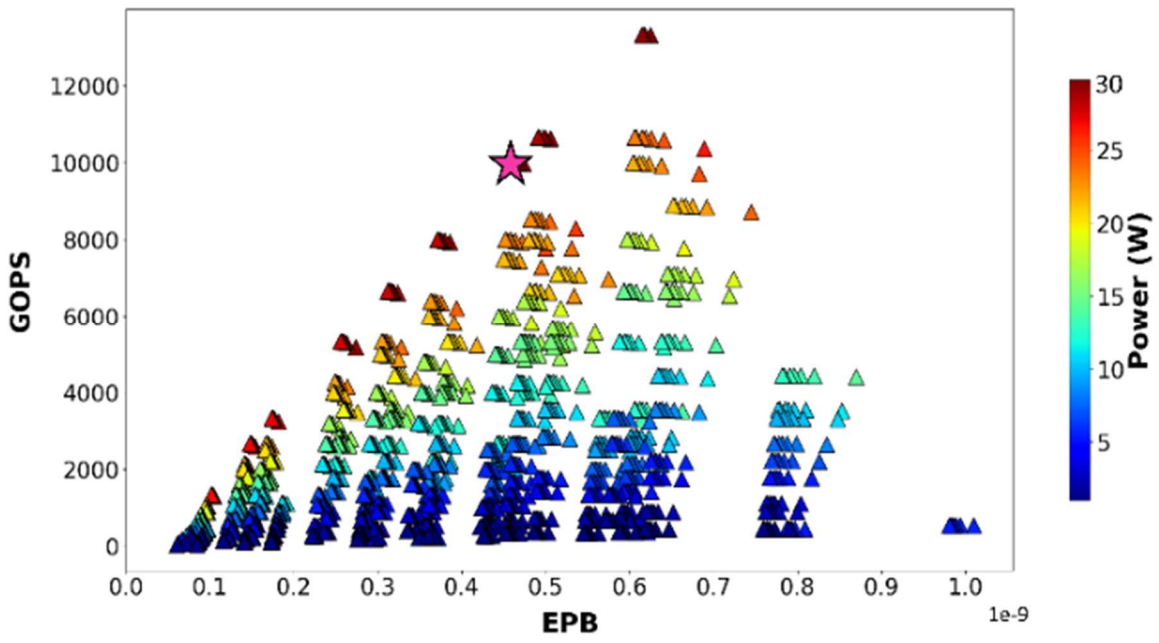


Figure 61. Architectural exploration analysis for *RecLight*, with the aim to find the optimal [v, N, M, N_{WG}] configuration with the best energy-efficiency and throughput. The best configuration, which is [15, 15, 40, 10], has the lowest EPB/GOPS value and is indicated using a pink star.

7.3.1. COMPARISON TO STATE-OF-THE-ART RNN ACCELERATORS

To analyze how *RecLight* compares to other accelerators when executing RNN models, we compare it against state-of-the-art electronic RNN accelerators: BBSL [182], C-LSTM [183], ELSA [184], and Chipmunk [185], which are LSTM accelerators, and with DeltaRNN [7] and EdgeDRNN [186], which are GRU accelerators. We do not show comparison results with other photonic accelerators as there is no prior work on *noncoherent* photonic RNN accelerators. We used energy and performance information as reported in the selected accelerators in our analysis to estimate the EPB and GOPS metrics for each accelerator, when executing the models described in Table 17.

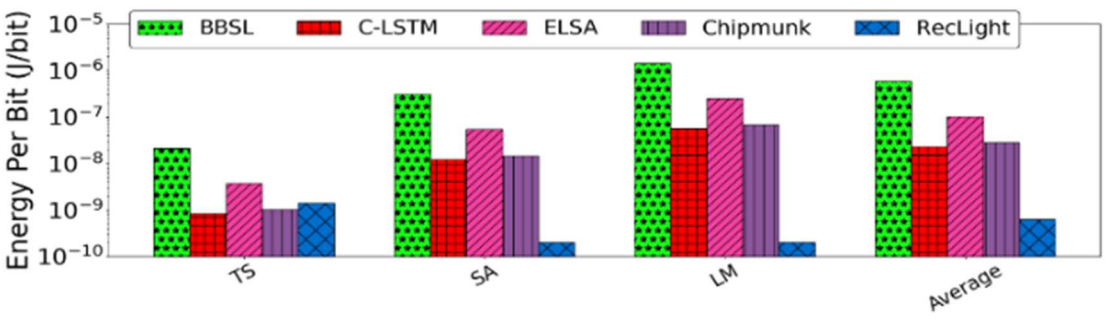


Figure 62. EPB comparison between LSTM acceleration. TS = time series, SA=Sentiment analysis, and LM= language modeling.

Figure 62 illustrates an energy-per-bit (EPB) comparison between the *RecLight* and the LSTM accelerators considered. We have not considered simple RNN and GRU model acceleration on the four accelerators from prior work as they are not designed to support these models. From the results, *RecLight* shows much lower EPB for LSTM acceleration. This is in part because of the low power consumption our accelerator achieves due to our device, circuit, and architecture level optimizations discussed in Section 7.2, and due to the low latency operation of the photonic substrate. *RecLight* does show higher EPB for the time series (TS) LSTM model as the model is simpler (see Table 17) and does not allow amortizing the static power overhead in our architecture. On average, *RecLight* obtains 956×, 37×, 167×, and 45× lower EPB than BBSL, C-LSTM, ELSA, and Chipmunk accelerators, respectively.

Figure 63 shows an EPB comparison between the GRU accelerators DeltaRNN [7] and EdgeDRNN [186], and *RecLight* running GRU models for inference (see Table 17). An EPB trend similar to what is shown in Figure 62 can be observed here again for *RecLight*, for the same reasons discussed earlier. From our analysis, *RecLight* obtains 1730× and 570× better EPB than DeltaRNN and EdgeDRNN accelerators, respectively.

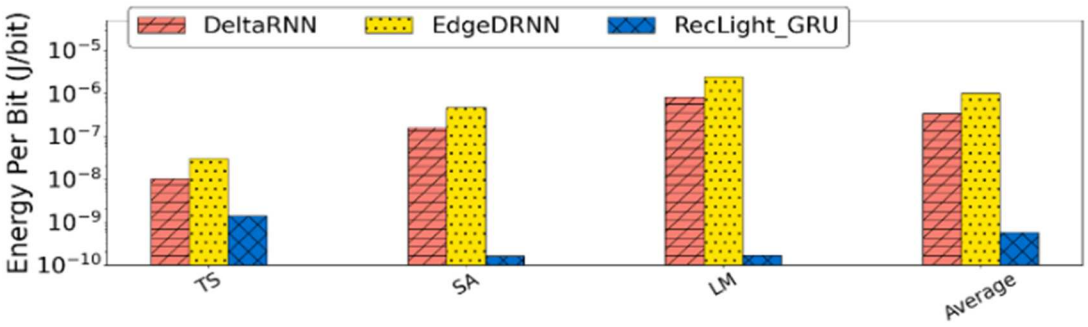


Figure 63. EPB comparison between GRU acceleration. TS = time series, SA =Sentiment analysis, and LM= language modeling.

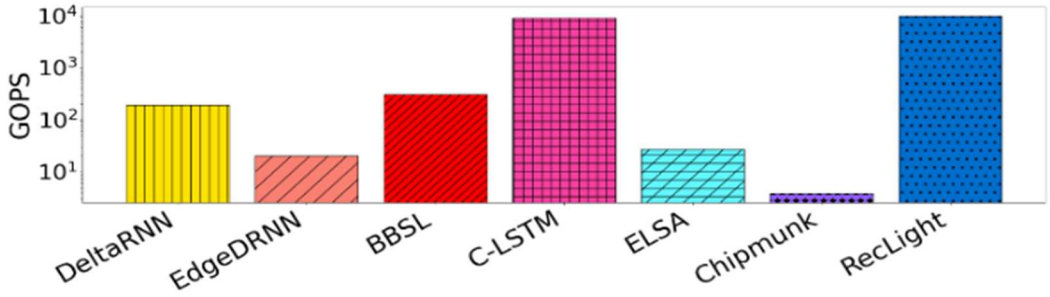


Figure 64. Throughput comparison among accelerators.

Finally, Figure 64 shows the GOPS comparison across all the accelerators. *RecLight* achieves 51.9×, 494.25×, 33.3×, 1.1×, 370.4×, and 2631.6× better throughput (y-axis is in log scale) in terms of GOPS compared to the DeltaRNN, EdgeDRNN, BBSL, C-LSTM, ELSA, and Chipmunk, respectively. The higher GOPS with *RecLight* can be attributed to its high-speed photonic computation with very few intermediate optical-to-electrical conversions.

7.4. CONCLUSIONS

In this chapter, we presented the first noncoherent photonic accelerator for RNN models, called *RecLight*. Our accelerator exhibits energy-per-bit improvements that range from 37× to 1730× when compared with six state-of-the-art electronic RNN accelerators. *RecLight* also demonstrates

up to 2631.6× better throughput than these electronic RNN accelerators. These results demonstrate the promising low-energy and high-throughput inference acceleration capabilities of our *RecLight* architecture. While in this work we focused entirely on the optoelectronic hardware design of our accelerator, with better software techniques for compressing RNN models, even better throughput and energy-efficiency improvements might be achievable with silicon-photonic-based accelerators.

CHAPTER 8: GRAPH NEURAL NETWORK ACCELERATION USING NON-COHERENT PHOTONICS

Deep learning has become a vital pillar in our lives due to its ability to solve many complex problems efficiently across diverse fields, including autonomous transportation, healthcare, industrial automation, and network security. This success of deep learning owes tremendously to the evolution of neural networks variants that are tailored for specific learning tasks. For example, Convolution Neural Networks (CNNs) [120] and Recurrent Neural Networks (RNNs) [179] have proven their efficiency in pattern recognition for images and sequence data, by extracting knowledge from the spatial and temporal dimensions of data. While these examples are prominent solutions for many tasks, they are limited in scope to non-arbitrary structured and Euclidean data. Arbitrary structured data, including graphs, require different techniques for efficient processing. Graph data processing is critical for many problems, e.g., social network analysis, recommender systems, and drug discovery [194].

Graph Neural Networks (GNNs) have emerged in recent years and established their proficiency in dealing with graph-structured data. These models can extract information from the graph structure and discover patterns in the data that may be difficult to identify with other deep learning methods [195]. Accordingly, many applications now benefit greatly from GNNs, and hence a lot of recent efforts have focused on enhancing GNN algorithms and improving their efficiency in handling large and various graphs. The continuing progress of GNN algorithms and models necessitates hardware platforms capable of providing GNN-specific support with high performance, while abiding by strict power constraints. Although hardware acceleration for neural networks such as CNNs and RNNs have been extensively studied, the processing of GNNs presents unique challenges due to their combination of dense and vastly sparse operations,

diversity of input graphs, and the various types of GNN algorithms and models [196]. Thus, hardware accelerators tailored to accelerate the processing of conventional neural networks cannot be directly and efficiently applied to GNNs.

Moreover, relying on traditional electronic accelerators creates limitations as these platforms face challenges in the post-Moore era due to high costs and diminishing performance improvements with semiconductor-technology scaling. Moving data through metallic wires is a well-known bottleneck in these accelerators, as it restricts the achievable performance in terms of bandwidth, latency, and energy efficiency [197]. Silicon photonics technology provides a promising solution to this data-movement bottleneck, offering ultra-high bandwidth, low-latency, and energy-efficient communication [32]. Optical interconnects, which are now being considered for chip-scale integration, have already replaced metallic ones for light-speed data transmission at almost every level of computing. It is also possible to use optical components for computations, such as matrix-vector multiplication [43]. The emergence of chip-scale optical communication and computation has thus made it possible to design photonic integrated circuits that offer low-latency and energy-efficient optical-domain data transport and computation. Furthermore, prior work, from both academia and industry, has demonstrated the significant benefits resulting from using silicon photonics for the acceleration of neural networks, as in [43], [110], [47], [49], [45], [44].

In this chapter, we introduce *GHOST*, the first silicon-photonic-based GNN accelerator that can accelerate inference of diverse GNN models and graphs. The key contributions in this chapter are:

- The design of a novel GNN accelerator hardware architecture using silicon photonics with the ability to accelerate multiple existing variants of GNN models;

- Detailed photonic device and circuit-level optimizations to mitigate crosstalk noise so that error-free GNN operations can be ensured in the accelerator;
- A detailed architectural optimization for the efficient handling and acceleration of diverse graph structures and GNN model architectures on the proposed hardware accelerator; and
- A comprehensive comparison with GPU, TPU, CPU, and state-of-the-art GNN accelerators.

The rest of the chapter is organized as follows. Section 8.1 provides a background on GNNs (different models, their acceleration challenges, and previous efforts on GNN acceleration) and on silicon photonics and performing optical computations. Section 8.2 describes the *GHOST* architecture and our optimization efforts at the device, circuit and architecture layers. Details of the experiments conducted, simulation setup, and results are presented in Section 8.3. Lastly, Section 8.4 presents concluding remarks.

8.1 BACKGROUND

8.1.1 GRAPH NEURAL NETWORKS

Prior to the emergence of GNNs, graph processing was mostly limited to traditional machine learning and graph algorithms. However, these methods had limitations in capturing the non-linear and complex relationships between vertices in a graph [195]. With the advent of GNNs, graph processing has been revolutionized, and there has been a significant improvement in graph-based machine learning tasks, such as node classification, link prediction, and graph classification. GNNs are a type of deep learning algorithm that can learn complex graph structures and relationships, and have now broadened the scope of Artificial Neural Networks (ANNs) to encompass non-Euclidean and irregular data found in graphs [196].

GNNs exploit the connections within a graph to understand and represent the relationships between vertices. They utilize an iterative approach that relies on a graph's structure and take in edge, vertex, and graph feature vectors that represent the known attributes of these elements. The general operations of a GNN can be broadly summarized in three main steps:

1) *Pre-processing:* an optional initial step that is typically performed offline for purposes such as sampling the graph, rearranging the graph to simplify the algorithm's processing and complexity, or encoding the feature vectors.

2) *Iterative updates:* the step where the main GNN computations occur through two main phases: aggregation and combination. The aggregation phase accumulates all the edges in a graph, and then for each vertex, it reduces all its neighbors and its own feature vectors into a single set. This feature set is combined and through performing linear transformations and non-linear activation functions, a new updated feature vector for each vertex is obtained. GNNs can be composed of several layers and the iterative process in a single layer updates every edge and vertex with information received from immediate neighbor vertices. This means that the relationships with nodes and edges that are progressively farther away can be gradually considered as more layers are processed.

3) *Readout*: the final step employed when a graph possesses a global feature vector, and it is updated once after the edge and node updates have been executed, usually in graph classification tasks.

Figure 65 illustrates an example of processing the first layer in a GNN. As shown, the aggregation phase iteratively gathers the neighbors of each vertex and then reduces all data into a single vector, h_{vi}^{a}. The reduce operation in this phase can be a variety of arithmetic functions, e.g.,

summation, mean, or maximum. This vector is then passed through the combination phase, which usually involves a neural network. Unlike conventional ANNs, where each layer has a different set of weights, vertices in a GNN all share the same weights.

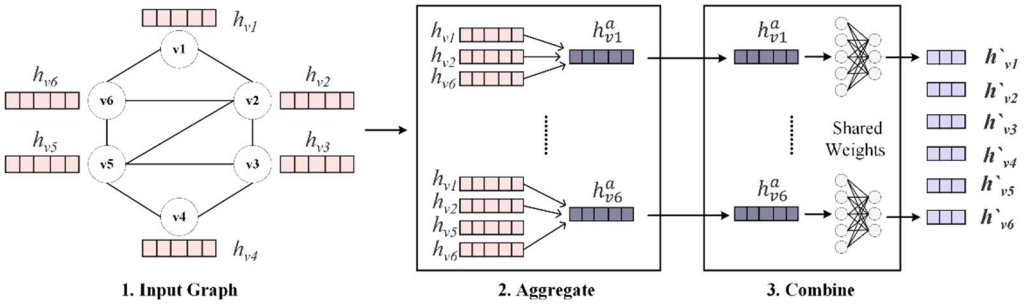


Figure 65. An example of GNN inference showing 1) Input graph to be processed; 2) Aggregation phase, where each vertex's neighbors are reduced to one feature vector; 3) Combine and Update phases, where each vertex is linearly transformed and updated using a non-linear activation function.

Since GNNs were initially introduced in [198], multiple GNN algorithms and models have emerged. Graph Convolution Network (GCN) [8] expands the idea of convolution in the graph space. In contrast to CNNs, where the convolutional operation is defined on regular grid-like data, the convolutional operation in GCNs is defined on irregular graph structures. GraphSAGE [199] and Graph Isomorphism Network (GIN) [200] are two models that are also based on graph convolutions. GraphSAGE employs custom sampling techniques to obtain a fixed number of neighbors for each vertex while GIN learns the isomorphism invariant representation of graphs by using a learnable parameter ε_l to adjust the weight of the central vertex. Graph Attention Networks (GATs) [9] are another class of GNNs that have demonstrated noteworthy results. GATs update node features through a pairwise function between nodes, which incorporates learnable weights. This results in an attention mechanism that can determine the usefulness of the edges.

8.1.2 GRAPH NEURAL NETWORK ACCELERATION

Processing GNNs presents many challenges. A system processing a GNN needs to have the capabilities to efficiently handle both dense and very sparse computations, adapt its execution and operations based on the specific input graph structure and the GNN algorithm variant employed, and scale effectively to extremely large graphs. Due to the irregularity and large size of most real-world graphs, GNNs often require very high memory bandwidth and multiple irregular memory accesses. Further, the unique combination of computing characteristics from deep learning and graph processing in GNNs results in having alternate execution patterns [196]. Such challenges are typically absent when processing traditional ANN models. Thus, utilizing ANN accelerators for GNNs can be inefficient and lead to low performance and high energy costs. While overcoming these challenges is a non-trivial task, many recent efforts have tackled this problem and advanced the field of GNN processing, as discussed below.

On the software side, several frameworks and graph libraries have been proposed to aid in the acceleration of GNN models. A few examples of relevant libraries are PyTorch Geometric [201], Deep Graph Library [202], and NeuGraph [203]. Several programming models that aim to abstract GNN operations have also emerged, such as SAGA [203] and GRETA [204].

On the hardware side, multiple electronic hardware accelerators for GNNs have been proposed. The accelerator in [205] presents a modular architecture where the core unit of the accelerator is a tile composed of an Aggregator module (AGG), a DNN Accelerator module (DNA), a DNN Queue (DNQ), and a Graph Processing Element (GPE). The main component in their AGG module is a bank of ALUs, and the DNA exhibits the architecture of existing spatial accelerators. Another electronic accelerator EnGN [206] has a unified architecture that handles GNNs in a single dataflow as a concatenated matrix multiplication of feature vectors, adjacency matrices, and

weights. An array of clustered PEs is utilized. To aggregate the results, each column of PEs is connected in a ring, and the results are passed along and added based on the adjacency matrix. HyGCN [207] is another electronic accelerator for GCNs, composed of two dedicated engines that handle the aggregation and combination stages, along a control mechanism that coordinates the sequential execution of both processes. The dense combination stage is computed using a conventional systolic array approach. In contrast, the aggregation stage has a more complex architecture that includes a sampler, an edge scheduler, and a sparsity eliminator. Lastly, GRIP [208] utilizes the GReTA programming model [204] to create an electronic accelerator with specialized units and accumulators for edges and vertices, which are separate and adaptable.

Several GNN accelerators based on ReRAM and Processing-In-Memory (PIM) have also been presented. For example, ReGNN [209] leverages Analog PIM (APIM) and Digital PIM (DPIM). The authors decompose the computations in the combination phase into multiple Matrix-Vector Multiplications (MVM) and handle them through a dedicated combination engine composed of an APIM ReRAM array, while non-MVM operations are processed by the DPIM array. ReGraphX [210] is another ReRAM-based architecture that can be used for both training and inference acceleration of GNNs.

Unlike previous efforts, *GHOST* is the first GNN accelerator that leverages silicon photonics. It also supports accelerating a broad family of GNN models, adapts efficiently to different graph shapes and sizes, and mitigates typical GNN memory and performance bottlenecks.

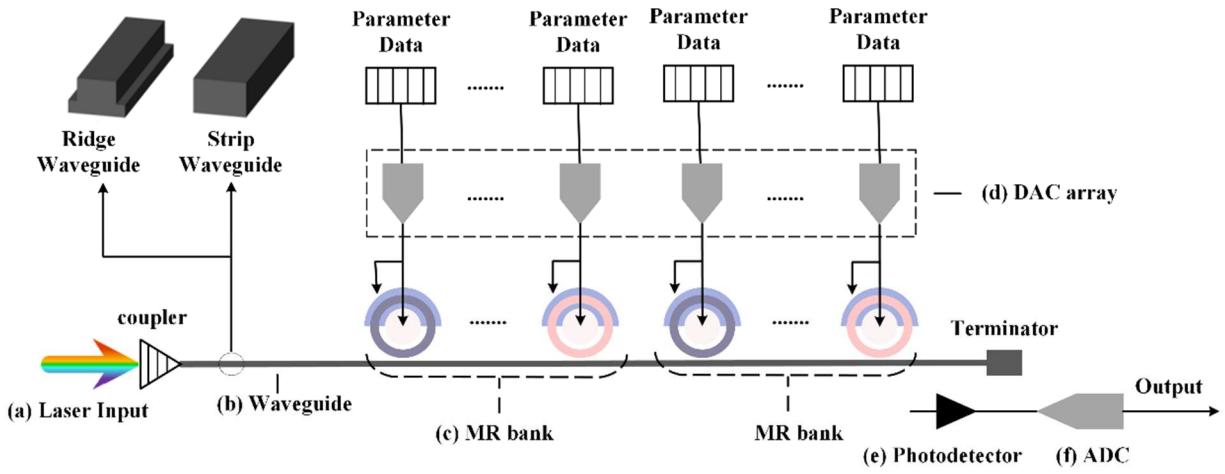


Figure 66. Overview of photonic circuit used to implement operations for ANN acceleration. This circuit is composed of (a) a laser source, which can be off-chip or on-chip; (b) a waveguide, which can be strip (for passive devices) or ridge (for active devices); (c) MR banks perform MAC operation; (d) the banks are tuned as per data from the electronic domain, using DACs; (e) the result from the MAC operation is detected and accumulated using a PD; (f) for converting the data to digital domain, for postprocessing or storage, an ADC can be used.

8.1.3 SILICON PHOTONICS

8.1.3.1 DEVICES AND CIRCUITS

Optical ANN accelerators have gained considerable attention from both academic researchers and industry in recent years because of their notable advantages in terms of energy-efficiency and performance [37]. There are two possible classes of optical ANN accelerators: coherent and non-coherent. In coherent architectures, a single wavelength is utilized to imprint parameters onto the optical signal's phase, which allows for Multiply and Accumulate (MAC) operations. Non-coherent architectures utilize multiple wavelengths and imprint parameters onto the optical signal's amplitude, enabling parallel operations to be performed using each wavelength [50]. As opposed to conventional compute platforms such as GPUs and CPUs, silicon photonic CMOS fabrication does not require advanced technology nodes which mandate elevated process complexity,

involving new lithography techniques and materials. Simple and less complex fabrication processes associated with older nodes are usually adopted instead [211]. The current focus of research in optical ANN accelerators is mainly on CNNs, MLPs, and RNNs. To the best of our knowledge, GHOST is the first optical accelerator for GNN models.

Figure 66 presents a general overview on the fundamental devices and circuits used for optical computing. The following are the main components needed:

- *Lasers*: used to generate optical signals that are needed to perform computation and communication in optical circuits. These lasers can either be on-chip or off-chip. While off-chip lasers have better light emission efficiency, there are significant losses when coupling the optical signals onto on-chip waveguides. Conversely, on-chip lasers, such as vertical cavity surface emission lasers (VCSELs), offer a higher level of integration density and lower losses.

- *Waveguides*: silicon photonic waveguides carry the optical signal(s) generated by the laser source. They are composed of two materials, resulting in a high-refractive-index contrast such as a core made of Silicon (Si) and a cladding made of Silicon Dioxide (SiO2). This allows for total internal reflection. The waveguides can be either ridge or strip in shape. Wave Division Multiplexing (WDM) allows a single waveguide to support multiple wavelengths simultaneously without any interference. This enables the transmission of ultra-high bandwidth signals and is used for performing MAC operations.

- *Microring Resonators (MRs)*: An MR add-drop filter is an optical modulator which is designed using a ring-shaped waveguide. Each MR can be specifically designed and adjusted to work at a particular wavelength, known as the MR resonant wavelength (λ_{MR}),

defined as $\lambda_{MR} = \frac{2\pi R}{m} n_{eff}$, where R is the MR radius, m is the order of the resonance, and n_{eff} is the effective index of the device. Electronic data can be modulated onto the optical signal passing an MR by carefully adjusting n_{eff} (and hence λ_{MR}) with a tuning circuit. MR banks consist of groups of MRs that share a single input waveguide and can be utilized for performing MAC and summation operations.

- *Photodetectors (PD)*: PDs are needed to detect processed optical signals and convert them into electrical signals. To be effective, a PD should be able to generate the desired electrical output using a small input optical signal. The input signal power from the laser source must be greater than the responsivity of the PD, while taking into consideration the different types of losses that may occur along the optical link.

- *Tuning circuits*: Tuning circuits are devices designed to control the effective index (n_{eff}) of MR devices to precisely modify an output optical signal. Typically, the tuning circuit is based on Thermo-Optic (TO) [39] or carrier injection Electro-Optic (EO) tuning [40], both of which cause a change in the effective refractive index (n_{eff}) and result in a resonant shift of $\Delta\lambda_{MR}$.

- *Digital-to-Analog Converters (DACs) and Analog-to-Digital Converters (ADCs)*: tuning the MR devices and converting the optical output to the electrical domain for intermediate buffering is all done using ADC and DAC devices. These devices represent one of the main performance bottlenecks in silicon-photonic-based systems due to their high latency and power costs. Accordingly, as will be discussed in Section 8.2, GHOST employs several techniques that aim to mitigate the downsides of these devices and reduce the needed opto-electric conversions.

8.1.3.2 OPTICAL COMPUTATION

Most of *GHOST's* core operations are performed using the opto-electronic tuning devices, MRs. Figure 67(a) shows a representation of the transmission plots for the input and the through ports' wavelengths after a parameter is imprinted onto the input signal. In most silicon-photonic-based systems, computations are performed as such by adjusting an MR's $\Delta\lambda_{MR}$, which leads to a predictable alteration in the amplitude of the optical signal's wavelength. GHOST leverages this to implement two main computations using MR devices: summation and multiplication. Summation is performed using coherent photonic summation. This entails using one optical signal with a single wavelength λ_{MR} and MR devices adjusted to operate at the same resonant wavelength λ_{MR}. Figure 67(b) illustrates an example where coherent summation is used to add the values a_1, a_2, and a_3. Using an analog biasing signal, VCSELs can be driven to produce an optical signal with a certain value imprinted onto it. Accordingly, the first value a_1 is imprinted onto the optical signal using the bottom VCSEL laser source. The top VCSEL produces an optical signal with a value of 1 which is split into two signals to be passed by the two MR devices. The first MR device then imprints the value a_2 onto the optical signal, while the second MR imprints the value a_3. When the optical signal generated by the bottom VCSEL unit ($a1$) and the one modulated by the first MR device ($a2$) meet, they undergo interference, resulting in a summation operation and an optical signal with the value $a1 + a2$ is generated. Similarly, when this optical signal meets the one modulated by the second MR device, they undergo interference, resulting in a summation operation and the final output $a1 + a2 + a3$ is computed. Coherent summation is ensured by using a laser phase locking mechanism [189], which guarantees that VCSEL output signals have the same phase for constructive interference to occur.

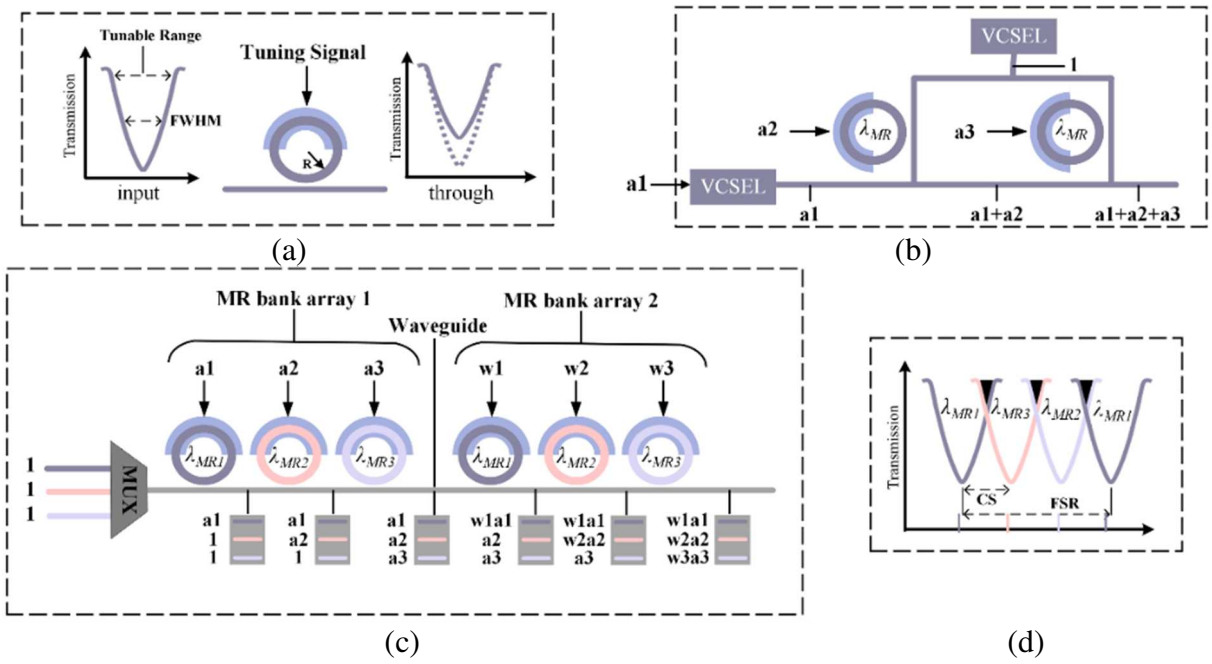


Figure 67. (a) MR input and through ports' wavelengths after imprinting a parameter onto the signal; (b) two MR devices used to perform optical coherent summation to add values a_1, a_2, and a_3; (c) MR bank arrays used to perform multiplication by imprinting input vector (a_1-a_3), followed by weight vector (w_1-w_3); (d) MR bank response and heterodyne crosstalk shown in black, where CS is channel spacing and FSR is free spectral range.

On the other hand, performing multiplications is done using non-coherent silicon photonics, where multiple optical signals with different wavelengths are multiplexed into the same waveguide using WDM. This enhances throughput and emulates neurons in ANNs as it involves combining multiple optical signals with different wavelengths into a single waveguide using an optical multiplexer [50]. Different wavelengths in the input waveguide pass through a series of MRs, with each MR tuned to a specific wavelength, allowing for several multiplications to be executed simultaneously in parallel. Figure 67 (c) illustrates an example of multiplying two vectors (activations vector A, and weights vector W) as follows

$$[a1 \quad a2 \quad a3] \times \begin{bmatrix} w1 \\ w2 \\ w3 \end{bmatrix} = a1w1 + a2w2 + a3w3 \tag{41}$$

Two MR banks, and three optical signals with different wavelengths are needed to perform this multiplication. The first MR bank array imprints the activation values $a1, a2,$ and $a3$, while the second MR bank imprints the weight values $w1$, $w2$, and $w3$. As shown in the figure, after imprinting the activation values, when the same signal with the same wavelength gets modulated by a second MR, a multiplication operation occurs between the previously imprinted value and the new one. Different optical wavelengths on the waveguide can then go through a PD to accumulate the output of the dot product. This method can be extended to perform MVMs since they can be decomposed into vector multiplications, by using several rows of the MR banks organization shown.

One of the main challenges of performing multiplications using non-coherent silicon photonics is the resulting heterodyne or incoherent crosstalk that is shown as the black portions in Figure 3(d). This occurs when a portion of an optical signal from neighboring wavelengths interferes with the MR spectrum of another wavelength. In Section 8.2.2, we discuss our efforts in optimizing the MR bank arrays design to reduce this crosstalk.

8.2 GHOST HARDWARE ACCEELRATOR

GHOST is a silicon photonic architecture that can accelerate the inference of a diverse family of GNN models. An overview of the architecture is shown in Figure 68. The photonic accelerator core is composed of aggregate, combine, and update blocks, enabling the execution of a wide range of GNN models and real-world graph datasets. Interfacing with the main memory, buffering the input graph, identifying the needed resources, and mapping the weight matrices to the photonic architecture are all handled by an integrated Electronic-Control Unit (ECU). The following subsections describe the *GHOST* architecture and the device, circuit, and architecture layer optimization solutions we have considered to efficiently accelerate GNN models.

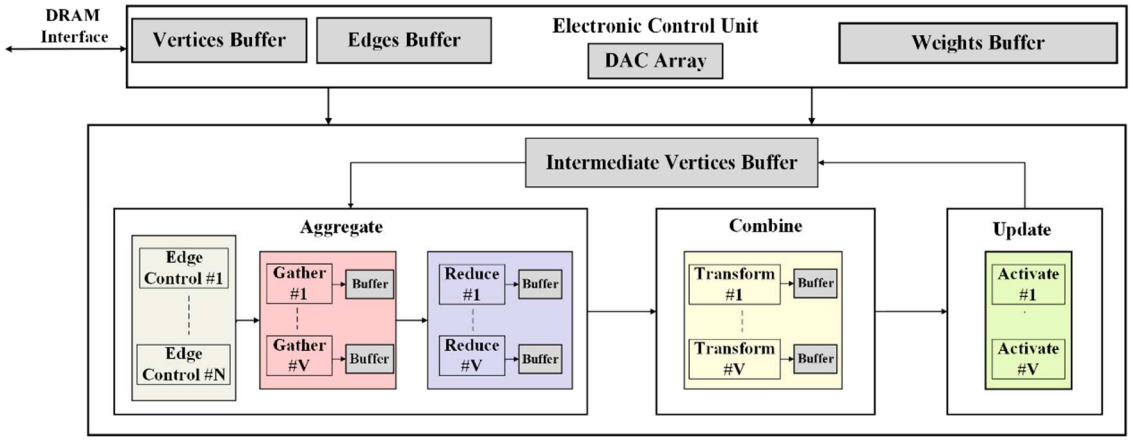


Figure 68. Overview of *GHOST* accelerator architecture showing the ECU, Aggregate, Combine, and Update blocks.

8.2.1 TUNING CIRCUIT DESIGN

As discussed earlier, MR devices require a tuning mechanism, which can be achieved using either EO or TO methods. In *GHOST*, we have implemented a hybrid tuning circuit that utilizes both methods to induce $\Delta\lambda_{MR}$. By doing so, we can capitalize on the advantages of each approach while mitigating their drawbacks. EO tuning is quicker (≈ns range) and consumes less power (≈4 μW/nm); however, it cannot be used for large tuning ranges [40]. Conversely, TO tuning offers a larger tunability range, but with the drawback of higher latency (≈μs range) and power consumption (≈27 mW/FSR) [39]. To address these challenges, we have integrated EO tuning to quickly induce small $\Delta\lambda_{MR}$ and reserved the slower TO tuning for cases where larger $\Delta\lambda_{MR}$ is required. The effectiveness of this hybrid approach has been previously demonstrated in [117]. To further reduce the power consumption of TO tuning and also reduce thermal crosstalk, we have employed the Thermal Eigenmode Decomposition method (TED) from [54]. We analyzed thermal interference between MR heaters and using Eigenmode decomposition, thermal tuning levels across heaters which do not cause thermal interference in MR banks were determined. Tuning the

MR heaters according to the tuning levels obtained with this approach allowed us to mitigate thermal crosstalk and minimize TO tuning power.

8.2.2 MR DEVICE OPTIMIZATION

To ensure that the MVM operations performed are error free, so that the deployed GNN can be executed correctly, it is necessary to manage various sources of noise in the analog photonic domain. There are several major noise sources in the photonic computing substrate, including thermal crosstalk, heterodyne (or incoherent) crosstalk, and homodyne (or coherent) crosstalk. The thermal crosstalk between TO tuning circuits is mitigated using our TED-based tuning mechanism (Section 8.2.1). But we still have to mitigate the impact of heterodyne and homodyne crosstalk in our design.

The presence of multiple wavelengths (or channels) in the same waveguide causes heterodyne or inter-channel crosstalk, where a portion of an optical signal from neighboring wavelengths (say λ_1 and λ_3) can leak into the MR spectrum of another wavelength (say λ_2) (see Figure 67(c)). This power leak causes MR2 output to have λ_1 and λ_3 content, and the MR downstream (in this example MR3), will receive lower signal power than designed for. Thus, the output from MR2 and MR3 will be erroneous. For this design optimization, we model heterodyne crosstalk using the following equations:

$$P_{signal} = \Phi(\lambda_i, \lambda_j, Q_{factor}) P_s(\lambda_i, \lambda_j), \tag{42}$$

$$P_{het\_noise} = \sum_{i=1}^{n} \Phi(\lambda_i, \lambda_j, Q_{factor}) P_S(\lambda_i, \lambda_j)(i \neq j), \tag{43}$$

where, Φ is the crosstalk coupling factor, i.e., the spectra overlap between the two neighboring wavelengths, Q_{factor} is the quality factor or Q-factor of the MR, and P_s is the input signal power to the MR.

Heterodyne crosstalk impacts signals with spectral overlap. To mitigate heterodyne crosstalk, this overlap should be minimized. This can be achieved by a well-designed channel spacing and Q-factor tuning while ensuring that the Signal-To-Noise ratio (SNR) in the output is higher than the photodetector sensitivity. Another factor to be considered is the tunable range of the designed MRs. The MRs should provide adequate Q-factor to improve SNR, but should also possess sufficient tunable range, i.e., 2×FWHM (FWHM=full width half maximum), so that necessary parameters can be imprinted error free. To mitigate heterodyne crosstalk, we optimize our design for high FWHM and SNR, where SNR is expressed as:

$$SNR = 10 \times \log_{10}(P_{signal}/P_{noise}), \tag{44}$$

and, given λ_{res} as the resonant wavelength of the MR being considered, FWHM can be modeled as:

$$FWHM = \lambda_{res}/Q_{factor}. \tag{45}$$

Homodyne or coherent crosstalk is a result of undesired mode coupling among signals of the same wavelength [116]. In some of the computation circuits in *GHOST* we rely on coherent signal processing. In such circuits, part of the signal on the same wavelength may leak through a device and experience a different phase. Such leaked signals interfere with the output signal (based on their phase difference with the output signal) as coherent crosstalk noise. The presence of homodyne crosstalk, similar to heterodyne crosstalk, impacts the SNR of the non-coherent optical circuitry. The homodyne crosstalk noise power can be modeled as follows:

$$P_{hom\_noise} = \sum_{i=1}^{n} P_{in} \cdot X_{MR}^{i}(\rho) \cdot L_{P}^{n-i}, \tag{46}$$

where P_{in} is the input optical power, $X_{MR}^{i}(\rho)$ is the crosstalk contribution from the i^{th} MR in a bank of n MRs, and ρ is the optical phase of the crosstalk signal, which is a function of the EO tuning voltage. ρ does not take into account phase errors from thermal crosstalk, as TED is employed to address those errors. Finally, L_P^{n-i} is the passing loss that the crosstalk signal experiences as it propagates through MRs in the coherent circuit.

For homodyne crosstalk mitigation we may increase the cross over coupling by increasing the gap between the input waveguide and ring waveguide. This reduces the amount of crosstalk signal being coupled over from the MR to the main waveguide, reducing the impact of crosstalk on the output signal. For achieving this, while meeting SNR constraints (discussed later) the Q_{factor}, attenuation in MR (a), and cross-over coupling coefficient (κ) has to be fine-tuned as follows [116]:

$$Q_{factor} = \frac{\pi n_g L \sqrt{(1-\kappa^2)a}}{\lambda_{MR}(1-a(1-\kappa^2))} \ . \tag{47}$$

Using our noise models described in (42), (43), and (46), we can identify the optimal design space for our MR banks which can ensure high SNR and a high tunable range (R_{tune}). We must also consider that the lowest optical power level (P_{lpar}) should be higher than P_{noise}.

$$P_{lpar} > P_{noise} \ , \tag{48}$$

$$\frac{P_{signal}}{P_{lpar}} < \frac{P_{signal}}{P_{noise}} , \tag{49}$$

$$10\log_{10}\left(\frac{P_{signal}}{P_{lpar}}\right) < 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right), \tag{50}$$

where P_{lpar} can be defined in terms of P_{signal}, from (42), as follows:

$$P_{lpar} = \frac{P_{signal} \times R_{tune}}{N_{levels}} . \quad (51)$$

Replacing P_{lpar} in (50) yields the following relation:

$$10 \log_{10}\left(\frac{N_{levels}}{R_{tune}}\right) < SNR. \quad (52)$$

Here, N_{levels} is the number of amplitude levels we need to represent across the available R_{tune}. For n-bit GNN parameter representation, N_{levels} will be 2^n. If positive and negative values are represented separately, as in the case with *GHOST*, then N_{levels} will be 2^{n-1}. The relationship in (52) can be rearranged to as follows:

$$\frac{2 \times \lambda_{MR}}{Q_{factor}} > N_{levels} \times 10^{-\frac{SNR}{10}} \quad (53)$$

Utilizing these models, we can identify the design space for our MRs and the MR banks they constitute, in terms of the Q_{factor}, N_{levels}, and SNR. We can obtain values for $\Phi(\lambda_i, \lambda_j, Q)$ in (41) and (42) and $X_{MR}(\rho) \cdot L_P^{n-i}$ in (45) through multi-physics simulations. The results from our exploration studies using our detailed models and the simulation tool suite from Ansys Lumerical [141] are presented later in Section 8.3.2.

8.2.3 GHOST ARCHITECTURE DESIGN

As illustrated in Figure 68, the main units in the *GHOST* architecture (aggregate, combine, update) are divided into V execution lanes. During inference, each lane is assigned one output vertex to process, in parallel with all the other lanes. The aggregate block gathers all neighbor vertices and the associated edge data, and performs a reduce function for each of the assigned output vertices. The combine block then applies a linear transformation on each aggregated vertex feature vector h_v^a. Finally, the update block applies a non-linear activation function to obtain the updated vertex feature vectors $h_v^{`}$. At the start of processing a GNN model, when processing the

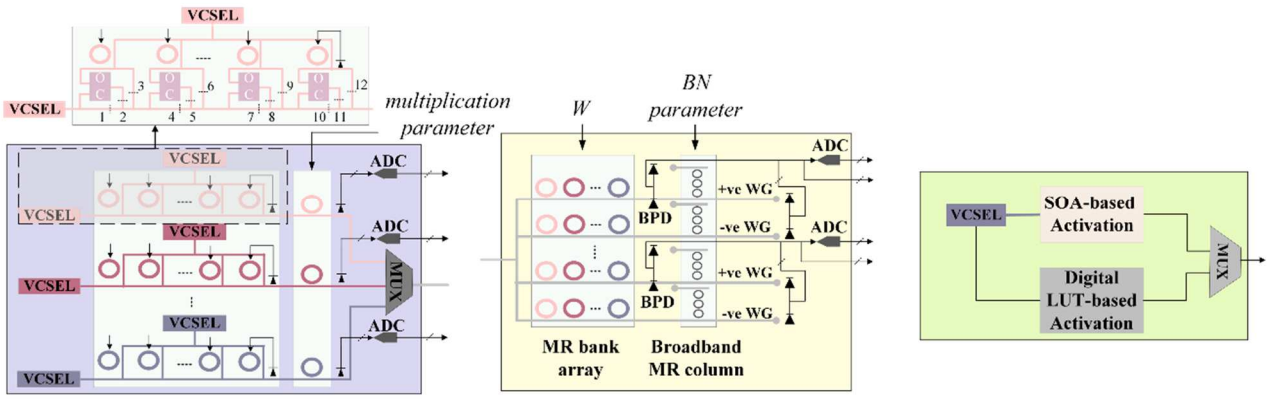
first layer, the aggregate block reads the graph data from the vertex and edge buffers in the ECU, which interface directly with main memory. As updated vertex data is computed, it is placed in the intermediate vertex buffer and thus, the aggregate block would read the vertex data when processing the next layers from the intermediate vertex buffer as needed.

8.2.3.1 AGGREGATE BLOCK

As most graphs can be extremely sparse and irregular, regulating their memory accesses to improve performance is a challenge. *GHOST* alleviates this bottleneck through employing a "buffer and partition" optimization technique which is explained in Section 8.2.4.1. In this technique, the source and destination vertices in the input graph are split into blocks of N and V. The aggregate block is thus composed of N edge control units, V gather units, and V reduce units. In each cycle, the V execution lanes in *GHOST* are assigned to one output vertex group at a time. The edge control units fetch N input nodes simultaneously and then each unit forwards its fetched node and edge data to all gather units to convert the vertex data to analog signals, which are used to tune the MRs in the reduce units. The corresponding edge data is used by the gather units to define whether an input vertex is a neighbor of the assigned output vertex. Accordingly, the total delay of the aggregate block is dependent on the node with the largest number of neighbors.

The reduce unit is an optical unit configured as a coherent summation block. Each row in the reduce unit is assigned a feature from the vertex feature vectors, while each column is assigned a neighbor input vertex (see Figure 69(a)). The rows and columns have the sizes R_r and R_c, respectively. Thus, one reduce unit can aggregate R_r features of R_c neighbor vertices. Different VCSEL and waveguide colors in the figure represent different optical wavelengths. Each VCSEL source generates an optical signal which is split into R_c signals. These R_c signals are then passed through the MR bank to imprint the neighbor nodes' feature values onto the signals. As explained

in section 8.1.3, summation of the two values occurs when the different waveguides carrying signals with the same wavelength meet and they undergo interference. Since for a particular vertex the number of its neighbors may be greater than Rc, multiple mappings of different source vertices may be needed. Accordingly, the output of each row in the reduce unit is converted to an analog signal using a PD, and used to tune the last MR in each lane (Figure 69(a)) such that the sum that is output from that cycle will be added to the feature values in the next cycle.



(a) (b) (c)

Figure 69. (a) Reduce unit showing the needed changes in each feature lane to support the max aggregation operation using an optical comparator; (b) detailed view of transform unit; (c) detailed view of activate unit.

The organization of the reduce unit allows for the flexibility of implementing a wide range of reduce operations, which encompass most, if not all, GNN models. After the summation step as explained above, the output of the reduce unit is $(h_v^a = h_v + \sum_{u=0}^{n} h_u)$. The last MR after the coherent summation block is used to implement the mean operation where the output summation value would be adjusted by the MR such that it is multiplied by $\frac{1}{number\ of\ neighbors\ (n)}$, resulting in a reduce unit output of $(h_v^a = h_v + \frac{1}{n}\sum_{u=0}^{n} h_u)$. Further, for implementing other reduce operations, such as maximum, the reduce unit includes an optical comparator [163]. An example

of one lane in the reduce unit in that case would be as shown at the top of Figure 69(a). By activating blockers 1,3,4,6,7,9,10, and 12, the output of the reduce unit becomes the maximum value among all nodes. The optical output from all the rows in the reduce unit would then be combined into a single waveguide. This optical waveguide is connected directly to transform units in the combine block (discussed in the next subsection) to undergo the needed linear transformations. The same output values may need to be passed to the transform unit multiple times, depending on the size of the weight matrix. Accordingly, as the output from the reduce unit is passed directly to the transform units, it will be converted to the digital domain and buffered.

8.2.3.2 COMBINE BLOCK

The combine block accumulates the results from the aggregate block and performs linear transformation using the learned weight parameters. This generates a new learned, more expressive representation that captures the important structural information of the graph. Additionally, the linear transformation usually results in a reduced dimensionality for the vertex data representation, making the model more efficient as the dimensionality of some graphs can be very large (as shown later in Table 20). The transform unit's linear transformation is computed in the optical domain using MR bank arrays, as shown in Figure 69(b). Since linear transformation operations in GNNs are mainly MVMs, where the feature vector of the vertex being computed is multiplied by the weight matrix, as discussed in Section 8.1.3.2, such operations can be computed in the optical domain using MR bank arrays by passing the weight values to the transform unit as analog signals to tune each MR, while the feature vectors values are imprinted onto the optical signals in the waveguide from the reduce units. Unlike the MR array used in reduce units to perform summation operations, as presented in Subsection 8.2.3.1, multiplications are computed using non-coherent silicon photonics where multiple optical signals with different wavelengths are multiplexed using

WDM into a single waveguide and each MR is adjusted to operate at one of those wavelengths used. Accordingly, each feature value output from the reduce units and imprinted onto one optical signal's wavelength is multiplied by the weight values in the transform units. The number of rows in a reduce unit R_r is equal to the number of columns in a transform unit such that the number of optical signals in the waveguide is equal to the number of MRs in one row in the transform unit. The number of columns in a transform unit however is T_r.

As Batch Normalization (BN) is commonly used in many GNNs after the linear transformation, *GHOST* leverages broadband MR devices to perform BN in the optical domain where the BN parameter is used to tune the broadband MRs to adjust the optical signals as needed to reflect the BN operation. The efficiency of performing BN optically with this configuration was demonstrated in [163]. It is important to note that the BN unit can be bypassed if not needed. The output from each row is then accumulated using Balanced Photodetectors (BPD). BPDs are photodetectors that have two separate arms for the same waveguide, one for positive and the other for negative signal polarities. This allows them to accommodate both positive and negative parameter values by detecting the absolute difference between the two signals. The BPD sums the output signal from the positive arm and the output signal from the negative arm separately. Then, the BPD subtracts the output signal from the negative arm from the output signal from the positive arm to obtain the net difference signal.

If the size of rows in the weight matrix is smaller than or equal to the number of columns in the transform bank array and only one mapping for each weight matrix row is needed, the output from the transform unit after the BPDs will be passed directly to the activate units. Otherwise, the output will need to be converted to the digital domain and buffered till all needed values are computed and accumulated, and then passed to the activate units in the update block (discussed in

the next subsection). *GHOST*'s versatility to adapt to different model architectures and sizes, and not having to always convert the values to the digital domain, greatly reduces the latency and power costs associated with Analog-to-Digital Converters (ADCs) and buffering.

8.2.3.3 UPDATE BLOCK

The update block is composed of V update units to apply a non-linear activation function to the output from the transform units. The work in [212] demonstrated how semiconductor-optical-amplifiers (SOAs) can be exploited to implement multiple non-linear functions such as *RELU*, *sigmoid*, and *tanh*. For example, when the gain in an SOA is adjusted to a value close to 1, the behavior resembles the *RELU* operation. Accordingly, such non-linear operations are implemented optically, resulting in considerably improved performance. The analog signals output from the transform units are used to directly drive VCSELs, generating optical signals with the output value imprinted into its amplitude, which is then passed through the SOA-based non-linear unit. For the non-linear activation functions that are harder to implement optically (such as softmax), a digital activation unit, such as the one described in [213] can be integrated to accommodate these functions using Look-Up Tables (LUTs) and simple digital circuits, such as add and subtract. One update unit consists of T_r rows of activate units, in compliance with the number of rows used in each transform unit. Accordingly, the output from each row of the transform unit corresponds to one value in the vertex data vector, as shown in Figure 69(c).

8.2.4 ORCHESTRATION AND SCHEDULING OPTIMIZATIONS

GHOST supports four main optimizations for efficient orchestration and scheduling of GNNs, namely 1) graph buffering and partitioning, 2) execution pipelining and scheduling, 3) weight DACs sharing, and 4) workload balancing. While performing computations in the optical domain

already offers significant performance and energy benefits, efficient optimizations targeting improved memory bandwidth utilization and enhanced execution flow are imperative for designing a scalable and robust GNN accelerator.

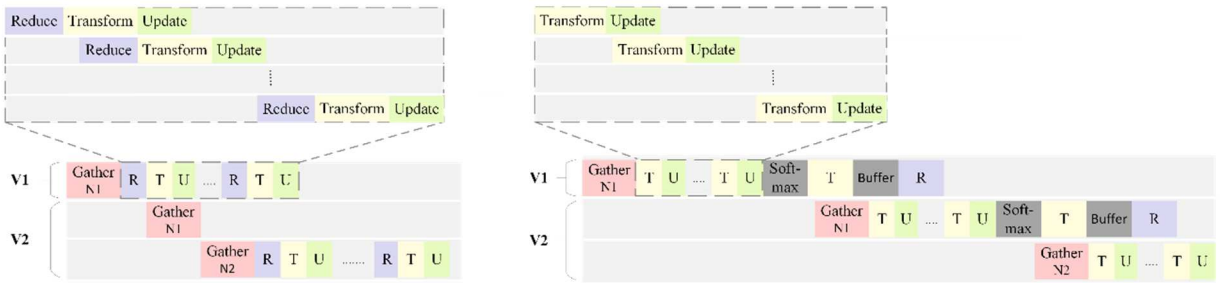
8.2.4.1 GRAPH BUFFERING AND PARTITIONING

Retrieving the entire graph from memory and processing it all at the same time entails tremendous memory bandwidth, resources, and computational costs. Hence, dividing the graph into several partitions and executing them separately is a widely used GNN optimization. But utilizing this approach alone can lead to increased latency as, in the worst case, while processing one vertex, it might necessitate loading another partition for each of its neighbor vertices. *GHOST*, on the other hand, uses a modified partitioning algorithm that builds on the one presented in [208]. As information regarding the graph edges is input, the adjacency matrix for the graph is generated. Columns of the adjacency matrix are identified as destination/output vertices while rows are the source/input vertices. Output and input vertices are then partitioned into groups of sizes V and N, respectively. Consequently, the edges data are also grouped into $V \times N$ chunks. For each partition where the group V_i is assigned to the vertex processing lanes, if for input nodes N_i, the corresponding group of edges contains one or more connected edges, N_i is prefetched and assigned to the edge control units, while all-zero blocks are skipped entirely. Generating the partition matrices and determining the memory access and fetching order is done once offline, as part of a graph preprocessing step. This significantly reduces sparsity in the graph and the need for complex techniques to deal with performing sparse operations. It also helps overcome the GNN's memory bottleneck and eases greatly the traffic and access frequency to and from the main memory.

8.2.4.2 EXECUTION PIPELINING AND SCHEDULING

GHOST can efficiently adapt the pipelining, computation scheduling, and execution ordering depending on the GNN model, as each model can require a different sequence of execution. For example, GCNs usually mandate having all nodes gathered first, reduced, transformed, and then updated. In contrast, GATs initially compute the attention coefficients, which involves gathering the nodes, performing linear transformations, and then applying a non-linear activation. In this case, the reduce step is performed at the end.

Given the buffering and partitioning optimization discussed in Section 8.2.4.1, *GHOST* performs pipelining at two levels of granularity. The first pipelining technique entails pipelining the operations within one output vertex group V_i. Initially as the output vertices are assigned to the execution lanes, all of their neighbor nodes are gathered. In case of models such as GCN, GraphSAGE, and GIN, which perform aggregation first and then the transform and update, *GHOST* pipelines the reduce, transform, and update executions. As soon as R_c (number of columns in a reduce unit) input vertices are gathered, the reduce units are initiated. Transform units are activated when R_r (number of rows in a reduce unit) feature values are ready and output from the reduce units, without the need to wait for all feature values to be computed. Similarly, the update units begin updating the vertex data directly after T_r (number of rows in a transform unit) values are linearly transformed. For the second pipelining technique between different vertex groups, the operations for a group V_{i+1}, are pipelined with those of group V_i. This scheduling approach ensures that the initial reduce for V_{i+1} is activated after the last reduce for group V_i. The pipelining model is illustrated in Figure 69(a).



(a) (b)

Figure 70. *GHOST* pipelining within one output vertex group V_I (top), and the pipelining between output vertex groups $V1$, $V2$, where $V1$ requires input vertices in $N1$ while $V2$ requires $N1$ and $N2$ for (a) GCN, GraphSAGE, GIN; and (b) GAT.

For other GNN models such as GATs, where transforming the vertex data occurs prior to the aggregation, the pipelining model is tailored to fit the model's specifications. Figure 70(b) shows an example of pipelining for a GAT model. For pipelining with one output vertex group, as the transformation of each vertex fetched by the edge control units is independent of the other vertices, gather operations are pipelined with the transform operations without the need to wait till all needed partitions are fetched and processed. Hence, the first group of input vertices is gathered, and the first linear transformation of multiplying the input with matrix W is handled by transform units, followed by attention vector multiplication. The output is then concatenated with transformed output vertices updated with a *leakyRELU* activation function by the activate units. When all the neighbor vertices $\in V$ are transformed, softmax is computed. Reduce is then computed at the end after the second transformation. Alternatively, when processing all output vertex groups V, the gather operations for the next output vertex group V_{i+1} are pipelined with the transform and update operations for vertex output group V_i.

8.2.4.3 WEIGHT DAC SHARING

A key characteristic of GNNs is that the same set of weights is applied across all vertices during processing. Accordingly, when all transform units in *GHOST* are operating with the same speed and processing vertices with the same feature sizes, the same weight values can be shared among all transform units. *GHOST* leverages this idea to implement a weight DAC sharing optimization. DAC devices are needed in *GHOST* to convert the digital values of weights that are read from the buffers in the ECU into analog signals. These analog signals are then used to tune optical MR devices to perform the linear transformations in each transfer unit. The DAC sharing optimization shares DACs across weights, thereby reducing the total number of DAC devices required, which is a significant factor in the power and latency budget of silicon-photonic accelerators, as normally one DAC device would be needed for each MR. With DAC sharing V (number of transform units) MRs would share the same DAC device and thus the total number of DAC devices is reduce by $\frac{\#MRs\ in\ combine\ block}{\#MRs\ in\ one\ transfer\ unit}$.

8.2.4.4 WORKLOAD BALANCING

Exploiting the buffer and partitioning optimization discussed in Section 8.2.4.1 implies having certain units/blocks in idle states during specific times. Due to some graphs' irregularity, the number of neighbors for each vertex within the same output vertex group V_i can vary considerably. As a result, when processing a GCN model for example, some gather units will be waiting in an idle state till the gather unit with the highest dimensionality vertex receives all its neighbor vertices. Our workload balancing optimization allows each lane to operate at its own rate without the need to wait while other lanes with higher dimensionality are still gathering their needed vertices. Accordingly, when such lanes complete processing their assigned vertices, the workload of the

other lanes will be split among the completed lanes. As a result, this can notably reduce the overall latency of the executing GNN model, especially when dealing with highly sparse and irregular graphs.

8.2.5 PROGRAMMING MODEL

GHOST's programming model is based on GReTA [204], an abstraction model tailored to processing a broad family of GNNs. GReTA utilizes four stateless User-Defined Functions (UDFs) to break down computation in each GNN layer, namely gather, reduce, transform, and activate. These UDFs are then performed in a series of three main execution phases: aggregate, combine, and update. Algorithm 1 explains the execution flow of the GReTA programming model as implemented by *GHOST*. *GHOST* takes as input the graph(s) as a set of edges and vertices represented by the notation *G(V,E)*, where edges are defined as two lists of vertices: destination/output (V), and source/input vertices (U). The aggregate phase iterates over all edges and invokes gather and reduce UDFs. The *Gather()* UDF collects each output vertex feature vector (h_v), its neighbor input vertices $(\forall h_u \in N(v))$ and the edges features $(h_{u,v})$ associated with the current output vertex being processed, and prepares a message value. The *Reduce()* UDF collects the messages that are output by gather, related to the same output vertex, and reduces them into a single value. The combine phase invokes the *Transform()* UDF where it takes in all the accumulated values for each vertex, employs the learned weight parameters, and performs linear transformation. Lastly, the update phase invokes the *Activate()* UDF and computes a non-linear activation function for each output vertex to generate the updated feature vectors for all vertices.

ALGORITHM 1: GHOST Programming Model

Input: Graph *G(V, E)*, source vertex feature data h_v, vertex feature data h_u, edges feature data $h_{u,v}$, weights *W*.
Output: Updated vertex feature data h_v'

1: // Edges Accumulate Phase
2: **for** each (u, v) in E:
3: h_{v_r} = Reduce(h_v, Gather(h_u, h_v, $h_{u,v}$))
4: // Vertices Accumulate Phase
5: **for** each v in V:
6: h_{v_t} = Transform(h_v, W)
7: // Update Vertices Phase
8: **for** each v in V:
9: h_v' = Activate(h_{v_t})

8.3 EXPERIMENTAL RESULTS

8.3.1 SIMULATION SETUP

To evaluate our proposed *GHOST* accelerator, we developed a comprehensive simulator in Python to estimate the power and latency of the accelerator. *GHOST* was simulated with attention to both software mapping and hardware mapping. For the software mapping, graph data is used to generate the partition matrix and retrieve needed information about the graph, such as the maximum number of neighbors in each partition. Then, we consider the layer-wise mapping and operation of each GNN model, and the architectural requirement for the mapping. For the hardware mapping, we modeled optoelectronic and electronic devices and circuits, and composed these into the blocks of the accelerator architecture. Compact models were used to analyze the losses associated with the device operation and also to determine device latency. The performance and energy estimates of all buffers used in *GHOST* were obtained using CACTI [214]. However, since CACTI only supports down to 20 nm technology, the obtained latency and energy values were scaled down to 7 nm using the set of scaling relations from [102]. The off-chip DRAM memory considered employs HBM2 with a size of 8GB, and it was simulated using DRAMsim3 [215].

The maximum bandwidth required to accommodate the largest graph dataset across all the different GNN models used in our experiments is 174.4 GB/s (the HBM2 memory system can support a maximum bandwidth of 256 GB/s [215]). The ECU is composed of 4 main buffers: input vertices buffer (128KB), output vertices buffer (128KB), edges buffer (256KB), and weights buffer (128KB). The memory bandwidth and access latencies of off-chip memory and on-chip buffers have all been accurately modeled using the methodology mentioned and are considered in all our simulations. For the softmax circuit needed for the GAT model using the update block, the design with a LUT and a maximum frequency of 294 MHz from [213] was utilized.

Table 19 displays the optoelectronic device and circuit parameters that were used during *GHOST*'s simulation-based analysis. Various factors were taken into account for assessing photonic signal losses, including waveguide propagation loss (1 dB/cm), splitter loss (0.13 dB [122]), combiner loss (0.9 dB [123]), MR through loss (0.02 dB [124]), MR modulation loss (0.72 dB [125]), EO tuning loss (6 dB/cm [40]), and TO tuning power (27.5 mW/FSR [39]). Also, as the number of wavelengths and waveguide length increase, so does the MR count, photonic loss, and required laser power consumption. Thus, the laser power required for each source used with multiple wavelengths in our architecture is modeled as follows:

$$P_{laser} - S_{detector} \geq P_{photo\_loss} + 10 \times \log_{10} N_{\lambda}, \quad (54)$$

where P_{laser} is the laser power (dBm), $S_{detector}$ is the PD sensitivity (dBm), N_{λ} is the number of laser sources/wavelengths, and P_{photo_loss} is the total optical loss (dB) due to the factors discussed.

Table 19 Parameters considered in *GHOST* analysis

Devices	Latency	Power
EO Tuning [40]	20 ns	4 μW/nm
TO Tuning [39]	4 μs	27.5 mW/FSR
VCSEL [47]	0.07 ns	1.3 mW
Photodetector [47]	5.8 ps	2.8 mW
SOA [47]	0.3 ns	2.2 mW
DAC (8-bit) [168]	0.29ns	3 mW
ADC (8-bit) [175]	0.82 ns	3.1 mW

Table 20 Graph datasets and associated parameters

Dataset	#Nodes (avg)	#Edges (avg)	#Features	#Labels	#Graphs
Cora	2,708	10,556	1,4322	7	1
PubMed	19,717	88,651	500	3	1
Citeseer	3,327	9,104	3,703	6	1
Amazon	7,650	238,162	745	8	1
Proteins	39	73	3	2	1113
Mutag	18	40	143	2	186
BZR	34	38	189	2	406
IMDB-binary	20	193	136	2	1000

A diverse set of GNN models, tasks, and graph datasets were used in our analysis. The following GNN models were considered: GCN [8], GraphSAGE [199], GIN [200], and GAT [9]. Each model processed four different graph datasets with the properties outlined in Table 20. The node-classification graph datasets (Cora, PubMed, Citeseer, Amazon) were processed with GCN, GraphSAGE, and GAT, while GIN was used for processing the graph-classification datasets (Proteins, Mutag, BZR, IMDB-binary). Further, GCN and GraphSAGE were implemented with two layers, while the MLP in GIN was implemented with eight layers. For the GAT model, two layers were implemented with the first one leveraging eight attention heads while the second used one attention head. The PyTorch Geometric library [201] was used to train and analyze each model's accuracy as shown in Table 21. Our analysis indicated that 8-bit model quantization

results in comparable algorithmic accuracy to models with full (32-bit) precision; thus, we targeted the acceleration of 8-bit precision GNN models.

Table 21 GNN model performances

Model	Dataset	Accuracy (32-bit)	Accuracy (8-bit)
GCN	Cora	88.70%	88.90%
	PubMed	87.40%	87.30%
	Citeseer	74.90%	74.40%
	Amazon	94.00%	93.70%
GraphSAGE	Cora	71.70%	70.80%
	PubMed	77.50%	76.90%
	Citeseer	63.30%	65.00%
	Amazon	77.10%	76.90%
GAT	Cora	78.30%	77.90%
	PubMed	76.70%	77.90%
	Citeseer	70.20%	69.10%
	Amazon	94.38%	94.64%
GIN	Proteins	74.00%	73.40%
	Mutag	94.74%	94.74%
	BZR	65.85%	65.85%
	IMDB-binary	77.00%	73.00%

In the following subsections, we present results from our analyses and experiments to determine optimal photonic device level configurations in *GHOST* (Subsection 8.3.2), the optimal values for *GHOST's* architectural parameters, which were discussed in Section 8.2.3, including V, N, R_r, R_c, and T_r, (Subection 4.3), sensitivity analysis to assess the impact of the orchestration and scheduling optimizations that were discussed in Section 8.2.4 (Subsection 8.3.4), and comparison with GNN accelerators proposed in prior work (Subsection 8.3.5).

8.3.2 DEVICE-LEVEL ANALYSIS

To ensure error-free photonic device operation in *GHOST*, we must reduce various noise sources in the analog domain, and ensure higher SNR than that dictated by (51) (see Section 8.2.2). We performed our device-level optimization analysis using optoelectronic simulation tools from

Ansys Lumerical [216]. For obtaining the operational characteristics of the active MRs, i.e., MRs with EO tuning as opposed to passive MRs without them, we used the FDTD, CHARGE, MODE solver, and INTERCONNECT tools [216]. FDTD was used to obtain the operational characteristics of the passive MR. The doping levels for the tuning junction's p and n doped regions was assumed to be $4e^{19}$ in our CHARGE simulations, to obtain the carrier distribution in the ring, under various voltage conditions (0V to 10V range). Using the carrier distribution to voltage relationship obtained from CHARGE, we performed MODE solver simulations to obtain the shift in effective refractive index (n_{eff}) over the voltage range. Finally, the data from these simulations was used in our INTERCONNECT simulations to obtain the operational characteristics of the MR under different biasing voltages. These analyses were performed over a range of design parameters for the MR and λ_{MR} values. From these simulations, we obtained our ideal MR design to have: ring and input waveguide width at 450 nm, radius of 10 µm, Gap of 300 nm, and a Q-factor of 3100. Using these parameters and (52) we can calculate the SNR required to be 21.3 dB.

The data from these tools was used for crosstalk and SNR analysis as mentioned previously in Section 8.2.2. Depending on the SNR requirements, the gap between the input and the ring waveguide and the width of the waveguides (input and ring) were explored and adjusted to achieve the required trade-off between Q-factor and the SNR at the output of the design as per (12). N_{levels} in our design is fixed from our software-level quantization to be 2^7, since we consider 8-bit quantization in our models (see Section 8.2.2 for discussion on this).

Using the device operational characteristics and the noise models from Section 8.2.2, we perform a sweep to determine the viable MR bank sizes for our coherent circuits ($P_{noise} = P_{\text{hom}\_noise}$) and non-coherent circuits ($P_{noise} = P_{het\_noise} + P_{\text{hom}\_noise}$), the results of which are shown in Figures 71(a) and 71(b).

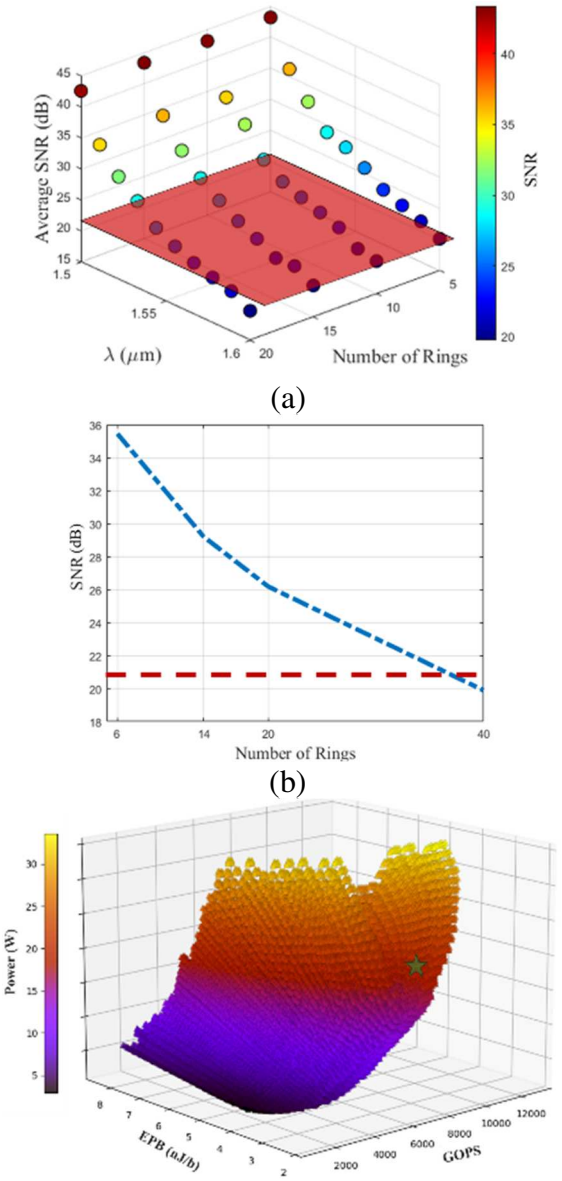


Figure 71. Design space exploration for (a) coherent and (b) non-coherent MR banks for *GHOST* architecture; (c) Architectural design-space exploration for *GHOST*, to find the optimal $[N, V, Rr, Rc, Tr]$ configuration with the best EPB/GOPS. The best configuration, [20,20,18,7,17] is shown with the green star.

For coherent MR banks we need to sweep for the wavelength to be used in the circuit, the number of MRs, and the SNR for the design. The cutoff SNR is shown as a red plane in the figure. From Figure 70(a), we can observe that it is possible to have up to 20 MRs in the coherent summation circuit, when the resonant wavelength is 1520 nm, while satisfying the SNR

requirements (red plane). Similarly, exploration for non-coherent circuits was also conducted and results are shown in Figure 71(b). The number of rings (MRs) along the x-axis is 2 times the number of wavelengths in the waveguide, as we need two MR banks to perform multiply and accumulate operations. We considered the first wavelength to be 1550 nm and used a channel spacing of 1 nm between wavelengths. The red line in Figure 71(b) indicates the cut-off SNR. From this analysis, we determined that the waveguide can host 36 MRs, or 18 wavelengths (1550 nm to 1568 nm) for non-coherent operation. These results were used to size and design the MR banks within the main architectural blocks of *GHOST*, to meet SNR goals while maximizing performance.

8.3.3 ARCHITECTURAL DESIGN SPACE EXPLORATION

The *GHOST* architecture relies on five main parameters, as outlined in Section 8.2: N, V, R_r, R_c and T_r. N refers to the number of edge control units or the size of each input vertices group in the partition matrix, while V refers to the number of execution lanes, which is also the size of each output node group in the partition matrix. R_c is the number of columns and R_r is the number of rows in the coherent sum MR array in the reduce units, which is also the number of columns for the MR bank array in the transform units. Lastly, T_r is the number of rows for the MR bank array in the transform units. To identify the optimal configuration for *GHOST*, which is determined by the combination of $[N, V, Rr, Rc, Tr]$ that offers the lowest EPB/GOPS (where EPB is energy-per-bit and GOPS is giga-operations-per-second), we conducted a detailed design space exploration as shown in Figure 71(c). Using the *GHOST* simulator described in Section 8.3.1, the EPB and GOPs values were obtained for each GNN model and each accompanying graph dataset for a wide set of possible values for $[N, V, Rr, Rc, Tr]$. The average EPB/GOPs values across all the GNN models

and datasets for each set of parameters were then obtained and the optimal configuration [20,20,18,7,17] was identified as the one with the lowest EPB/GOPS value.

8.3.4 ORCHESTRATION AND SCHEDULING OPTIMZATION ANALYSIS

We conducted a sensitivity analysis to assess the impact of each of the orchestration and scheduling optimizations described in Section 8.2.4. The normalized energy results are shown in Figure 72. The baseline configuration does not utilize any of the optimizations and each gather unit requests the needed neighbor vertices sequentially from the ECU. The Buffer and Partition (BP), Pipelining (PP), and DAC weight sharing (DAC_Sharing) optimizations and their viable combinations were explored in this analysis. For Workload Balancing (WB), implementing it in isolation was found to inefficient as the memory and buffer accesses are not synchronized and occur sequentially and on-demand. For instance, the processing lane that handles the vertex with the smallest dimensionality may not be the first to finish execution, as the order of memory access is also a critical factor. Moreover, employing WB necessitates having each lane possibly operating at different speeds, making it difficult to utilize the weight DAC sharing optimization. Therefore, implementing WB in isolation is impractical, and we only show results of considering WB alongside BP and PP to observe its benefits.

The results shown in Figure 72 are normalized to the energy consumption of the baseline model. As can be observed, employing BP, PP, and DAC sharing optimizations simultaneously results in the least energy values for all the GNN models as well as all the graph datasets used. On average, when using DAC sharing combined with BP and PP, the energy consumption is reduced by 4.94× compared to the baseline. On the other hand, using BP, PP, and WB reduces the overall energy consumption by 2.92×. Hence, while *GHOST* supports all four optimizations described in

Section 8.2.4, we leverage BP, PP, and DAC sharing for our *GHOST* configuration that is used in the subsequent sections for comparisons with other GNN accelerators.

Figure 72 also indicates that the different optimization techniques have different impacts across the various GNN models and graph datasets used. The optimizations have a greater impact with datasets having large number of vertices, and a very high degree of sparsity (e.g., PubMed). This demonstrates the scalability of *GHOST* to larger and more complex graphs and how the optimization techniques can alleviate the expensive memory and computational costs associated with GNN inference.

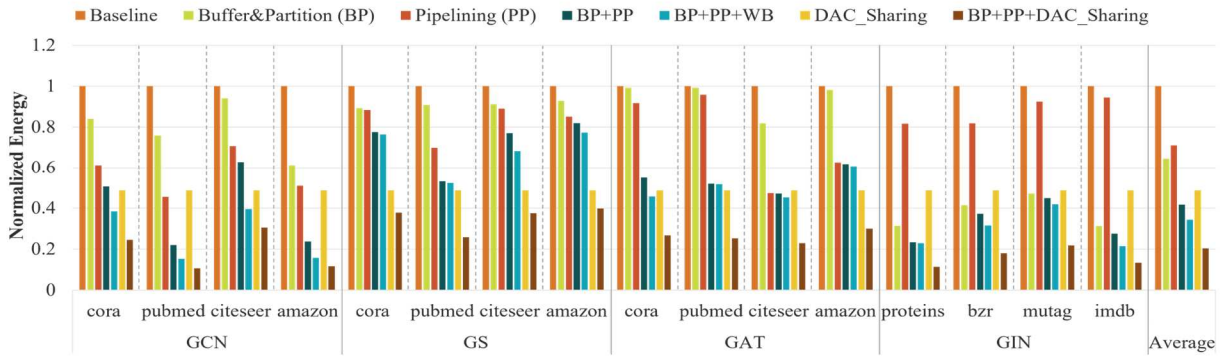


Figure 72. Impact of each orchestration and scheduling optimization on normalized energy consumption in *GHOST*.

Another key observation is the effect of BP and PP when processing different graphs. When processing larger graphs such as Cora, PubMed, Citeseer, and Amazon, PP results in lower energy consumption values than BP. Conversely, for datasets used with GIN, BP leads to lower energy values. While the graph datasets used with GIN are each composed of multiple graphs, each individual graph in the datasets is considerably smaller than the other graphs used with GCN, GraphSAGE and GAT. Accordingly, as pipelining is determined based on each individual graph, the impact of PP with small graphs diminishes. On the other hand, BP reduces sparsity and number

of memory accesses. Consequently, as we need to offload an entire graph from memory every time *GHOST* starts processing a new graph, employing BP results in notable energy reductions.

8.3.5 GHOST ARCHITECTURE COMPONENT-WISE PERFORMANCE ANALYSIS

To understand the performance of the major blocks in the GHOST architecture, we present a breakdown in Figure 73 in terms of latency when processing each GNN model and graph dataset for each of the main blocks: aggregate, combine, and update. It is evident from Figure 73 that the performance and contribution of each block to the overall inference latency depends on the GNN model being processed and the graph dataset used. In general, aggregate consumes more than half of the latency budget when processing GCN and GS models. This is mainly due to the models operating on graph datasets with large feature vectors and relatively high node degrees. Accordingly, the aggregate phase takes longer time as all neighbor node groups from the partitions matrix (from the graph buffering and partitioning technique in Section 8.2.4.1) need to be fetched and added. On the other hand, while GAT processes the same graph datasets, it follows a different execution ordering and pipelining as explained in Section 8.2.4.2. The GAT model used in our experiments is composed of eight attention heads, which are computed using the combine block. Moreover, the softmax function performed by the update block and included in the GAT computations is more time-consuming than the non-linear activation RELU used in GCN and GS models. The aggregation is performed only once at the end. Consequently, the high latencies observed with processing GATs are mainly attributed to the combine and update phases. Lastly, the graph datasets processed by the GIN model are multiple graphs used for graph classification tasks. However, each single graph is much smaller and possesses lower node degrees than the other graphs used with GCN, GS, and GAT. Accordingly, since a smaller number of neighbor nodes

need to be reduced, the aggregate phase is not as time consuming as with other models and the performance bottleneck can be attributed to the combine phase.

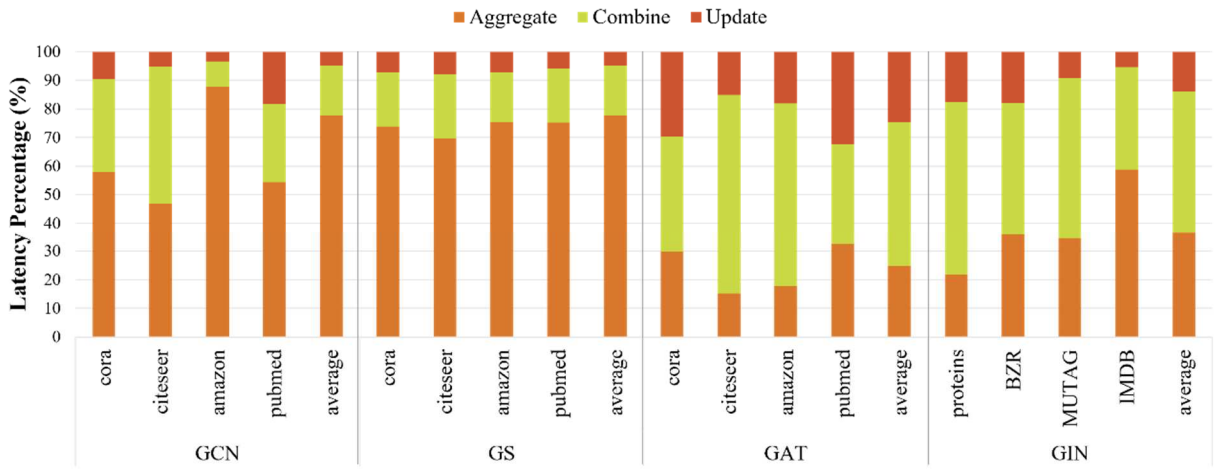


Figure 73. Breakdown analysis results showing the impact of each block in GHOST on the performance for each GNN model and graph dataset.

8.3.6 COMPARISON STUDIES

GHOST is compared against multiple computing platforms and state-of-the-art GNN hardware accelerators: GRIP [208], HyGCN [207], EnG [206], HW_ACC [205], ReGNN [209], ReGraphx [210], Google TPU v4, Intel Xeon CPU, and NVIDIA A100 GPU. We used power, latency, and energy values reported for the selected accelerators, and directly obtained results from executing models on the GPU, CPU, and TPU platforms to estimate the EPB and GOPS for each model and graph dataset.

We compared each hardware accelerator on the models supported by them, as outlined in their papers. For the models used in our analysis (Table 21), the GRIP and HyGCN accelerators support processing GCN, GraphSAGE, and GIN; EnG supports GCN and GraphSAGE; and HW_ACC supports GCN and GAT. The ReRAM-based hardware accelerators, ReGNN and ReGraphx, were

used for the GCN and GraphSAGE model comparisons. As mentioned, one of the key attributes of *GHOST* is its versatility in accommodating a diverse set of GNN models, enabling it to support the different models used in our analysis.

8.3.6.1 THROUGHPUT COMPARISON

Figure 74 shows the GOPS throughput comparison for *GHOST* with the computing platforms and GNN hardware accelerators considered. Our accelerator achieves on average 102.3×, 325.3×, 40.5×, 10.2×, 12.6×, 150.6×, 1699.0×, 1567.5×, 584.4× better GOPS when compared to GRIP, HyGCN, EnG, HW_ACC, ReGNN, ReGraphx, TPU, CPU, and GPU, respectively. While *GHOST* demonstrates promising improvements across all GNN models and datasets, the largest GOPS improvements are observed with the GIN graph dataset used for graph classification tasks. Across all datasets, processing GIN yielded on average 87.4× more GOPS when compared to the GNN hardware accelerators and 2168.9× when compared to GPU, CPU and TPU. This can be attributed to the small sizes of the graphs in each GIN dataset. These results are also consistent with those in Section 8.3.4, which illustrated that the partitioning optimization had the greatest impact on processing the graphs associated with the GIN model, leading to significant speedup. These findings highlight *GHOST*'s proficiency in handling diverse graph processing tasks. Also, there are significant GOPS improvements with *GHOST* for the GraphSAGE model, where it performed on average 100.9× better than other GNN accelerators and 1743.1× better than the GPU, CPU and TPU. This showcases how our accelerator is efficiently able to handle complex models, as it was able to overcome the complexity associated with supporting the sampling technique used in GraphSAGE.

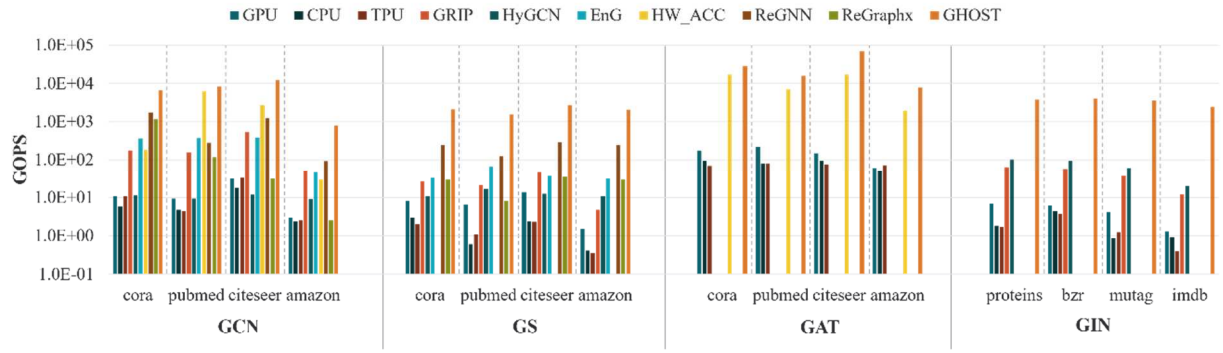


Figure 74. Throughput comparison between GPU, CPU, TPU, GNN hardware accelerators and *GHOST*.

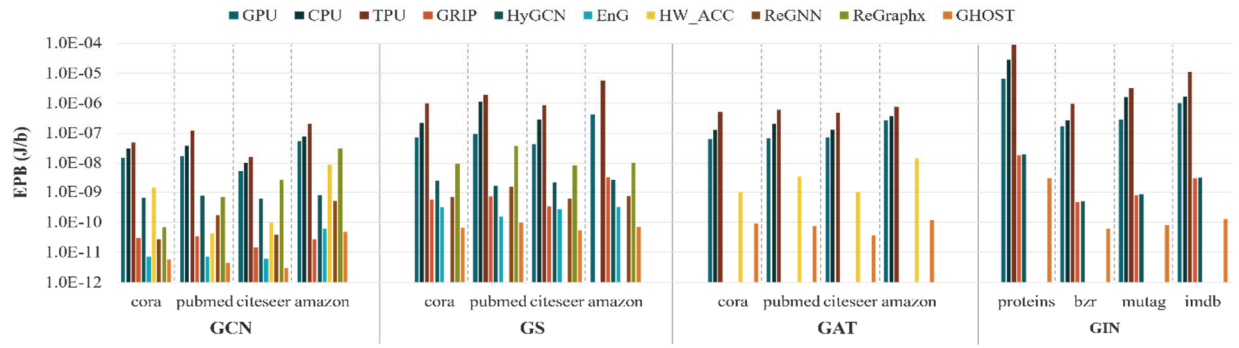


Figure 75. EPB comparison between GPU, CPU, TPU, GNN hardware accelerators and GHOST.

8.3.6.2 ENERGY EFFICIENCY COMPARISON

The EPB comparison results for *GHOST* with the computing platforms and GNN accelerators considered are shown in Figure 75. On average, *GHOST* attains 11.1×, 60.5×, 3.8×, 85.9, 15.7×, 313.7×, 24276.7, 6178.8×, 2585.3× lower EPB compared to GRIP, HyGCN, EnG, HW_ACC, ReGNN, ReGraphx, TPU, CPU and GPU, respectively. Our accelerator exhibits lower EPB values across all the GNN models and the graph datasets. In particular, GCN, which is the most widely used GNN model, achieved the lowest EPB values, with an average EPB reduction of 116.7× with *GHOST*, when compared to the GNN hardware accelerators and an average reduction of 7120.4×, in comparison to GPU, CPU and TPU. This is due to GCN's uniform operation pofile, which

involves processes each vertex independently based on its immediate neighbors only, without considering other vertices in the graph. Overall, the improved energy efficiency can be explained in terms of *GHOST*'s significant low latency operation in the optical domain, as well as its relatively low power consumption of 18W compared to the other hardware accelerators and compute platforms.

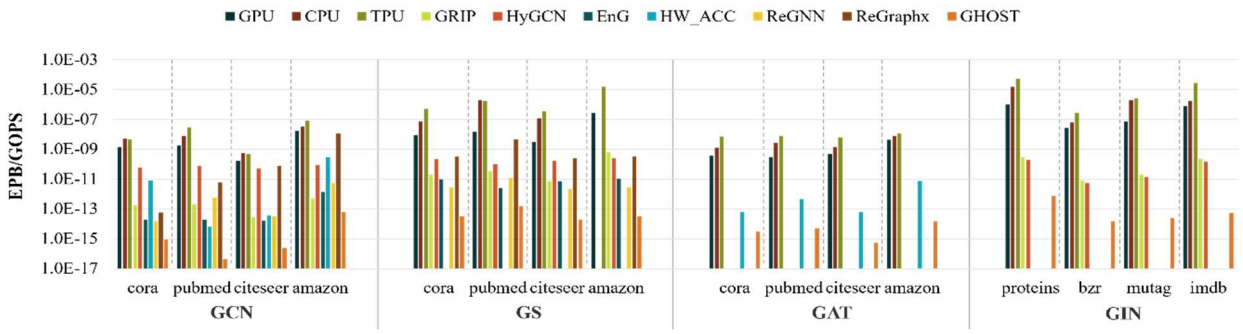


Figure 76. EPB/GOPS comparison between GPU, CPU, TPU, GNN hardware accelerators and *GHOST*.

8.3.6.3 EPB/GOPS COMPARISON

The primary motivation for the design of *GHOST* is to obtain both an energy-efficient and high-throughput GNN acceleration. Accordingly, to showcase *GHOST*'s performance and energy improvements together, we show the EPB/GOPS comparison for *GHOST* against the computing platforms and GNN accelerators from prior work in Figure 76. It can be seen that *GHOST* achieves exceptionally lower EPB/GOPS values across all models and datasets. On average, *GHOST* attains 2.7e3×, 190.3e3×, 197.2×, 1.9e3×, 1.9e3×, 90.1e3×, 121.4e6×, 22.8e6×, 4.8e6× lower EPB/GOPS, compared to GRIP, HyGCN, EnG, HW_ACC, ReGNN, ReGraphx, TPU, CPU, and GPU, respectively. These results demonstrate how the cross-layer optimizations employed in *GHOST* across the circuit-level, device-level, and architecture-level along with performing most of the GNN operations in the optical domain, help to mitigate the various GNN inference challenges.

8.4 CONCLUSION

In this chapter, we presented the first silicon photonic GNN accelerator, called *GHOST*. In comparison to nine computing platforms and state-of-the-art GNN accelerators, our proposed accelerator exhibited throughput improvements of at least 10.2× and energy-efficiency improvements of at least 3.8×. These results demonstrate the promise of *GHOST* in terms of energy-efficiency and high-throughput inference acceleration for GNNs. This chapter focused on the hardware architecture design with silicon photonics and employed various device-, circuit-, and architecture-level optimizations. When combined with software optimization techniques to reduce the high memory requirements in GNNs, we expect that even better throughput and energy efficiency can be achieved.

CHAPTER 9: TRANSFORMER ACCELERATION USING NON-COHERENT PHOTONICS

Transformer neural networks have gained significant popularity in the last few years, surpassing the performance of traditional Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) [217]. As the network architecture in transformer models relies on attention mechanisms and positional encodings instead of recurrence, it enables much higher parallelization than RNNs for sequence modeling and transduction problems. Since the introduction of the first transformer in 2017 [10], considerable progress has been made, with the emergence of powerful transformer-based pre-trained natural language processing (NLP) models, such as BERT [218] and Albert [219], and computer vision models, such as the Vision Transformer [220].

Despite the remarkable success of the transformer model, its size, number of parameters, and operations still require significant computational resources, hindering its progress and usage in resource-constrained systems. This consequently highlights the main issues with these models, which includes long inference times, large memory footprint, and low computation-to-memory ratio. Existing work on inference acceleration of conventional artificial neural networks (ANNs), such as CNNs and RNNs, mainly focuses on compute-intensive operations and optimizations at the layer-level granularity, which makes extending it to transformers—with its unique layer architecture and memory-intensive requirements—challenging.

Several transformer accelerators have been proposed in recent years to overcome these challenges with transformer execution [221], [222], [223], [224]. However, most of the work presented so far either focuses on accelerating a specific transformer architecture or is based on electronic components. Electronic accelerators are susceptible to the limits of the post Moore's

law era, where diminishing performance improvements are being observed with technology scaling. Such limitations also present major performance and energy bottlenecks for electronic dataflows [106]. On the other hand, silicon photonics has proven its proficiency as a solution beyond high-throughput communication in the telecom and datacom domains, and it is now being considered for chip-scale communication. Moreover, CMOS-compatible silicon photonic components can be used for computations, such as matrix-vector multiplications and logic gate implementations. Accordingly, the integration of silicon photonics is now actively being considered for deep learning accelerator platforms [37].

In this chapter, we introduce *TRON*, the first silicon-photonic-based transformer accelerator that can accelerate the execution of a broad family of transformer models. The novel contributions of this chapter are:

- The design of a novel transformer accelerator architecture with non-coherent silicon photonics, with the ability to accelerate any existing variant of transformer neural network models,
- Detailed crosstalk analyses, to improve signal-to-noise ratio (SNR) and tunability range for photonic microresonator (MR) banks,
- A framework for adaptive transformer accelerator design that can synthesize configurations for power-limited edge applications, and
- A comprehensive comparison with GPU, TPU, CPU, and state-of-the-art transformer accelerators.

The rest of the chapter is organized as follows. Section 9.1 presents a background on transformers, their acceleration, and ANN acceleration using silicon photonic devices. Section 9.2 provides an overview of the *TRON* architecture, including details of micro-architectural designs.

Section 9.3 discusses our experimental setup and results of comparisons with other accelerators, followed by the conclusions in Section 9.4.

9.1 BACKGROUND

9.1.1 TRANSFORMER NEURAL NETWORK MODELS

The attention mechanism has emerged as a prominent technique in sequence learning and NLP, where long-term memory is required. By utilizing the attention mechanism, transformers have outperformed RNNs (LSTMs, GRUs) across many NLP tasks. As shown in Figure 77, the original transformer model [10] designed for sequence learning has two main blocks: encoder and decoder. The encoder is responsible for mapping the input sequence into an abstract continuous representation. The decoder then processes that representation and gradually produces a single output while also being fed the previous outputs. Before being sent to the encoder, each input sequence is mapped onto a vector, and positional encoding is used to embed information regarding the position of each vector in relation to the original input sequence. The processed input is then passed through to the encoder/decoder block.

The encoder and decoder blocks often consist of *N* stacked layers. As can be seen from Figure 77, the two main sub-blocks in the transformer architecture are the multi-head attention (MHA) and feed forward (FF) layer, along with residual connections for each, followed by layer normalization. Self-attention is applied in MHA where it links each element (e.g., word) to other elements (e.g., words) in a sequence. Each MHA has *H* self-attention heads, and each attention head generates the query *(Q)*, key *(K)*, and value *(V)* vectors to compute the scaled dot-product attention. *Q*, *K,* and *V* vectors are generated by multiplying the MHA's input sequence *X* by the

query, key, and value weight matrices: W_Q, W_K, and W_V. The self-attention output is then computed through a scaled dot-product operation as follows:

$$Head(X) = attention(Q,K,V) = softmax\left(QK^T/\sqrt{d_K}\right)V, \quad (55)$$

where X is the input matrix and d_k is the dimension of Q and K. The output of the MHA is the concatenation of the self-attention heads' outputs, followed by a linear layer. The FF network is composed of two dense layers with a *RELU* activation in between.

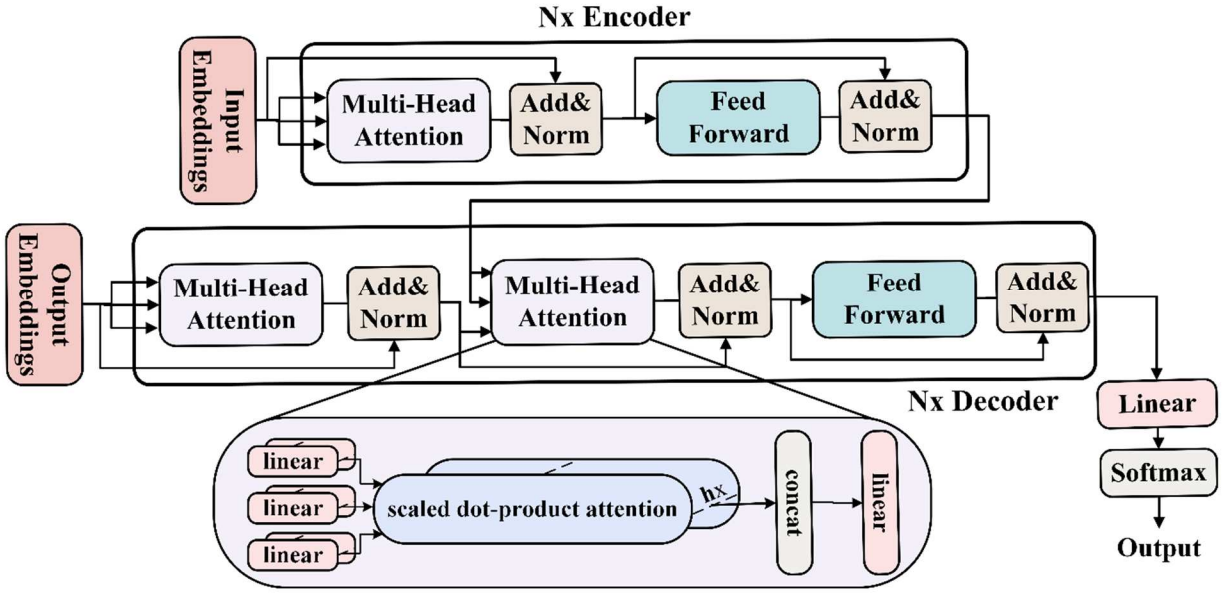


Figure 77. Transformer neural network model architecture overview.

More recent transformer-based pre-trained language models, such as BERT [218] and its variants [219], include the transformer encoder block only, as a cascaded set of N layers, followed by an FF layer, then GELU, and normalization layers. Similarly, the Vision Transformer (ViT) is composed of N encoder layers, followed by a multi-layer perceptron [220], where the ViT's inputs are sequence vectors representing an image.

9.1.2 TRANSFORMER ACCELERATION

Transformer accelerators to date have focused on accelerating either a specific subset of transformer models or specific transformer layers. For instance, [222] proposed an FPGA-based hardware accelerator, for accelerating MHA and FF layers. Their approach involves efficiently partitioning the weight matrices used in the MHA and FF layers to allow both layers to share hardware resources. In [224], another FPGA-based acceleration framework was proposed with a pruning technique for efficient model compression and an optimized method for storing the sparse matrices. An in-memory processing-based transformer accelerator called TransPIM was presented in [221], with a novel token-based dataflow for optimized data movements along with hardware modifications to the high bandwidth memory. The work in [223] focused on accelerating ViTs and proposed an automated framework, VAQF, that guides the quantization and FPGA resource mapping for a specific ViT. *Unlike prior efforts, TRON supports accelerating a broad family of transformer models for both NLP and computer vision tasks.*

9.1.3 SILICON PHOTONICS FOR ANN ACCELERATION

Due to the significant benefits offered by optical ANN accelerators in terms of performance and energy efficiency, they have garnered a lot of traction from academic and industry researchers [37]. Optical ANN accelerators are either coherent or non-coherent. In coherent architectures, which use a single wavelength, parameters are imprinted onto the optical signal's phase [113] to perform multiply and accumulate (MAC) operations. Alternatively, non-coherent architectures leverage multiple wavelengths and imprint parameters onto the optical signal's amplitude. Each wavelength can be used to perform operations in parallel. While coherent architectures exhibit high computation performance, they are vulnerable to phase encoding noise, phase error accumulation, and scalability limitations [164]. Consequently, more attention is being directed

towards non-coherent architectures. Current research in optical ANN accelerators has focused mainly on CNNs, MLPs, and RNNs [47]. To the best of our knowledge, *TRON* is the first optical accelerator for transformer ANN models.

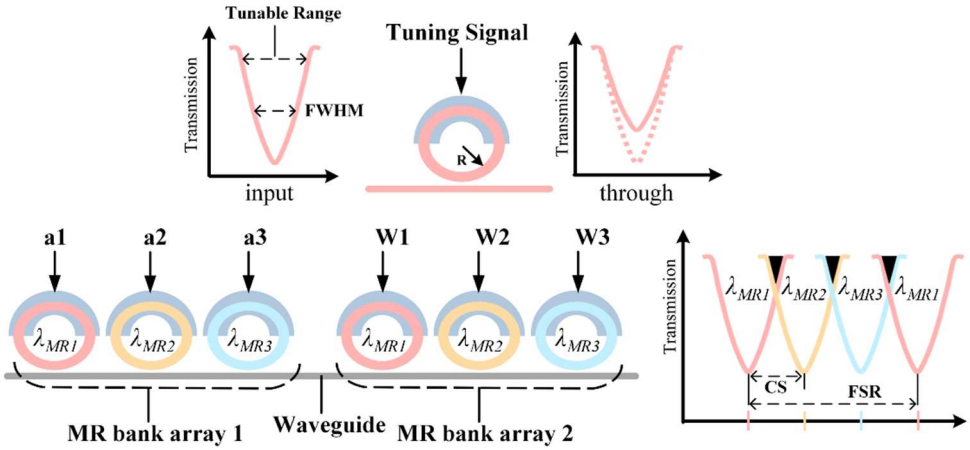


Figure 78. Top MR shows input and through ports' wavelengths after imprinting a parameter onto the signal. Bottom MR bank arrays perform multiplication by imprinting input activations (a_1-a_3), followed by weight vector values (W_1-W_3).

TRON is a non-coherent optical accelerator and uses MRs (see Figure 78) as the main opto-electronic device for carrying out key operations. Each MR can be designed and tuned to work at a specific wavelength, called MR resonant wavelength (λ_{MR}), defined as:

$$\lambda_{MR} = \frac{2\pi R}{m} n_{eff}, \tag{56}$$

where R is the MR radius, m is the order of the resonance, and n_{eff} is the effective index of the device. By carefully altering n_{eff} with a tuning circuit, we can modulate electronic data onto an optical signal passing by (in the vicinity of) an MR. The tuning circuit used is usually based on either thermo-optic (TO) [39] or carrier injection electro-optic (EO) tuning [40]. Both would result in a change in n_{eff}, and hence a resonant shift of $\Delta\lambda_{MR}$ in the MR. In non-coherent networks,

computations and, specifically multiplications, are done by tuning an MR's $\Delta\lambda_{MR}$, resulting in a predictable change in the optical signal's wavelength amplitude.

To increase throughput and mimic neurons in ANNs, non-coherent architectures use multiple wavelengths as part of wavelength-division multiplexing (WDM). This entails having multiple optical signals with different wavelengths in a single waveguide using an optical multiplexer [37]. The waveguide would then pass by a bank of MRs, each tuned to a certain wavelength in the waveguide, enabling us to perform several multiplications in parallel. Figure 78 illustrates an example of multiplying an input vector [a_1, a_2, a_3] by a weight vector [W_1, W_2, W_3]. Two MR bank arrays are used: the first imprints input activations onto the optical signals and the second performs the multiplication. The dot product output can thus be calculated by summing the three signals in the waveguide, which can be done by a photodetector (PD).

9.2 TRON HARDWARE ACCEELRATOR OVERVIEW

TRON is a non-coherent photonic accelerator architecture that can accelerate the inference of a broad family of transformer models. An overview of the architecture is shown in Figure 79. The photonic accelerator core is composed of MHA and FF units. Such composition allows reuse of resources for the encoder and decoder blocks. Interfacing with the main memory, buffering of the intermediate results, and mapping the matrices to the photonic architecture, are all handled by an integrated electronic-control unit (ECU). The following subsections describe the TRON architecture and the hardware optimizations we have considered to efficiently accelerate transformer ANN models.

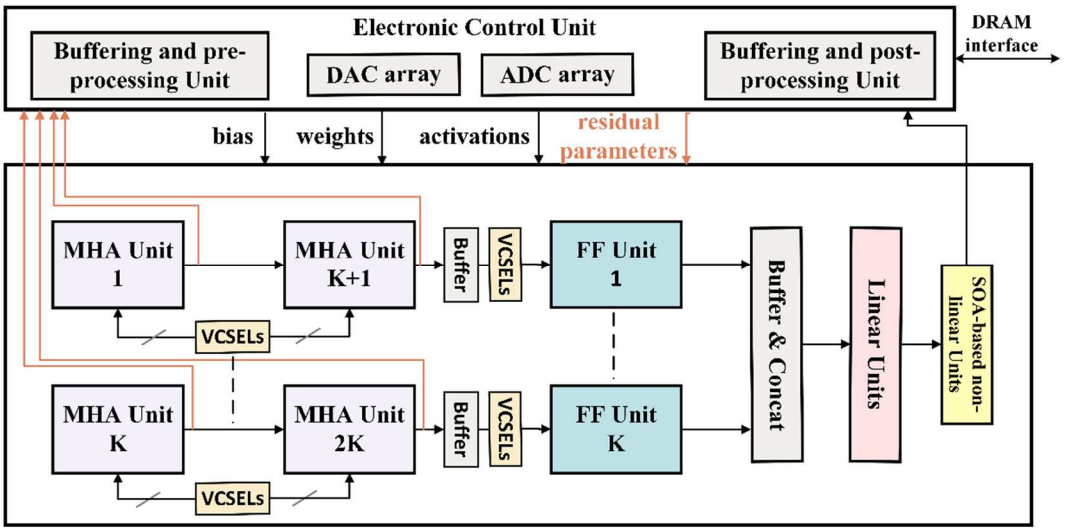


Figure 79. An overview of the proposed TRON accelerator architecture.

9.2.1 MR TUNING CIRCUIT DESIGN

As discussed earlier, MR devices in non-coherent architectures require a tuning mechanism, based on EO or TO. In *TRON*, we employ a hybrid tuning circuit where both TO and EO are used to induce $\Delta\lambda_{MR}$. This enables us to combine the advantages of both while overcoming their disadvantages. EO tuning is faster (≈ns range) and requires less power (≈4 μW/nm), but it cannot be used for large tuning ranges [40]. Conversely, TO tuning accommodates a larger tunability range but at the expense of higher latency (≈μs range) and power (≈27 mW/*FSR*) [39]. Accordingly, in our design, EO tuning is adopted for fast induction of small $\Delta\lambda_{MR}$ in MRs, while slower TO tuning is used only when larger $\Delta\lambda_{MR}$ is required. The effectiveness of this hybrid approach was previously demonstrated in [117]. To further reduce the power overhead of TO tuning, we adopt thermal eigen decomposition method (TED) from [54]. TED entails tuning all MRs within a bank array together, which reduces power consumption. Moreover, the approach uses microheaters to perform thermal tuning which reduces thermal crosstalk noise from heat dissipated from adjoining TO circuits.

9.2.2 MR BANK DESIGN-SPACE ANALYSIS

To ensure error-free MAC operations in the optical domain, it is necessary to manage various sources of noise, namely thermal and crosstalk noise, which can interfere with parameter imprinting and degrade the network performance and accuracy. Our TED-based tuning mechanism alleviates the thermal noise that can arise from TO tuning. But non-coherent architectures, like *TRON*, are inherently noise prone due to multiple wavelengths propagating in the same waveguide which creates inter-channel crosstalk. In inter-channel crosstalk, a portion of the optical signal from neighboring wavelengths can leak into one another, causing signal distortion (see Figure 78; bottom right). This phenomenon is further exacerbated with the presence of multiple MR banks in series, where multiple wavelengths can undesirably drop into an MR. With well-designed channel spacing (CS) and Q-factor in the MR, this can be managed by ensuring that the signal-to-noise ratio (SNR) is better than the detector sensitivity. The design of an MR should ensure adequate Q-factor to improve SNR. Additionally, the MR design should also possess sufficient tunable range, so that necessary parameters can be imprinted free of error. Here, tunable range refers to the wavelength band on the MR through port spectrum across which parameters can be imprinted. Mathematically, tunable range can be represented as 2× FWHM, where FWHM is full width half maximum (see Figure 78; top left). We optimize MR design for high FWHM and high SNR. For this optimization, we use the following models from [86]:

$$SNR\ (dB) = 10 \times \log_{10}\left(P_{signal}/P_{noise}\right), \tag{57}$$

$$P_{signal} = \Phi\left(\lambda_i, \lambda_j, Q\right) P_s\left(\lambda_i, \lambda_j\right), \tag{58}$$

$$P_{noise} = \sum_{i=1}^{n} \Phi\left(\lambda_i, \lambda_j, Q\right) P_S\left(\lambda_i, \lambda_j\right) (i \neq j)\ , \tag{59}$$

where Φ is the crosstalk coefficient corresponding to the inter-channel crosstalk between neighboring channels λ_i and λ_j, which is given by:

$$\Phi(\lambda_i, \lambda_j, Q) = \left(1 + \left(\frac{2Q(\lambda_i - \lambda_j)}{\lambda_j}\right)^2\right)^{-1}. \quad (60)$$

Here, $(\lambda_i - \lambda_j)$ represents the channel spacing CS, i.e., the spectral distance between two adjoining wavelengths. This is also an optimizable parameter within the confines of the free spectral range (FSR) we are considering. P_S in (58) and (59) is the signal power of λ_i that reaches the MR sensitive to λ_j, and can be defined as:

$$P_S = \psi(\lambda_i, \lambda_j) P_{in}(i), \quad (61)$$

where P_{in} is input power to the waveguide, calculated by considering the detector sensitivity and the signal power loss of λ_i before the MR with resonance wavelength λ_j within the bank, represented by ψ. When an optical signal in a waveguide passes by an MR, the crosstalk induced power suppression in its power can be modeled as a through loss, which is defined as γ times the signal power before it passes by the MR. This suppression factor γ and hence ψ can be calculated as follows:

$$\gamma(\lambda_i, \lambda_j, Q) = \left(1 + \left(\frac{2Q(\lambda_i - \lambda_j)}{\lambda_j}\right)^{-2}\right)^{-1}, \quad (62)$$

$$\psi(\lambda_i, \lambda_j) = \prod_{k=1}^{(k-1)<j} \gamma(\lambda_i, \lambda_k, Q). \quad (63)$$

For calculating FWHM, we use the following model:

$$FWHM = \frac{\lambda_{res}}{Q - factor}, \tag{64}$$

where λ_{res} is the resonant wavelength of the MR being considered.

Using these models, we can identify the optimal design space for our MR banks which can ensure high SNR and high tunable range (R_{tune}). To further narrow down our design space, we must consider that the lowest optical power level (P_{lpar}) should be higher than P_{noise}.

$$P_{lpar} > P_{noise}, \tag{65a}$$

$$\frac{P_{signal}}{P_{lpar}} < \frac{P_{signal}}{P_{noise}}, \tag{65b}$$

$$10\log_{10}\left(\frac{P_{signal}}{P_{lpar}}\right) < 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right), \tag{65c}$$

where P_{lpar} can be defined in terms of P_{signal} as follows:

$$P_{lpar} = \frac{P_{signal} \times R_{tune}}{N_{levels}}. \tag{66}$$

Replacing P_{lpar} in (65c) yields the following relation:

$$10\log_{10}\left(\frac{N_{levels}}{R_{tune}}\right) < SNR, \tag{67}$$

where N_{levels} is the number of amplitude levels we need to represent across the available R_{tune}: for an n-bit parameter (ANN weight or bias) representation, N_{levels} will be 2^n. If positive and negative values are represented separately, as in the case with *TRON*, then N_{levels} will be 2^{n-1}. The relationship in (67) can be rearranged to yield the following relationship between R_{tune} and

SNR:

$$R_{tune} > N_{levels} \times 10^{-\frac{SNR}{10}} \quad (68)$$

Utilizing these models, we can identify the ideal design space for our MR banks, as discussed later in Section 9.3.1.

9.2.3 MULTI-HEAD ATTENTION (MHA) UNIT DESIGN

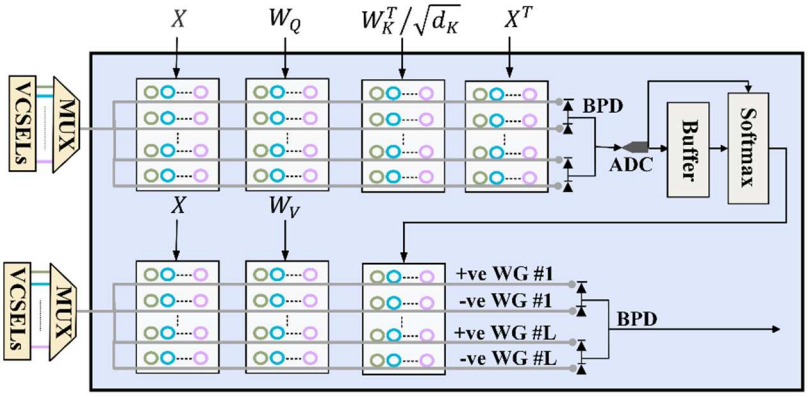
The major challenge with transformer inference acceleration is the time-consuming matrix multiplications (MatMuls). These operations can be decomposed into vector dot-product operations as outlined for optical CNN acceleration in [117]. Looking closely at the self-attention in each head (1), the computation of MatMul $(Q.K^T)$ cannot be performed until the generation and storage of K^T completes. This dependency would infer significant power and latency overhead as we would first need to generate K matrix $(K = XW_K)$ optically, convert the output to digital domain, buffer the values, generate K^T, then convert the matrix to the optical domain again to calculate the next MatMul $(Q.K^T)$. Alternatively, using MatMul decomposition, we can rewrite the operation as two cascaded MatMul steps as follows:

$$Q.K^T = Q.(X.W_K)^T = (Q.W_K^T).X^T \quad (69)$$

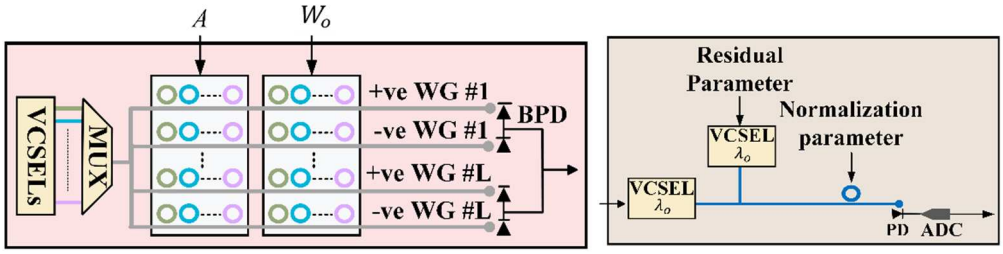
As shown by the top four MR bank arrays in Figure 80(a), no intermediate buffering is thus needed to compute $Q.K^T$. The first two MR bank arrays generate Q, then by having W_K^T and X^T previously stored and used to tune the MRs in the following two MR bank arrays, we can directly get the output of (69) optically without any intermediate buffering or expensive opto-electric conversions. To further reduce the latency and power overhead, we propose including the scaling factor in (1) within the weight matrix (W_K^T) storage in the ECU. As such, the individual MR tuning

values would be $W_K^T{}_i/\sqrt{d_k}$, instead of having an additional MR bank array to perform the scaling operation. As in most transformer models d_k (dimension of Q and K) is usually 64, a simple 3-bit left shift circuit should be able to efficiently handle the division.

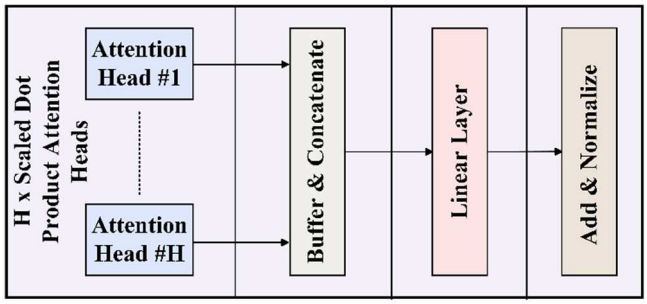
For the MatMul operations, most optical ANN accelerators (such as [47]) calculate them one-by-one, by having sperate MAC units with MR bank arrays to perform the multiplication operations. Then directly afterwards, they accumulate and add the partial sums. As there are more than two consecutive MatMul operations ((1) and (15)) involved in the attention computation, we avoid the accumulation of intermediate values and pass the individual multiplication results generated by the first MR bank array to the following MR bank arrays directly. The summation of all the multiplications and partial sums is then done at the end, before the softmax block, as shown in Figure 80(a). This approach avoids the latency and power costs from early summations, intermediate buffering, and associated opto-electric conversions. Moreover, as outlined in Section 9.3.1, we have ensured minimal crosstalk noise, that would normally be an issue due to such MR arrangement. Following the calculation of $(Q.K^T)$ by the upper MR bank arrays shown in Figure 80(a), all partial sums are accumulated using balanced photodetectors (BPDs). BPDs help accommodate both positive and negative parameter values by placing separate positive and negative arms for the same waveguide. The sum acquired from the negative arm is subtracted by the BPD from the sum from the positive arm. The results are then converted to the digital domain, to undergo softmax computation.



(a)



(b) (c)



(d)

Figure 80. (a) Attention head unit comprised of seven MR bank arrays for MatMul operations, each with dimension $K \times N$; (b) Linear layer comprised of an MR bank array with dimension $K \times N$; (c) Add and Normalization layers using coherent photonic summation and an MR for imprinting the normalization parameter; (d) MHA unit composed of H attention heads, buffer and concatenate block, linear layer, and an add and normalize block.

Another challenge in MHA is the softmax operation. It is performed in each attention head and restricts parallelism as all results from the previous MatMul need to be generated first. For its

implementation, we propose two optimization solutions. First, we avoid the computationally expensive division and numerical overflow by employing the log-sum-exp trick, used in a few previous works such as [222], as follows:

$$Softmax(\chi_i) = \frac{\exp(\chi_i - \chi_{max})}{\sum_{j=1}^{d_k} \exp(\chi_j - \chi_{\max})} , \quad (70)$$
$$= \exp\left(\chi_i - \chi_{max} - \ln\left(\sum_{j=1}^{d_k} \exp(\chi_j - \chi_{max})\right)\right),$$

where softmax can be divided into four main operations: finding χ_{max}, subtraction, natural logarithm (*ln)*, and exponential *(exp).* Finding χ_{max} and the subtraction can be computed using simple digital circuits. As shown in Figure 80(a), the analog-to-digital converter (ADC) output is buffered while also being fed to a comparator circuit, so that finding χ_{max} would be computed in parallel to the MatMuls. The natural logarithm (*ln)* and exponential *(exp)* computations can be calculated using look-up tables (LUTs). Prior work, such as [225], showcased the promising results and reliable implementations for memristor-based content addressable memories (CAMs) and LUTs. Accordingly, we propose utilizing similar memristor-based LUTs for the *ln* and *exp* computations. This also helps get the final softmax output as an analog value from the memristor cell in the LUT, which can be used to directly tune the MR bank array, without the need for further opto-electronic conversions. Furthermore, our scaled dot-product attention design enables high parallelism because the bottom vertical cavity surface emission laser (VCSEL) array (Figure 80(a)) can be synchronized to only be turned on when the softmax operation is done. Accordingly, the total delay for one pass in an attention head is as follows:

$$\delta_{Att_{Head}} = \delta_{A_{array}} + \delta_{W_{Q_{array}}} + \delta_{QK^T/\sqrt{d_K}_{array}} + \delta_{PD} + \delta_{ADC} + \delta_{buffer} + \delta_{softmax} + \delta_{MatMul} + \delta_{PD}. \quad (71)$$

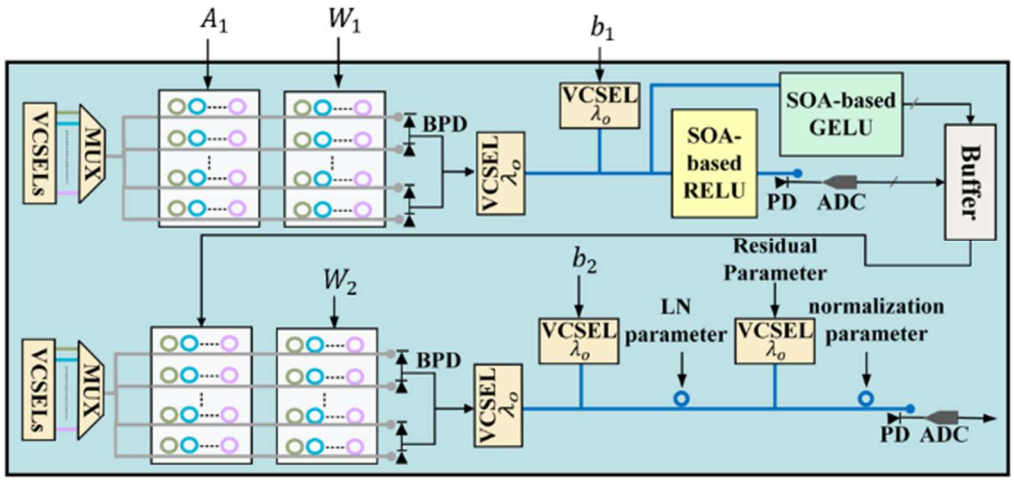
The linear layer in MHA is also implemented optically using two MR bank arrays (Figure 80(b)). For adding the MHA input to its current output (implementing the residual connection), coherent photonic summation is employed, as shown in Figure 80(c), where the output signal from the linear layer is used to directly drive a VCSEL with wavelength λ_o. Another VCSEL with the same wavelength, is driven by value(i) from the residual connection, and thus, when the two waveguides meet, they undergo interference, resulting in the summation of the two values. Coherent summation is ensured by using a laser phase locking mechanism [189], which guarantees that VCSEL output signals have the same phase for constructive interference to occur. Lastly, layer normalization (LN) is performed optically using a single MR, tuned by the LN parameter. The entire MHA architecture is shown in Figure 80(d).

9.2.4 FEED FORWARD (FF) UNIT DESIGN

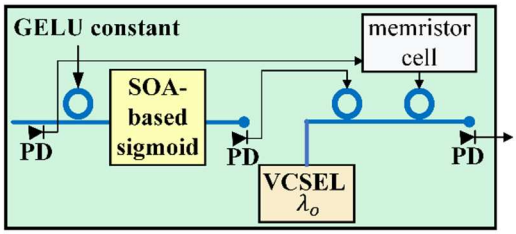
The FF Unit (Figure 81(a)) is composed of two fully connected (FC) layers, with a non-linear activation in between. Each FC layer is accelerated using two MR bank arrays, with dimensions $K \times N$: one to imprint the input activations and the second to compute the MatMul between the inputs and the weight matrices. The bias values are added using coherent photonic summation, discussed in the previous section. For the non-linear unit, we implemented an optical *RELU* unit, with semiconductor-optical-amplifiers (SOAs). When the gain in an SOA is adjusted to a value close to 1, the behavior becomes almost linear, resembling the *RELU* operation. The work in [212] demonstrated how SOAs can be exploited to implement other non-linear functions such as *Sigmoid* and *tanh*. This expands the scope of *TRON* and enables us to implement the *GELU* operation (used in ViT) instead of the *RELU*, optically. The *GELU* operation can be approximated as follows [226]:

$$GELU(x) = x\Phi(x) = 0.5x(1 + \tanh\left[\sqrt{2/\Pi}\,(x + 0.044715x^3)\right])$$

$$= x\sigma(1.702x). \qquad (72)$$



(a)



(b)

Figure 81. (a) FF block composed of four-MR bank arrays with dimensions $K \times N$, SOA-based RELU and GELU units, and bias and residual connection additions, done with coherent photonic summation; (b) GELU unit composed of three MRs, a semiconductor-optical-amplifiers (SOA), and a VCSEL.

As shown in Fig 5(b), the first multiplication $(1.702x)$ is implemented using a single MR, and the sigmoid function is computed using the SOA implementation, described above. The last multiplication of the input with the sigmoid output is calculated using two MRs. To store the input signal and use it to tune the second MR, a low-power, local storage mechanism is used where the analog input signal from the PD is stored in a memristor cell to directly tune the last MR.

The output from the non-linear unit is then buffered and used to tune the MRs in the first bank array of the second FC layer (Figure 81(b)), to be multiplied by the weight matrix (W_2). Following the second FC layer, the normalization layer is implemented using an MR, the residual connection is added through coherent photonic summation, and the final normalization layer is implemented with another MR.

9.2.5 *TRON* ARCHITECTURE

The architecture of *TRON* (Figure 79) is designed to accelerate various transformer ANN models. Given a transformer's sequential nature and unique structure, *TRON* is designed to support efficient sequential operations where needed, while also implementing parallel hardware to accelerate parallelizable operations. *TRON* is composed of two sets of MHA units and one set of FF units. Each set has a dimension of *L*. Such an arrangement enables both the encoder and decoder blocks to easily reuse most of the units. In case of the encoder block, the first VCSEL array will be used to drive the input to the second set of MHA units only. The MHA unit can be divided into two parts: before and after the softmax operation. As softmax (see (1)) cannot be computed till the first part is completed, both parts cannot be parallelized. However, the MatMul operations in the second part can be parallelized with the MatMul operations in the FF unit. For the decoder block, the first VCSEL array is used to drive the input to the first set of MHA units. Its output is used as the input to the second MHA unit whose output then drives the FF unit. Moreover, VCSEL-reuse by having a shared VCSEL array and reusing the same wavelengths across all rows of each MR bank array reduces the laser power consumption and inter-channel crosstalk. Accordingly, single VCSEL arrays are shared among rows in each MR bank array and used to imprint the input activations.

9.3. EXPERIMENTS AND RESULTS

We performed detailed simulation-based analyses to assess the efficiency of our proposed *TRON* architecture. Four transformer models were considered in our analyses: Transformer-base [10], BERT-base [218], Albert-base [219], and ViT-base [220]. The model parameters are shown in Table 22, where d_{model} and d_{ff} are the dimensionality of input/output and FF layers. We developed a simulator in Python to estimate the area, performance, and energy costs associated with running each model. The area, performance, and energy estimates for all electronic buffers used in *TRON* were estimated using the CACTI tool [101] at 28nm; while the electronic circuit in softmax, was synthesized using Xilinx Vivado at 28 nm and the resulting power/delay estimates were used in our analyses. Tensorflow 2.9 was used to train and analyze each model's accuracy.

Table 22 Transformer model configurations

Model	Parameters	Layers	Heads	d_{model}	d_{ff}
Transformer-base	52M	2	8	512	2048
BERT-base	108M	12	12	768	3072
Albert-base	12M	12	12	768	3072
ViT-base	86M	12	12	768	3072

Table 23 Transformer model performance

Model	Dataset(s)	Accuracy (32-bit)	Accuracy (8-bit)
Transformer-base	Ted_hrlr_translate	66.73%	70.4%
BERT-base	Sentiment-Analysis-of-IMDB-Movie-Reviews	85.8%	85.8%
Albert-base	Sentiment-Analysis-of-IMDB-Movie-Reviews	88.3%	88.7%
ViT-base	ImageNet/Cifar-10	97.7%	98.0%

The achieved accuracies and the datasets associated with each model are as shown in Table 23. As shown, Transformer, BERT, and Albert models were used for NLP tasks (language translation and sentiment analysis). ViT was evaluated using an image classification task, with

pre-training on ImageNet and fine-tuning on Cifar-10. Our analysis concluded that 8-bit model quantization results in comparable algorithmic accuracy to models with full (32-bit) precision; thus, we targeted the acceleration of 8-bit precision transformer models.

The optoelectronic parameters examined for *TRON's* simulation-based analysis are shown in Table 24. We considered various factors that contribute to photonic signal losses such as: waveguide propagation loss (1 dB/cm), splitter loss (0.13 dB [122]), combiner loss (0.9 dB [123]), MR through loss (0.02 dB [124]), MR modulation loss (0.72 dB [125]), EO tuning loss (6 dB/cm [40]), and TO tuning loss (27.5 mW/*FSR* [39]). Increasing the number of wavelengths and the waveguide length will in turn increase the MR count, photonic loss, and the required laser power consumption. Accordingly, we modeled the required laser power used in our architecture for each source as:

$$P_{laser} - S_{detector} \geq P_{photo\_loss} + 10 \times \log_{10} N_{\lambda}, \quad (73)$$

where P_{laser} is the laser power in dBm, $S_{detector}$ is the PD sensitivity in dBm, N_{λ} is the number of laser sources/wavelengths, and P_{photo_loss} is the total optical loss encountered by the signal, due to the factors discussed. In the next subsection, we describe our analyses to determine the optimal values for *TRON's* architectural parameters *H, L, K,* and *N*, which were discussed in section 9.2.

Table 24 Parameters considered for *tron* analysis

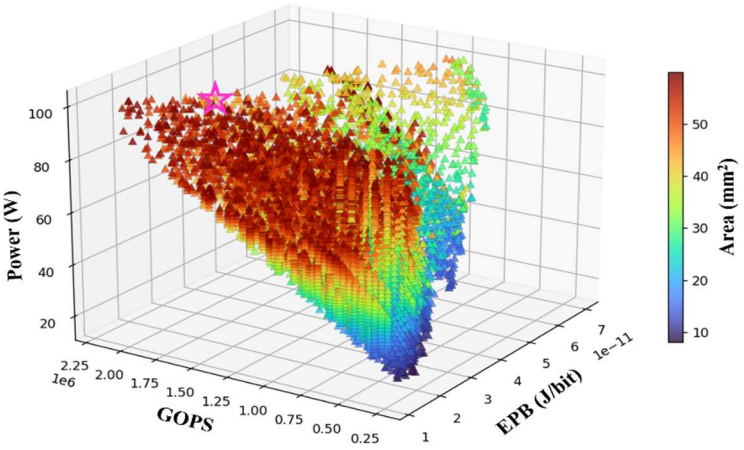
Devices	Latency	Power
EO Tuning [40]	20 ns	4 μW/ nm
TO Tuning [39]	4 μs	27.5 mW/*FSR*
VCSEL [47]	0.07 ns	1.3 mW
Photodetector [47]	5.8 ps	2.8 mW
SOA [47]	0.3 ns	2.2 mW
DAC (8 bit) [168]	0.29 ns	3 mW
ADC (8 bit) [175]	0.82 ns	3.1 mW
Memristor cell [47]	0.1 ns	0.07 μW

9.3.1 *TRON* ARCHITECTURE DESIGN OPTIMIZATION

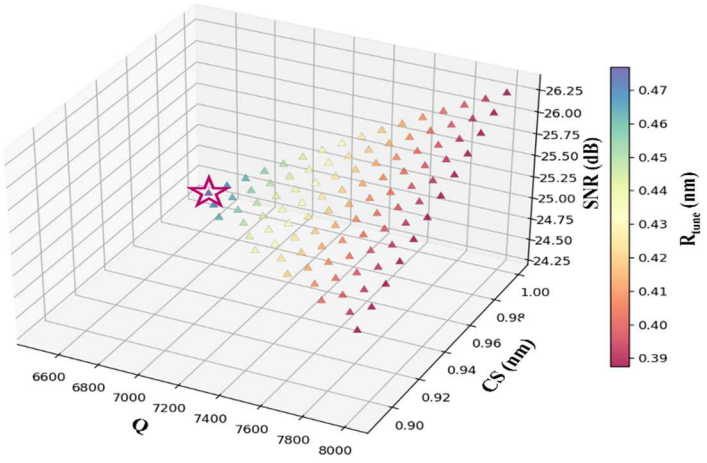
The *TRON* architecture design is dependent on four key parameters, as discussed in Section 9.2: *H* (the number of heads in the MHA unit*), L* (the number of layers), *K* (the number of rows), and *N* (the number of columns in each MR bank array).

We performed an exploration to determine the optimal [*H,L,K,N*] configuration for *TRON,* defined as the configuration with lowest EPB/GOPS, where EPB is a measure of energy-efficiency (energy-per-bit) and GOPS is a measure of throughput performance (giga-operations-per-second). We also set a maximum power limit of 100W for the configuration. The result of this exploration is presented in the scatterplot in Figure 82(a). The optimal configuration [4,2,51,17], is highlighted with the pink star. This configuration is used in the comparative analysis with GPU, CPU, TPU, and other accelerators in the following subsections.

For the MR-bank design, the models described in Section 9.2.2 were used to perform another exploration study. Using the SNR model (3), with the R_{tune} constraint (14), we explored the MR bank design space to find the parameters [R_{tune},*Q,SNR,CS*], with the aim of maximizing tuning range R_{tune}. We considered N_{level} of 2^{8-1}, an FSR of 20 nm, Q-factor ranging from 2000 to 8000, and channel spacing ranging from 0.1 to 1 nm. The result of the exploration is as shown in Figure 82(b), where we have selected the data point with the best R_{tune}: [0.45, 6500, 24.3, 1].



(a)



(b)

Figure 82. (a) Architectural optimization for *TRON*, to find the optimal [*H, L, K, N*] configuration with the best energy-efficiency and throughput. The best configuration, [4,2,51,17], has the lowest EPB/GOPS value and a maximum power of 100W is shown with the pink star; (b) MR bank optimization for *TRON*, aiming to identify optimal [R_{tune},*Q,SNR,CS*] design point. The best point [0.45,6500,24.3,1] with highest R_{tune} value, is shown with the pink star.

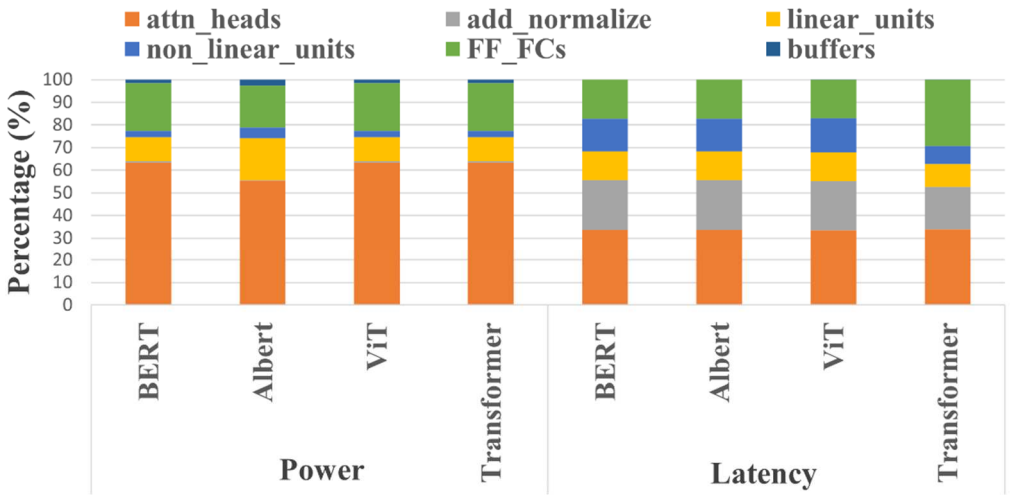


Figure 83. Power and latency breakdown across *TRON* components

9.3.2 *TRON* ARCHITECTURE COMPONENT-WISE ANALYSIS

To understand the performance of the major components within the *TRON* architecture, we present a breakdown in terms of power and latency for these components in Figure 83. We focused on the contributions for the main operations across transformer models: MatMuls in attention heads, addition and normalization layers, linear layers, non-linear layers, FC layers in the FF blocks, and the RAM-based buffers. For the power, it is evident that MatMul operations in the attention heads contribute to more than half of the architecture's power overhead. This is because of the large dimensions of the matrices being multiplied in the MHA blocks, in each attention head. This requires many digital-to-analog converters (DACs), whose power consumption is considerable. Moreover, the sequential dependency in the attention head also contributes notably to the latency overhead. As Albert shares all attention and FF parameters across layers [219], this leads to a minimization of the number of active DACs, reducing the overall power consumption for the Albert model.

9.3.3 COMPARISON TO STATE-OF-THE-ART ACCELERATORS

We compared *TRON* execution on multiple processors and state-of-the-art transformer accelerators: Tesla V100-SXM2 GPU, TPU v2, Intel Xeon CPU, TransPIM [221], FPGA transformer accelerator in [222] (FPGA_Acc1), VAQF [223], and FPGA transformer accelerator in [224] (FPGA_Acc2). VAQF focuses on vision transformers and FPGA-Acc2 on traditional encoder-decoder transformer architectures and transformer-based language models; results for these two platforms are thus restricted to the models they are targeted for. We used power, latency, and energy values reported for the selected accelerators, and results from executing models on the GPU/CPU/TPU platforms to estimate the EPB and GOPS for each model. The *TRON* architectural configuration used in the comparisons has a maximum power of 100W, and is the one described in Section 9.3.1.

Figure 84 shows the GOPS comparison between *TRON* and the other architectures considered. Our architecture achieves on average 262×, 1631×, 1930×, 14×, and 55× better GOPS than GPU, TPU, CPU, TransPIM, and FPGA_Acc1, respectively. When comparing transformer model-specific accelerators, *TRON* has on average 352× higher GOPS than FPGA_Acc2 for transformer, BERT, and Albert models, and 846× higher GOPS than VAQF for ViT. The higher throughput over all compute platforms can be explained in terms of *TRON*'s high-speed execution in the optical domain and the minimal computations in the digital/electric domain.

Figure 85 presents the energy-per-bit (EPB) comparison. On average, *TRON* attains 4231×, 12397×, 10971×, 14×, and 8× lower EPB than GPU, TPU, CPU, TransPIM, and FPGA_Acc1. For model-specific accelerators, we achieve on average 802× lower EPB than FPGA_Acc2 for transformer, BERT, and Albert models, and 32× lower EPB than VAQF for ViT. These EPB

improvements can be attributed to *TRON*'s significant low latency operation, and the relatively lower power compared to some of the compute platforms considered.

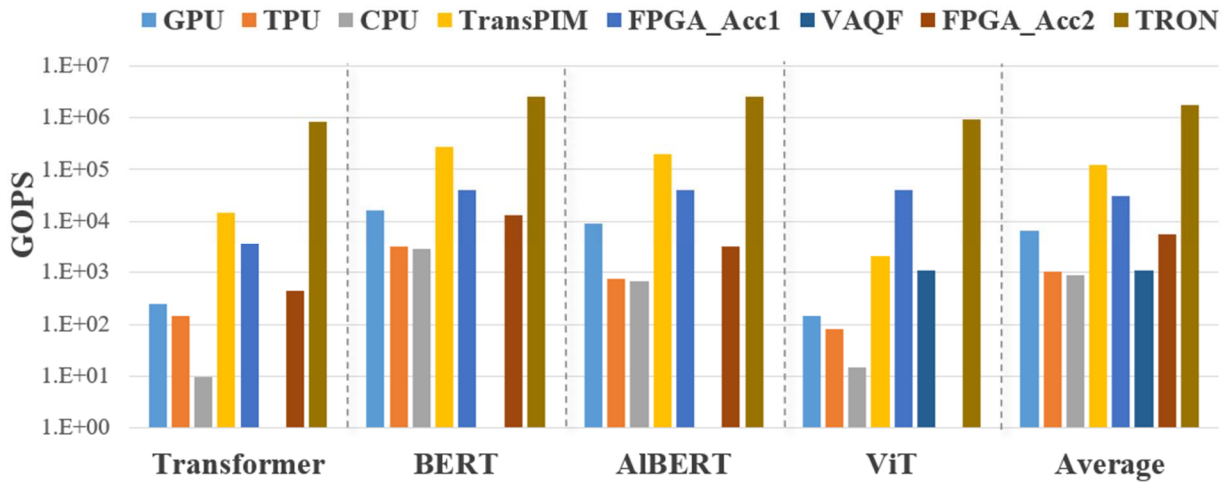


Figure 84. Throughput comparison between transformer accelerators and platforms.

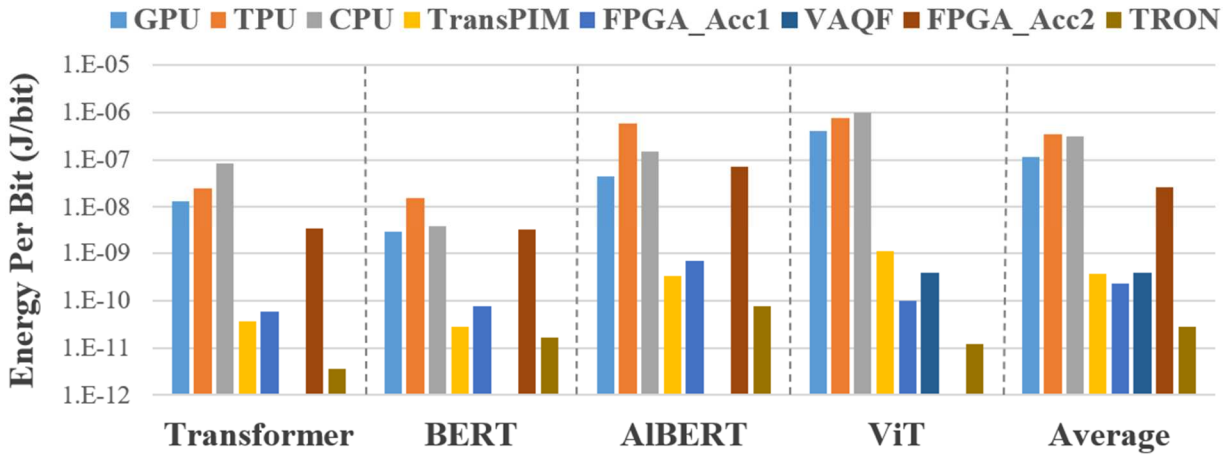


Figure 85. EPB comparison between transformer accelerators and platforms.

9.3.4 *TRON* FOR EDGE ENVIRONMENTS

Edge computing environments have stringent power constraints for accelerators. We performed a design space exploration, similar to that described in Section 9.3.1, to find an edge-

friendly *TRON* configuration with a power limit of 10W (instead of 100W that we considered earlier). We identified the optimal edge configuration values for [*H,L,K,N*] as [4,1,12,12]. Figure 86 illustrates a comparison for the average power, GOPS, and EPB values across models, among *TRON_edge, TRON,* and the platforms previously discussed. The values shown are normalized to those obtained for the CPU. Our *TRON_edge* accelerator consumes on average, considerably lower power (~10W). While the GOPS values slightly decrease, the edge configuration's throughput still outperforms all compute and accelerator platforms by at least 16%. The EPB value for *TRON_edge* is higher than for *TRON* but it still is on average 4× to 6292× lower than all other platforms. In this manner, TRON can be customized to provide the best performance and energy-efficiency for any given target power consumption constraint.

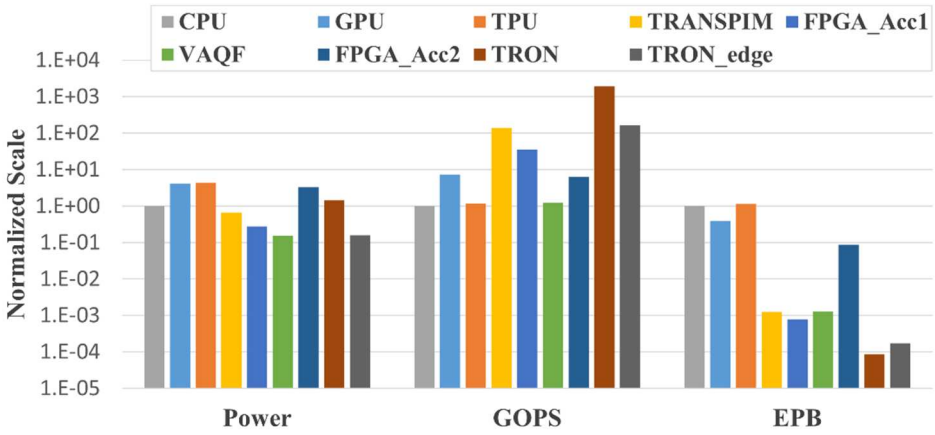


Figure 86. *TRON*_edge's power, throughput, and energy comparison.

9.4 CONCLUSIONS

In this chapter, we presented the first non-coherent silicon photonic hardware transformer accelerator, called *TRON*. Our proposed accelerator architecture exhibited throughput improvements of at least 14× and energy-efficiency improvements of at least 8× when compared to eight different processing platforms and state-of-the-art transformer accelerators. We also

showed how to adapt *TRON* to different scenarios, such as edge computing environments. These results demonstrate the promise of *TRON* in terms of energy-efficiency and high-throughput inference acceleration for transformer neural networks. This chapter focused on the hardware architecture design with silicon photonics. When combined with software optimization techniques that aim to reduce a transformer's large memory footprint, such as model compression, parameter sharing, and attention window-size reduction, even better throughput and energy efficiency may be achieved.

CHAPTER 10: LEVERAGING 2.5 D PLATFORM FOR IMPROVING NONCOHERENT ACCELERATOR PERFORMANCE

Deep neural networks (DNNs) are ubiquitously employed today in a wide range of applications, including, but not limited to, autonomous vehicles, medical diagnosis, network security, recommendation systems, and navigation solutions [227], [228], [229], [230], [231]. To cater to the objectives of these applications, DNNs have become quite varied, with the DNN family including convolution neural networks (CNNs), recurrent neural networks (RNNs), graph neural networks (GNNs), transformers, etc. A commonality among these DNN variants is the increasing model complexity and upward trend of parameter count. To meet the processing and latency demands of these applications, the hardware architecture must also scale in terms of processing capabilities, on-chip memory capacity, and on-chip communication capabilities. Graphic processing units (GPUs) are usually tasked with accelerating DNN execution today, but several limitations of the general-purpose nature of GPU architectures have become apparent in recent years. These limitations include high power consumption, increasing area overhead, reducing performance per watt, and memory bandwidth limitations [232].

The limitations of GPUs highlight the need for more efficient domain-specific accelerator architectures. However, the growing processing requirements of modern DNNs does not favor monolithic (single chip) architectures [233]. Monolithic implementations of domain specific accelerators can face scalability, power density, fabrication yield, and latency issues [234]. To tackle these problems and to effectively accelerate modern DNNs in a scalable manner, 2.5D architectures are actively being considered today [235].

Scaling 2.5D architectures comes with the challenge of increasing inter-chiplet distances. In this scenario, it can be shown that inter-chiplet metallic interconnects pose a major challenge to system performance due to excess latency and energy consumption [236]. Because of these limitations, electrical interconnect alternatives must be considered to ensure that 2.5D accelerators can deliver on the demands for low latency and energy-efficient DNN acceleration.

Optical interconnects based on silicon photonics can overcome the limitations posed by metallic interconnects through advantages such as high bandwidth communication [237], single-hop data propagation [238], and high energy efficiency [239]. Silicon photonic interconnects also allow for ease of broadcast [239], [99], which is a desirable feature for DNN acceleration [240], [241], [242]. Further energy and latency benefits can be extracted from photonics by utilizing photonics for computation as well. Many prior efforts have shown that photonic processing for DNN inference acceleration provides significant benefits in terms of latency and energy efficiency [243], [162], [244], [43], [46], [111], [138], [44], [47]. Thus, it stands to reason that utilizing photonics for both communication and computation may amplify the aforementioned benefits. In this work, we explore the benefits provided by silicon photonic chiplets and networks for DNN acceleration in 2.5D chiplet platforms.

The organization of the remainder of this chapter is as follows. An overview of silicon photonics is provided in Section 10.1. Section 10.2 discusses silicon-photonic-based DNN accelerators. In Section 10.3, state-of-the-art silicon photonic interposer networks are presented. Section 10.4 describes our 2.5D chiplet-based DNN accelerator. Section 10.5 presents various experimental results. Finally, Section 10.6 provides conclusions and open challenges in this emerging area.

10.1 SILICON PHOTONICS: BACKGROUND

Silicon photonics emerged as a CMOS-compatible technology to enable chip-scale optical communication. To achieve this, silicon-on-insulator (SOI) waveguides are employed, which use silicon (Si) for the core material and silicon-dioxide (SiO_2) for cladding and substrate material. Moreover, by using wavelength-division multiplexing (WDM), optical signals on different wavelengths can simultaneously traverse the same waveguide. Silicon photonics promises high energy efficiency, bandwidth density, and low latency, as the overall scale in terms of communication distances increases [32]. Due to the benefits that silicon photonics offers, there has been a growing interest in using silicon photonics for computation, including realizing digital logic using photonics [245], [246]. To realize any photonic-based computation or network system, there is a need for many fundamental components, as discussed next.

Silicon photonic waveguides are analogous to metallic wires in electrical chips and enable optical signal transmission and routing in chips. Photonic waveguides operate on the principle of total internal reflection (TIR) to contain and guide optical signals [247]. To ensure TIR, these waveguides require high refractive index contrast between their core and cladding materials (e.g., an SOI platform).

Lasers are a key requirement for any photonic system as they act as light sources for communication and computation. The laser sources employed can be on-chip or off-chip [248]. Off-chip lasers offer better light emission efficiency, but they face high optical power losses during coupling to on-chip waveguides. On-chip lasers provide better integration density and lower optical loss, as there is no need to couple light, but they suffer from low light emission efficiencies. In most photonics-based systems, the laser that is commonly utilized is a vertical cavity surface

emission laser (VCSEL) or a microring laser. If off-chip lasers are used, then couplers are necessary devices to couple the optical signals from off-chip sources to on-chip waveguides. Coupling solutions employed can be surface grating couplers or edge couplers [249].

Microring resonators (MRs) are photonic devices that are widely used to design modulators, switches, and optical filters [116]. In computation systems, they can be used to perform multiplication operations, through amplitude modulation [37]. MRs are fabricated with a ring-shaped silicon photonic waveguide (see Figure 87). An MR can be in one of two different states of on- or off-resonance, based on which the optical signal can be switched to different ports. The resonant wavelength of an MR can be tuned using electro-optic (EO) or thermo-optic (TO) effects of silicon that alter the effective index of the composite waveguide of the device. In a communication system, active MRs or MRs with a tuning circuit are used to filter wavelengths that correspond to 0s in an on-off keying (OOK) modulation scheme. In advanced modulation schemes such as 4 pulse amplitude modulation (PAM-4) [88], MRs can be used to modulate signal amplitude on four distinct levels. Multiple MRs sensitive to the same wavelength can be used for consecutive amplitude modulation resulting in parameter multiplication. Microdisk resonators [37] are similar to MRs but are composed of a disk structure instead of a ring structure. They are more compact than MRs but have higher operation losses.

Mach-Zehnder interferometers (MZIs), are made of two 3-dB directional couplers and two waveguide arms with phase shifters. The phase shifters, implemented using electro-optic or thermo-optic tuning, can change the optical phase in one or both arms of the MZI, introducing constructive or destructive interference at the output, to switch an optical signal between the output ports. MZIs are applied to the design of optical modulators, switches, and filters. MRs have a

smaller footprint and lower power consumption than MZIs. However, MZIs provide better thermal stability in operation and better extinction ratios than MRs .

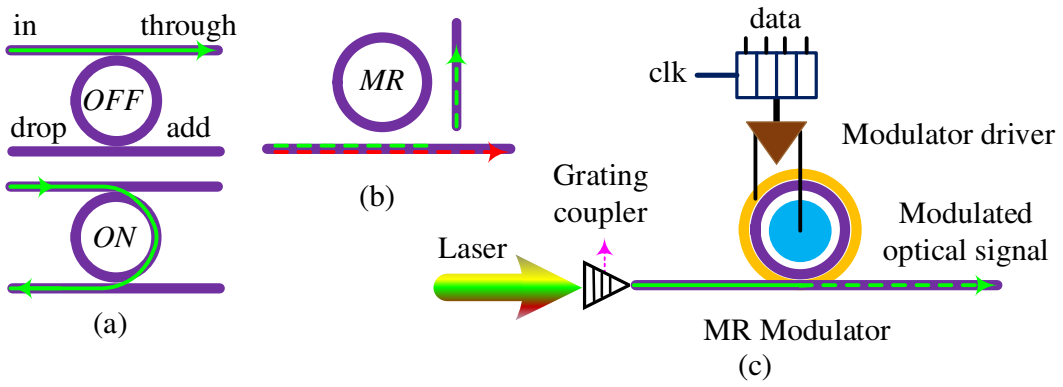


Figure 87. Microring resonator (MR): (a) off / on states, (b) MR filter, and (c) MR modulator.

Photodetectors (PD) are used to convert photonic signals to electrical signals. The operation of a PD may trade off bandwidth of operation with power efficiency. An efficient PD provides the desired electrical output with a small optical signal at its input. However, this small optical signal at the input of a PD may result in a low-bandwidth performance. For an efficient conversion of a photonic signal to an electrical signal, the intensity of the photonic signal received by the PD should be larger than the responsivity of the PD. High-bandwidth PDs can be employed in photonic computation to perform accumulation operations across signals of different wavelengths [32].

10.2 SILICON-PHOTONIC-BASED DNN ACCELERATORS

With the promise of improved energy efficiency and latency, silicon photonics for DNN acceleration has become increasingly prominent in both academic and industrial research [37]. Photonics is especially suited to accelerate DNN inference operations, which rely heavily on fixed matrix multiplications. The linear transformations involved in matrix multiplication can be

implemented efficiently in the analog domain using photonics. Silicon photonic DNN accelerators can be implemented as either coherent or non-coherent architectures, as described below.

Coherent architectures utilize a single wavelength and rely on constructive and destructive interference to change the relative power levels of a coherent optical beam [243], [162], [244]. Optical phase control is used to imprint the parameters onto the light wave signals. To achieve this, coherent architectures make use of MZIs, with phase modulators embedded on their arms. Weighting occurs with electrical field amplitude attenuation proportional to the weight value, and phase modulation that is proportional to the sign of the weight. Cascaded combiners, which facilitate coherent interaction of the signals, are used for accumulation. A lot of work in this field is focused on reducing the computational complexity of the DNN being implemented on-chip. The reduction of computational complexity is achieved using pruning methods [243] or singular value decomposition (SVD) [162], [244].

Noncoherent architectures, such as [43], [46], [111], [138], [44], [47] use multiple wavelengths, where each wavelength can be used to perform computations in parallel. In these architectures, parameters are imprinted onto the signal amplitude using wavelength-selective devices, such as MRs. Several prior works, as mentioned above, have explored DNN acceleration using non-coherent photonic principles. In [43], an MR-based DNN accelerator architecture was proposed which utilizes modular vector-dot-product units with optimized MR designs and tuning circuit optimization, for energy and throughput efficiency. For further optimizing power and energy consumption of non-coherent accelerators, especially at the electrical-photonic interface, [46] employed heterogeneous quantization (i.e., potentially different parameter bit-widths for each DNN layer) along with hardware-software co-optimization. For lowering area and power consumption, the work in [111] utilized microdisks instead of MRs. For further reducing the power

consumption at the electro-optical interface, binarized neural networks can be considered. A microdisk-based photonic accelerator was proposed in [138] for fully binarized DNNs (single-bit weight and activation parameters). While fully binarized neural networks offer higher efficiency in storage and power consumption, they may lack in achievable accuracy. To tackle the accelerator needs of partially binarized neural networks, the work in [44] proposed an MR-based partially binarized DNN accelerator. Non-coherent architectures have also been proven effective for RNN acceleration, as shown in [47]. In [47], the speed of operation of the photonic accelerator substrate was used to perform large-scale matrix operations needed for different types of RNNs, including deep long short-term memory (LSTM) and gated recurrent unit (GRU) models.

10.3 SILICON PHOTONIC INTERPOSER NETWORKS

Conventionally, chiplet systems are packaged using passive [250] and active [251] electronic interposers. Compared to passive electronic interposers, active electronic interposers employ an interconnection fabric with logic elements, instead of only passive metal interconnects to offer better communication scalability. However, both active and passive electronic interposers are unable to efficiently support a system with a large number of chiplets due to the inherent limitations of metallic interconnects: high latency for long interconnects and low bandwidth per each interconnect.

On the other hand, as optical interconnects offer low latency and high bandwidth, a photonic interconnection fabric can be a promising solution for interposer designs. Therefore, photonic interposers have recently received much attention in chiplet systems [236], [237]. For example, [236] employs high-bandwidth arrayed-waveguide grating routers (AWGRs) to get around the high latency and low bandwidth of conventional electronic interposers used in chiplet systems.

Besides the high bandwidth and low latency in long interconnects, silicon photonic interposers are inherently capable of dynamic inter-chiplet bandwidth tuning. PROWAVES [237] describes a photonic interposer network that dynamically manages inter-chiplet bandwidth by tuning the number of active wavelengths with respect to the traffic load. Under a low traffic load, where low bandwidth is required, PROWAVES utilizes a smaller number of wavelengths and deactivates unused wavelengths in an off-chip laser to save power consumption. On the other hand, under a high traffic load, PROWAVES activates a larger number of wavelengths to offer a high bandwidth and, as a result, high performance for inter-chiplet communication at the cost of higher power consumption. To handle high traffic load in PROWAVES, a high bandwidth gateway on each chiplet is used (i.e., a gateway with a large number of wavelengths). However, in this architecture, the high bandwidth gateway can create congestion on the chiplet as all the nodes on the same chiplet utilize the same gateway to communicate with the interposer. Moreover, access to the gateway is not enabled in a fair manner for the nodes across the chiplets. For example, a node far from the gateway can encounter very high latency to reach the gateway.

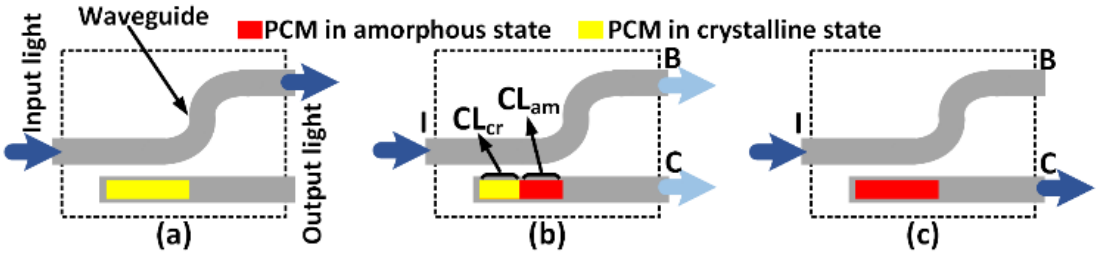


Figure 88. PCM-based coupler (PCMC) used in ReSiPI with three states: (a) crystalline, (b) partially crystalline, and (c) amorphous.

ReSiPI [252] improves on the PROWAVES design by employing several gateways on a chiplet with a relatively lower number of wavelengths. Moreover, ReSiPI manages inter-chiplet bandwidth while considering the online traffic by tuning the number of active gateways instead of

the number of active wavelengths. In the ReSiPI architecture, the traffic load of inter-chiplet communication is monitored in time epochs and the number of active gateways is defined based on the required inter-chiplet bandwidth. Activating or deactivating gateways is done using a phase-change-material-based coupler (PCMC), based on the coupler design in [253]. As shown in Figure 88, a PCMC can be in three states: 1) crystalline state to guide input light to the Bar (B) output, 2) partially crystalline state to guide a portion of input light to the Cross (C) output and the rest to the Bar output, and 3) amorphous state to guide the light to the Cross output. CL_{am} and CL_{cr} are the coupling lengths of the amorphous and crystalline states, respectively. By tuning the ratio of CL_{am} to CL_{cr} the appropriate input optical power from an optical laser to a writer gateway can be adjusted. Typically, in a silicon photonic network where a writer gateway modulates data on an optical signal to be received by a reader gateway, passive splitters are used to divide and deliver the optical signal from the optical laser to the writers. However, passive splitters prevent the network from dynamically deactivating writer gateways, while the PCMC can tune optical input of each writer and facilitate dynamic gateway activation and deactivation. Using the PCMC, the ReSiPI interposer is designed to reconfigure the number of active gateways and improve power consumption of the network. A controller is used to tune the number of active gateways in each chiplet according to the inter-chiplet traffic of that chiplet. Based on the number of active gateways, the PCMCs are tuned to deliver appropriate optical power to each gateway. Besides tuning the PCMC, the controller also tunes the laser power accordingly, to save the power consumption of the laser.

10.4 SILICON PHOTONIC 2.5D DNN ACCELERATORS

To explore the implications of accelerating DNNs on 2.5D interposer platforms, we present a case study that involves extending the CrossLight [43] photonic DNN accelerator to the 2.5D chiplet platform.

CrossLight is a neural network accelerator designed to perform high speed multiply and accumulate (MAC) operations in the photonic domain. However, the original monolithic CrossLight architecture suffers from low scalability and relatively low energy efficiency. We propose to use a ReSiPI-based photonic interposer architecture to design a more scalable and energy-efficient 2.5D CrossLight implementation. A high-level overview of our chiplet-based 2.5D CrossLight accelerator with a photonic interposer is shown in Figure 89. Several chiplets are packaged on a silicon photonic interposer substrate. We consider different types of chiplets as part of a heterogeneous architecture. Such a heterogeneous design allows system-on-chip (SoC) designers to utilize appropriate off-the-shelf chiplets and create diverse 2.5D packages to meet their design targets [234].

The chiplets in the proposed architecture consist of various computational and memory chiplets. One or more chiplets consist of an optically-interfaced memory architecture, such as high bandwidth memory (HBM; shown in Figure 89), with a dedicated gateway to communicate with the rest of the system. Each compute chiplet (e.g., chiplets 1-4 in Figure 89) hosts several photonic MAC units and has its local gateway(s) to read data from the memory chiplets and write data to them through the interposer network. Each gateway has two main parts: electronic circuitry on the chiplet and a Microring Resonator Group (MRG) on the interposer. The electronic part of a gateway is connected to the microrings of an MRG using the microbump technology.

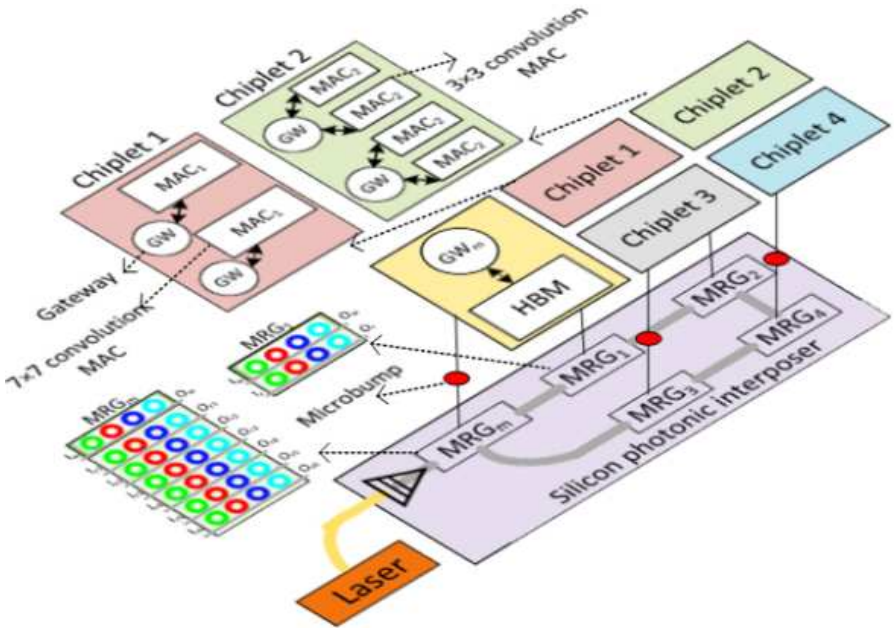


Figure 89. Overview of proposed 2.5D interposer chiplet-based DNN accelerator architecture.

The photonic MAC units utilize noncoherent photonics to perform multiply operations between parameters, and photodetectors are used to obtain the sum of products. The weights and activations are imprinted on wavelengths using banks of wavelength-specific MR filters. The imprinting process follows the broadcast-and-weight protocol as described in [38]. Even though the proposed design utilizes photonic communication to move data, moving multi-bit amplitude modulated data in a robust manner is challenging. Thus, the interposer relies on on-off keying (OOK) based data transmission, with intermediate photonic-to-electronic conversion at the gateways and buffering of the parameters at the MAC units. The buffered data is used to tune the respective MRs so that the parameter value can be represented using the wavelength amplitude. For tuning the MRs, EO tuning is used. The proposed architecture employs heterogeneous MAC unit sizes (size referring to the size of the vectors that can be deployed) across different chiplets to

cater to the different kernel sizes and to handle the large-scale MAC operations needed for the fully connected layers. For example, in Figure 89, Chiplet 1 includes 3×3 convolution MACs, while Chiplet 2 contains 7×7 convolution MACs. Moreover, as footprint of MACs with various sizes are different, the number of MACs per chiplet can vary for each chiplet. Figure 90 shows an overview of a MAC unit.

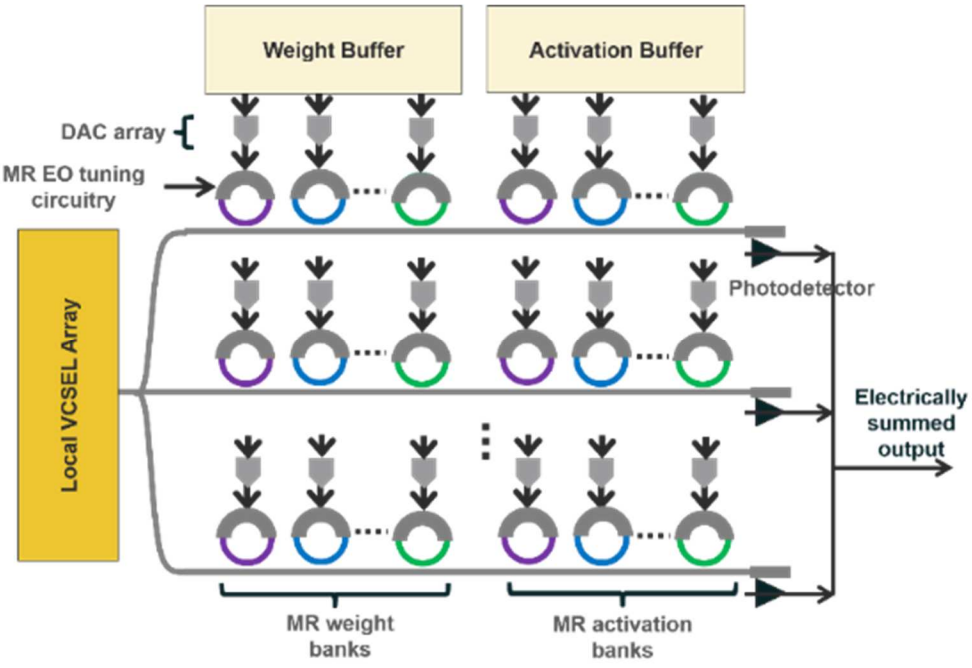


Figure 90. MAC unit architecture (DAC: digital to analog converter).

An example of the optical interface and communication on the interposer in this architecture is shown in Figure 91. In this example, MACs are reading data from the HBM on a separate chiplet. For successful communication, a writer gateway, including buffers to store and forward data, is utilized in the HBM chiplet, and similarly, a reader gateway is utilized on the chiplet with MAC units. The stored data in the buffers of the writer gateway is modulated on the optical signals which are generated by an off-chip laser. Different colors of modulators show that they are used to modulate different optical signals on different wavelengths.

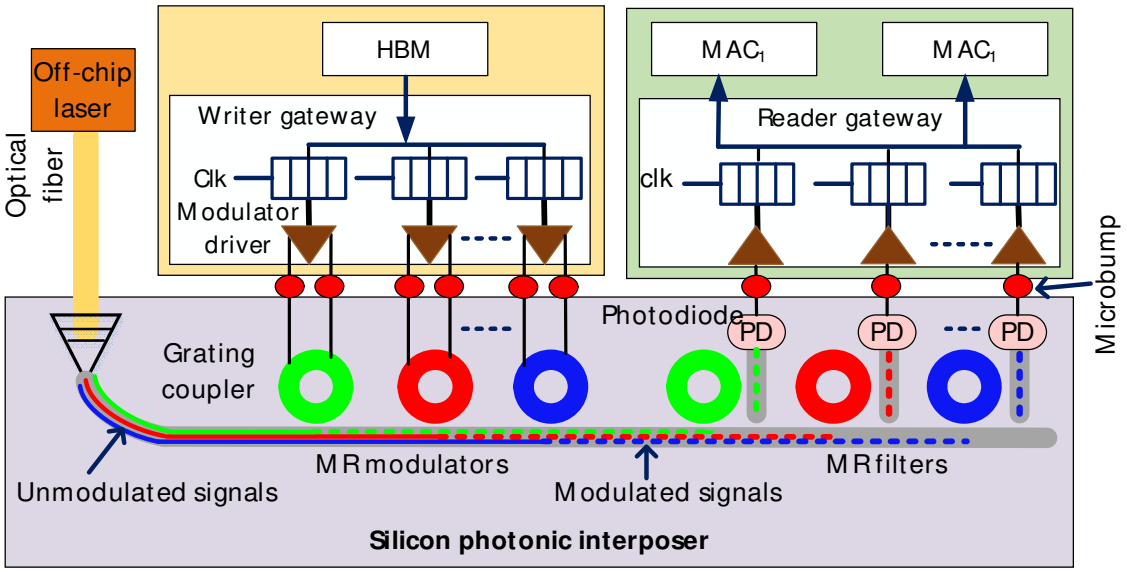


Figure 91. Example of optical communication on interposer: MACs are reading data from memory.

As discussed earlier, employing several optical signals with different wavelengths enables our network to transmit more data at the same time on the same waveguide, to improve the communication bandwidth. Several MR filters are also connected to the reader gateways. Each MR filter is tuned at a specific wavelength to filter and drop the specified optical signal. After this step, the optical signal is converted to an electronic signal using a photodiode, and this signal is delivered to the reader chiplet using microbumps. The reader gateway converts the electronic signal to digital data, and stores the received data in its buffer. Finally, the data will be forwarded to the MACs. Such a protocol for optical communication, where a reader is receiving data from a writer using a waveguide, is called the single writer single reader (SWSR) protocol. Similarly, if several readers are receiving data from a writer using a waveguide, the protocol is referred to as single writer multiple reader (SWMR) protocol.

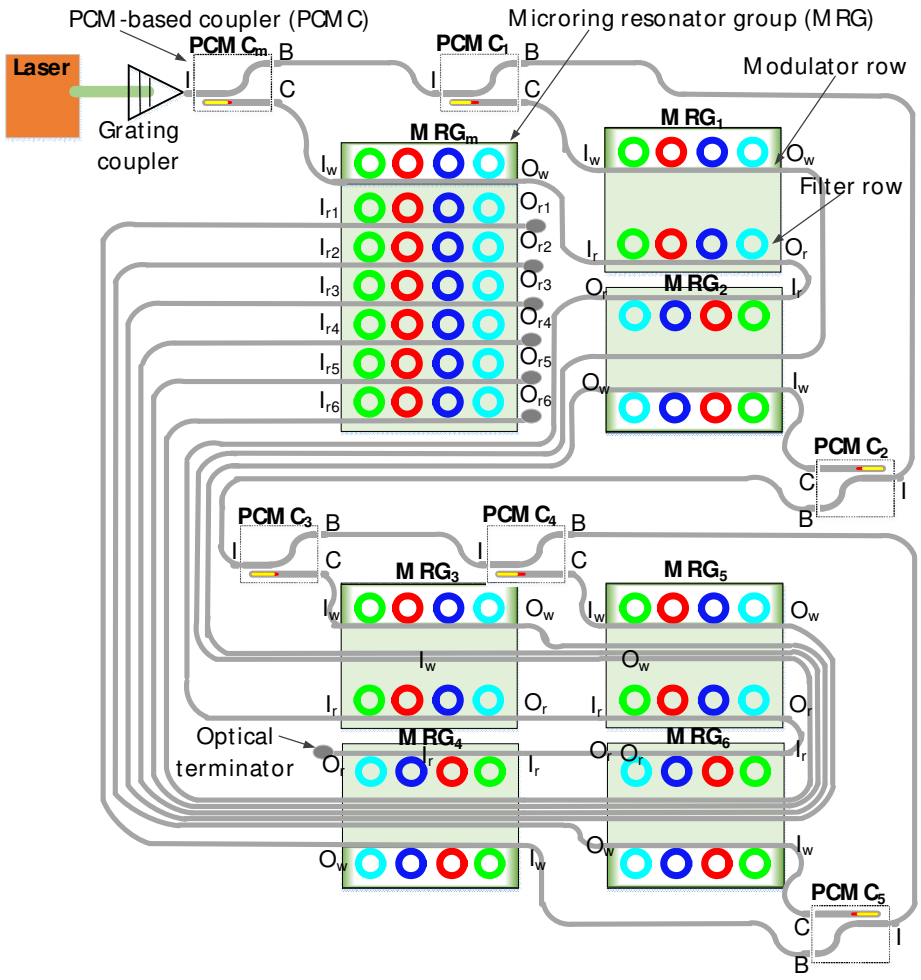


Figure 92. Silicon photonic network in our 2.5D chiplet-based DNN accelerator. Each MRG is connected to a gateway on a chiplet.

In our architecture, we have two types of traffic between the chiplets: 1) reading weights and inputs needed by MACs from memory, and 2) writing MAC outputs to the memory. As a result, from the memory chiplet to the compute chiplets, we utilize the SWMR protocol to perform reads from memory. Moreover, from the compute chiplets to the memory, we use the SWSR protocol. Therefore, the MRG of the memory chiplets requires several sets of MR filters (each set of MR filters is a row of the MRG shown in Figure 89) to receive data from the compute chiplets. On the

other hand, a compute chiplet requires only one set of MR filters as it only receives data from the memory. Both compute and memory chiplets require one set of MR modulators to send data.

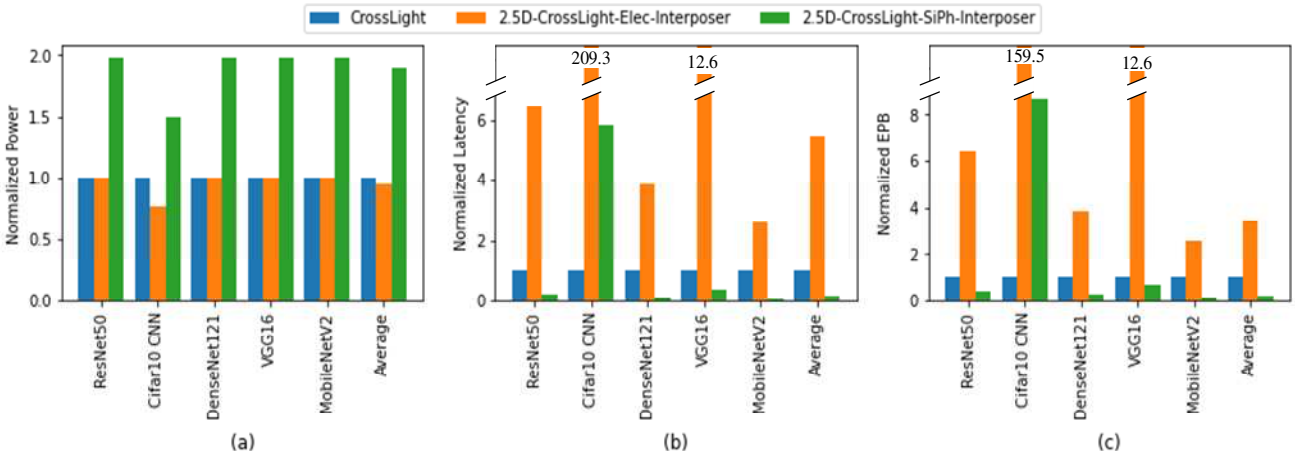


Figure 93. Performance analysis of CrossLight, 2.5D-CrossLight with electronic interposer, and 2.5D-CrossLight with silicon photonic interposer, (a) normalized power consumption, (b) normalized total latency, and (c) normalized energy-per-bit

An example of our 2.5D CrossLight with the integrated ReSiPI interposer is shown in Figure 93. Although this example is shown with six gateways (associated with one memory chiplet and five compute chiplets), the interposer design can be extended to a larger system without loss of generality. As shown in Figure 93, the MRG of the memory chiplet (MRG_m) has six filter rows to receive data from the six gateways of the compute chiplets (MRG_1 – MRG_6), while MRG_m has one row of modulators to send data to all the gateways. The photonic interposer network architecture is a passive network to save energy. This means that there is a specific waveguide to transmit data from each writer gateway to each reader gateway and the route (waveguide) does not change.

Table 25 Modeling parameters

Parameter		Value
Data rate of optical link (per wavelength)		12 Gb/s
Gateway frequency		2 GHz
Electrical network-on-chip link width		128 bits
Electrical network-on-chip frequency		2 GHz
Number of wavelengths		64
Number of memory-chiplets		1
Number of compute-chiplets		8
100 unit dense MAC	Number of chiplets	2
	Number of MACs per chiplet	4
	Number of MACs per gateway	1
7×7 convolution MAC	Number of chiplets	1
	Number of MACs per chiplet	8
	Number of MACs per gateway	2
5×5 convolution MAC	Number of chiplets	2
	Number of MACs per chiplet	16
	Number of MACs per gateway	4
3×3 convolution MAC	Number of chiplets	3
	Number of MACs per chiplet	44
	Number of MACs per gateway	11

Table 26 Considered dnn models in our evaluation.

Model	CONV layers	FC layers	Parameters
LeNet5	3	2	62,006
ResNet50	53	1	25,636,712
DenseNet121	120	1	8,062,504
VGG16	13	3	138,357,544
MobileNetV2	52	1	3,538,984

10.5 EXPERIMENTAL RESULTS

We designed two variants of the 2.5D CrossLight architecture: with a ReSiPI-based interposer [252] (*2.5D-CrossLight-SiPh-Interposer*), and an electrical mesh interposer [251] (*2.5D-CrossLight-Elec-Interposer*). We also compare the two 2.5D CrossLight variants with the original monolithic (single-chip) CrossLight architecture in terms of power, latency, and energy efficiency.

The model parameters assumed in this study are summarized in Table 25. We also employ the power model and power parameters used in [237] and [252]. We consider one memory chiplet and eight compute chiplets in which two of the chiplets include dense-layer MACs and six of them include convolution layer MACs (3×3, 5×5 and 7×7 convolution MACs). We used various DNN models, summarized in Table 26, for our evaluation.

The performance results are shown in Figure 93. In general, *2.5D-CrossLight-SiPh-Interposer* is able to achieve superior energy efficiency and latency across almost all models, except for very small ones (e.g., LeNet5). The heterogeneous chiplets and high bandwidth inter-chiplet photonic network enable more energy-efficient execution of DNNs than in the monolithic *CrossLight* case.

2.5D-CrossLight-SiPh-Interposer imposes a non-trivial power overhead as its photonic network consumes higher power for communication than an electronic network. However, *2.5D-CrossLight-SiPh-Interposer* has lower power consumption in the smaller DNN models (e.g., LeNet5) as the ReSiPI controller reconfigures the photonic interposer and deactivates unnecessary gateways. Nonetheless, for the smaller models, where each layer only takes up a small fraction of the overall compute real estate, the *2.5D-CrossLight-SiPh-Interposer* overheads become significant and adversely affect energy efficiency (e.g., LeNet5).

For larger models where multiple layers are mapped to chiplets, the *2.5D-CrossLight-SiPh-Interposer* overheads in terms of power consumption are amortized across these mappings. The controller also activates gateways in the large models to cope with high traffic volumes, which helps to improve inter-chiplet latency. Although *2.5D-CrossLight-Elec-Interposer* has lower power consumption, it suffers due to the significantly higher latency of metallic interconnects, especially for relatively long distances on large interposers.

On average, in comparison with monolithic *CrossLight*, *2.5D-CrossLight-SiPh-Interposer* shows 6.3× lower latency, which also results in 5× lower energy-per-bit (EPB). Compared to *2.5D-CrossLight-Elec-Interposer*, *2.5D-CrossLight-SiPh-Interposer* offers 34.2× lower latency and 17.2× lower EPB. Such significant improvement comes from the ability in *2.5D-CrossLight-SiPh-Interposer* to select appropriate chiplets to map layers of each DNN model and tuning the required inter-chiplet bandwidth accordingly. As *2.5D-CrossLight-SiPh-Interposer* performs well for larger models, such a photonics-based platform is scalable to support emerging large DNN model acceleration.

Table 27 Average power, latency, and energy-per-bit across electronic and photonic dnn accelerator platforms.

	Power (W)	Latency (ms)	EPB (nJ/bit)
CrossLight [43]	50.8	8	3.49
2.5D-CrossLight-Elec	48.3	43.8	11.9
2.5D-CrossLight-SiPh	95.93	1.28	0.69
Nvidia P100 GPU	250	13.1	12.3
Intel 9282 CPU	400	86.5	64.4
AMD 3970 CPU	280	141.3	73.7
Edge TPU	2	2366.4	17.6
Null Hop [155]	2.3	8049.3	68.9
Deap_CNN [110]	122	619.01	1959.4
HolyLight [111]	66.5	86.4	40.3

We also compared *2.5D-CrossLight-SiPh-Interposer* accelerator with state-of-the-art accelerators in terms of average power, latency (total latency of layers), and EPB (Table 27). *2.5D-CrossLight-SiPh-Interposer* can be seen to outperform these accelerators in terms of latency and EPB.

10.6 CONCLUSIONS AND OPEN CHALLENGES

In this chapter, we presented a 2.5D chiplet platform-based photonic DNN accelerator where both communication on the interposer and computation on the chiplets employ silicon photonics. Compared to a monolithic photonic accelerator, a chiplet-based one not only improves fabrication yield and cost, but also reduces latency using a high-bandwidth photonic network on the interposer. Moreover, chiplets can be designed heterogeneously and off-the-shelf chiplets can be integrated in 2.5D packages to make various systems with different computation power budgets and capabilities.

There are several open challenges in this field to design a more efficient silicon photonic DNN accelerator: 1) power consumption of the state-of-the-art photonic devices are relatively high, and there is a need for device-level efforts to design low-power devices; 2) designing an efficient electronic controller is essential to efficiently control the communication and computation operations with low latency; and 3) the silicon photonic 2.5D DNN accelerator architecture requires design-space exploration (e.g., in terms of the number of wavelengths, number of gateways per chiplet, and number of MACs per chiplet) to create an optimized architecture tailored to DNNs of interest.

CHAPTER 11: NONCOHERENT PHOTONICS FOR PERSISTENT MAIN MEMORIES

Over the past several decades, the emergence of big data and machine learning workloads has given rise to massive data-driven applications. These applications, which include large language models [254], intrusion detection systems [255], and graph processing frameworks [256], [257], consume and generate data at unprecedented rates, requiring data storage in the order of terabytes (*TB*) and memory bandwidths in the order of *TB/s*. Conventional electronic memory technologies such as dynamic random-access memory (DRAM) are struggling to keep up with such demands for increasingly higher bandwidth [258] and energy efficiency [259]. Additionally, DRAM technology also faces challenges associated with scaling towards the 10-nm technology node. Current DRAM nodes, such as Micron's 1α and 1β, are fabricated at 12–14 nm. At lower node scales, it has been shown that the DRAM cell's charge retention diminishes, cell structural integrity deteriorates, and delay and power penalties associated with bit lines increase dramatically [258]. While 3D-stacking technologies and through silicon vias have enabled high bandwidth memory (HBM), the increasing demand for capacity, throughput, and energy efficiency warrants the exploration of new main memory technologies.

Non-volatile memories (NVMs) address the data retention challenges in DRAMs and can help avoid the need for refreshes and associated latency concerns. But NVM candidates based on ferroelectric (FRAM) [260] and resistive metal oxide (RRAM) [261] technologies generally suffer from reliability and write endurance issues. To achieve higher reliability while retaining the advantages that NVMs offer, NVMs based on phase change materials (PCMs) can be considered [262], [263], [264]. PCM cells show higher energy efficiency, bit densities, and bandwidth than other NVM cell types [265], [266]. PCMs can transition between two material states: amorphous and crystalline. These states offer high resistance contrast between them and hence can be used to

store data as resistance levels. In electrically controlled PCM (EPCM) cells, the phase transitions are brought about by using current pulses. The state transition between amorphous and crystalline can be controlled to achieve different levels of crystallization of the PCM to achieve multi-level cells (MLCs) as well. But relying on PCM resistance as a way to represent data has caveats. The resistance levels achieved in PCMs have a non-linear dependence on the write voltage [267]. This makes achieving an intermediate state, between the fully amorphous and fully crystalline states, challenging. Furthermore, the written resistance level can also face resistance drift, limiting electrical PCM bit density to just a few bits per cell [268].

One solution to these limitations with EPCM cells is to utilize optically controlled PCM (OPCM) cells, where the PCM is deposited on top of a photonic (e.g., silicon-on-insulator (SOI)) waveguide. In such OPCM cells, state transitions can be achieved by using laser pulses. The power delivered by the laser pulses can heat up the material, enabling state changes between amorphous and crystalline states. The refractive index contrast between the states affects the optical transmission of the cell, enabling storing and reading out data optically. If the PCM candidate selected has a sufficiently high contrast between the two states, intermediate states between amorphous and crystalline states can be used to create PCM-based MLCs. Moreover, an optically controlled PCM memory comes with the added advantage of being able to leverage high bandwidth silicon photonic links for data transfer. Given the emergence of optical computing [37], an optical memory can also enable high-speed and energy-efficient fully photonic computing systems with minimal electro-optic conversions.

In this chapter, we present the design of the first cross-layer optimized optical PCM-based main memory architecture, named *COMET*. The proposed main memory system is characterized

by high bit density per cell, lower energy consumption, and high memory throughput and bandwidth compared to the state-of-the-art. Our novel contributions in this chapter are:

- We comprehensively explore different PCM candidates with the goal of selecting the most efficient PCM for optical memory applications;
- We design a low-loss and energy-efficient silicon photonic PCM-based multi-level memory cell as a basic building block;
- We design and optimize an all-optical, loss-aware silicon photonic PCM-based photonic main memory architecture; and
- We present detailed comparison of our designed photonic main memory architecture against state-of-the-art electronic and photonic main memory architectures.

11.1 BACKGROUND AND RELATED WORK

In this section, we discuss the fundamentals of PCMs and PCM-based optical memories.

11.1.1 PCM: FUNDAMENTALS AND PROPERTIES

PCMs are capable of changing phase from amorphous to crystalline, and vice versa, based on the thermal energy supplied to the material. The thermal energy supplied should be sufficient to change the temperature across the bulk of the material. This change in temperature should match the melting temperature (T_l; for phase change to amorphous state) or the crystallization temperature (T_g; for phase change to crystalline state). Amorphization is the more power-hungry process as $T_l > T_g$. Additionally, PCMs can be set to an intermediate state if the provided thermal energy converts only part of the material to either amorphous or crystalline state [264]. The energy necessary to achieve these state transitions can be delivered to the PCM electrically, thermally, or

optically. Microheaters can be used for applying thermal power directly, while PN junctions can be used to supply heat electrically. To trigger phase transitions optically, a laser pulse is required. The specific power and duration at which the laser pulse delivers thermal energy depend on the material and the energy required by that material to achieve state transition. Three widely considered phase-change materials in prior work include $Ge_2Sb_2Te_5$ (GST), $Ge_2Sb_2Se_4Te$ (GSST), and Sb_2Se_3 [264].

The change in PCM phase brings with it a change in the electrical and optical properties of the material. PCM's states have different electrical resistances. Typically, the high-resistance amorphous state is used to represent a binary 0, and the low-resistance crystalline state is used to represent a binary 1. This non-volatile change in resistance allows the PCM cell to be paired with an access transistor to form a 1T-1R EPCM memory, as described in many prior works (e.g., [269], [270], [271], [272]). But as discussed earlier, EPCM memories face many challenges, such as asymmetric and high write latencies [273], non-linear response to write voltage, and resistance drift.

Optical PCM-based (OPCM) memories depend on the change in the refractive index of the material phases. The change in *refractive index* changes the optical transmission across the cell, which allows data storage and readout. To implement such an OPCM memory effectively, understanding the optical properties of the PCM material is important. For PCM candidates, high refractive index contrast and hence contrast in optical transmission between the phases is essential. Having higher refractive index contrast between the amorphous and crystalline state enables better tolerance to optical signal losses and noise which can otherwise cause readout errors. This high contrast also allows multiple intermediate levels of phase transition to be achieved without them being susceptible to these same losses. In this regard, achieving high refractive index contrast

serves the same purpose as achieving high resistance contrast between phases in EPCM memories, and leads to better signal-to-noise ratio (SNR) at the readout.

A higher *extinction coefficient* between states is also an important requirement in an OPCM cell. Extinction coefficient in photonics is a measure of the optical power dropped from an optical signal as it traverses a material. Higher extinction coefficient indicates that the material extracts more power from the signal. The benefit of this metric depends on the application. From the perspective of propagation (e.g., in optical interconnects), high extinction coefficient is not preferred as this results in higher losses and power consumption, but for a filter (e.g., in optical switches), high extinction coefficient means that the device will be able to filter out as much of the optical signal as possible. In OPCM memory applications, a higher extinction coefficient in memory cells is beneficial. More efficient laser power absorption due to a high extinction coefficient in the crystalline state allows for energy-efficient transition to the amorphous state, and vice-versa.

11.1.2 OPCM MEMORY

Given a specific phase change material, it is essential to design an efficient memory cell that can grant access to the material for reads and writes. There are several ways in which the OPCM cell design has been approached in the literature, as discussed next.

The work in [274] proposed a simple crossbar-based cell design (Figure 94(a)) where the OPCM material is placed on top of waveguide crossings. The work proposed a main memory architecture called *COSMOS* using this OPCM cell design. Access to the cell in this architecture is provided through row and column access wavelength signals. These signals have to be present simultaneously to ensure write operations. The proposed architecture employed a subtractive read

approach where the entire subarray is read, followed by a reset signal to the row that needs to be read, erasing the contents, and finally, the subarray is read again. The two read values are then subtracted at the memory controller (MC) to obtain the intended row values. This approach paired with the 4 bits/cell assumption ensures a high bit density for such an architecture.

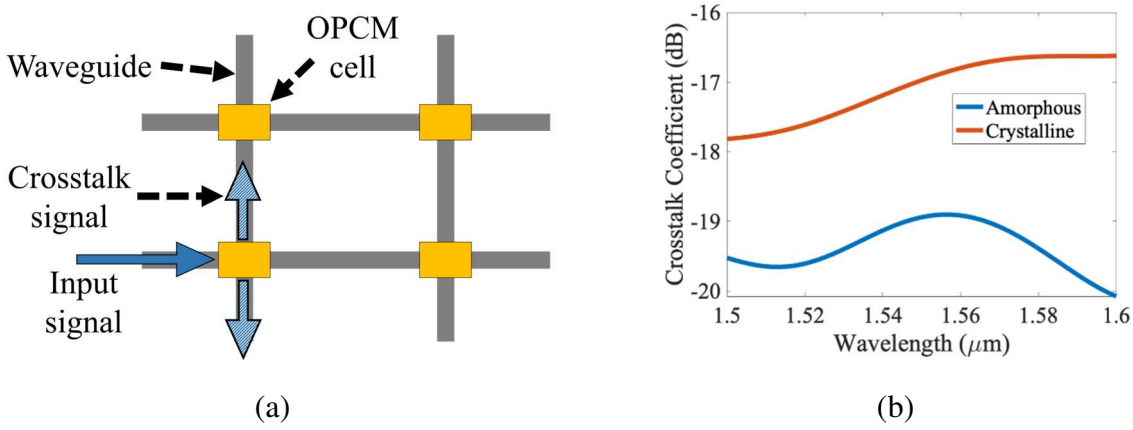


Figure 94. (a) OPCM array structure from [274]; (b) Crosstalk experienced in (a).

However, the cell design in the *COSMOS* architecture from [274] causes the rows to be susceptible to crosstalk from the write operations on the adjacent rows. The crosstalk signal (Figure 94(a)), although small, can trigger changes in the OPCM cell due to the thermo-optic effect [275]. The energy from the write pulses can cause temperature changes and hence refractive index changes in the adjacent cells, causing severe data corruption. This effect is exacerbated when multiple bits are stored per cell, as a small change in the OPCM's refractive index can considerably impact the data stored. *COSMOS* uses GST cells designed in [276] which require up to 750 pJ to operate. But *COSMOS* assumes a 135 pJ operational energy which is insufficient for GST cell operation. Even with a 750 pJ energy delivered at the crossbar, the thermo-optic effect due to the ~ (–18 dB) crosstalk (see Figure 94(b)) can introduce 12.6 pJ energy to the adjacent cells in the *COSMOS* architecture. This extraneous energy can trigger an 8% change in a neighboring OPCM

cell's refractive index, which can easily alter data stored in a cell with 16 programmable refractive-index levels (i.e., 4 bits/cell) [275] with the < 8% contrast between levels assumed in *COSMOS*. Without corrective measures after every write operation, data stored in the *COSMOS* architecture can get severely corrupted as shown in Figure 95. This is without considering the fact that a contiguous array of GST cells that read out data through other cells in the row will have to account for the optical losses as the signal passes through subsequent GST cells. Without accounting for these variable losses, as different GST cells will store different data and hence create different losses, the readout from the top row of the array may not reach the controller for detection. These losses can range from 0.24 dB for the amorphous state cells to as much as 21.8 dB for cells in the crystalline state. This further renders the proposed read and write approaches in [274] prone to severe error.

The issues highlighted above make the crossbar-based cell design from [274], although attractive from the bit density perspective, unusable from a memory reliability perspective. To ensure proper data retention and readout, the memory cells have to be isolated and access control mechanisms need to be in place. This can be achieved by using microring resonators (MRs) as access control mechanisms for these cells, as described in [277], [278]. These works proposed using thermo-optic tuning to regulate MR operation and grant photonic access to the cell. However, thermal tuning [39] to ensure that the MR is in-resonance (access granted) or off-resonance (no access to cell) is a slow process with microsecond (μs)-scale latencies. Although this is still much faster than the refresh mechanisms that have millisecond(ms)-scale latencies in DRAMs, the tuning has to be performed every time a cell is accessed and hence it will severely increase the latency and reduce achievable bandwidth of the OPCM memory. In our design (described in the next section), we propose using electro-optic tuning [40], where the resonance of

the MR is controlled using carrier injection through a PN junction at nanosecond(*ns*)-scale latencies. Note that this imposes increased optical losses which must be accounted for in our design considerations, which is discussed in the next section.

Figure 95. Data corruption in crossbar-based OPCM memory from [274] due to crosstalk; (*left*) original image; (*right*) image after 4 writes to adjoining rows.

Once the cell design is finalized, multiple cells can be tiled to form subarray and bank structures to realize a photonic main memory architecture. We discuss our material selection, cell design, and architecture-level design for *COMET* in the next section.

11.2 *COMET* OPCM-BASED MAIN MEMORY DESIGN

In this section, we describe the components of our proposed *COMET* OPCM-based main memory architecture. Our memory architecture ensures reliable data writes and reads while achieving high data throughput. The memory cells have access control mechanisms that isolate the cells from each other and ensure crosstalk elimination, to prevent data corruption. We also adopt a combination of mode-division multiplexing (MDM) and wavelength-division multiplexing (WDM) to enable parallel accesses for reads and writes. This access mechanism allows *COMET* to be designed as a hierarchical multi-banked design, rather than a simple array of memory cells.

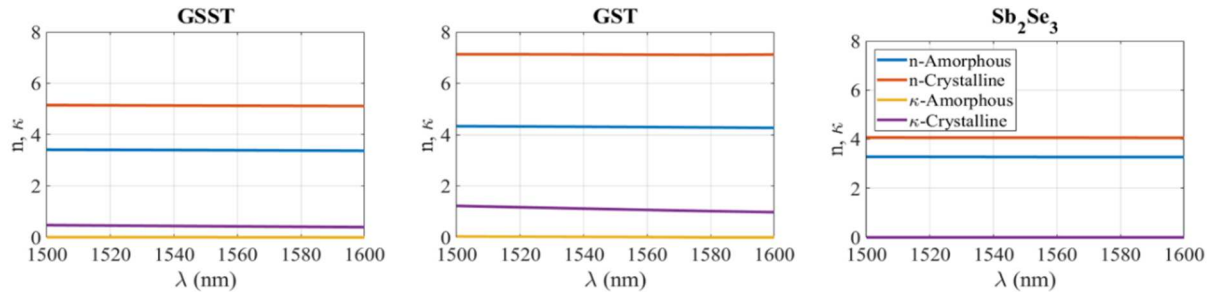


Figure 96. Comparison of the refractive index (n) and extinction coefficient (κ) between GSST, GST, and Sb_2Se_3 in the optical C-band range.

11.2.1 PHASE CHANGE MATERIAL SELECTION

The refractive index contrast and extinction coefficient are two of the most critical design parameters to consider when determining the most suitable material for OPCM-based main memory applications, as discussed in Section 11.1.1. The refractive index contrast and extinction coefficient of three well-known PCM candidates $Ge_2Sb_2Te_5$ (GST), $Ge_2Sb_2Se_4Te$ (GSST), and Sb_2Se_3 can be modeled using the Lorenz model [279]. We modeled and analyzed the two key design parameters for the PCM candidates, with results shown in Figure 96. It can be observed that for C-band (1530–1565 nm), GST exhibits the highest refractive index contrast (difference between blue and yellow lines, and difference between red and purple lines) and high extinction coefficient between amorphous and crystalline states. This makes GST the most suitable candidate for OPCM-based memory cells. Thus, we consider GST for our cell design. Prior works have also demonstrated that it is possible to store more than 34 unique states in GST-based OPCM memory cells [276], thus enabling up to 5 bits/cell capacity in OPCM memories.

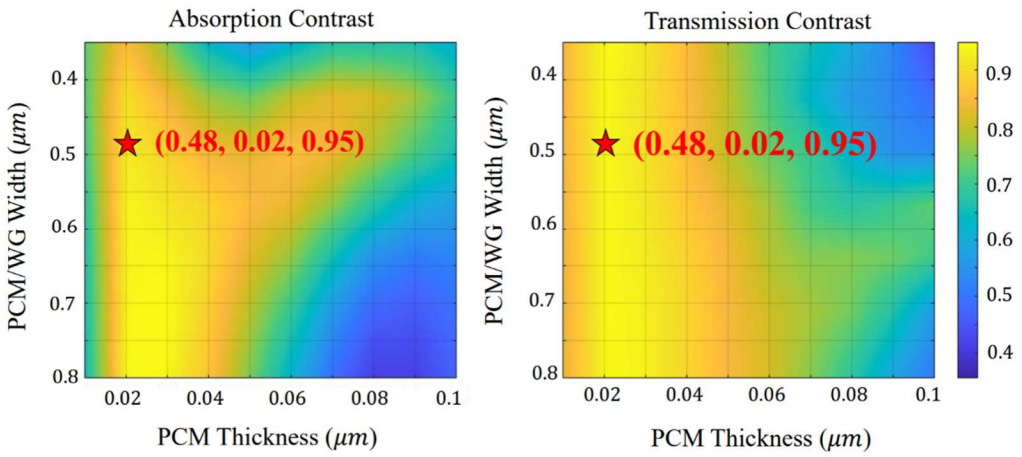


Figure 97. Optical absorption contrast and optical transmission contrast of GST cell for different cell geometry (width and thickness) in the *COMET* architecture. The stars represent the geometric configuration selected with values for (width, thickness, absorption or transmission contrast ratio), based on our analysis.

11.2.2 OPCM MEMORY CELL DESIGN

In our OPCM cell design, we consider GST deposited on a silicon-on-insulator (SOI) strip waveguide. As it was shown in Figure 96, the extinction coefficient of the GST in the crystalline state is much higher than its amorphous state. This leads to negligible optical transmission due to high absorption of the electric field in the PCM. The optical absorption contrast and optical transmission contrast between fully crystalline and fully amorphous state for OPCM memory cells of different geometries and materials are shown in Figure 97. Note that the optical transmission contrast is not only a function of optical absorption in the cells but also partially originates from the optical-refractive-index mismatch between the PCM and SOI waveguide due to high refractive-index contrast between silicon and GST. To avoid optical-refractive-index mismatch when designing GST-based OPCM memory cells, it is important to select a design where both optical transmission contrast and optical absorption contrast are maximized. Doing so ensures that

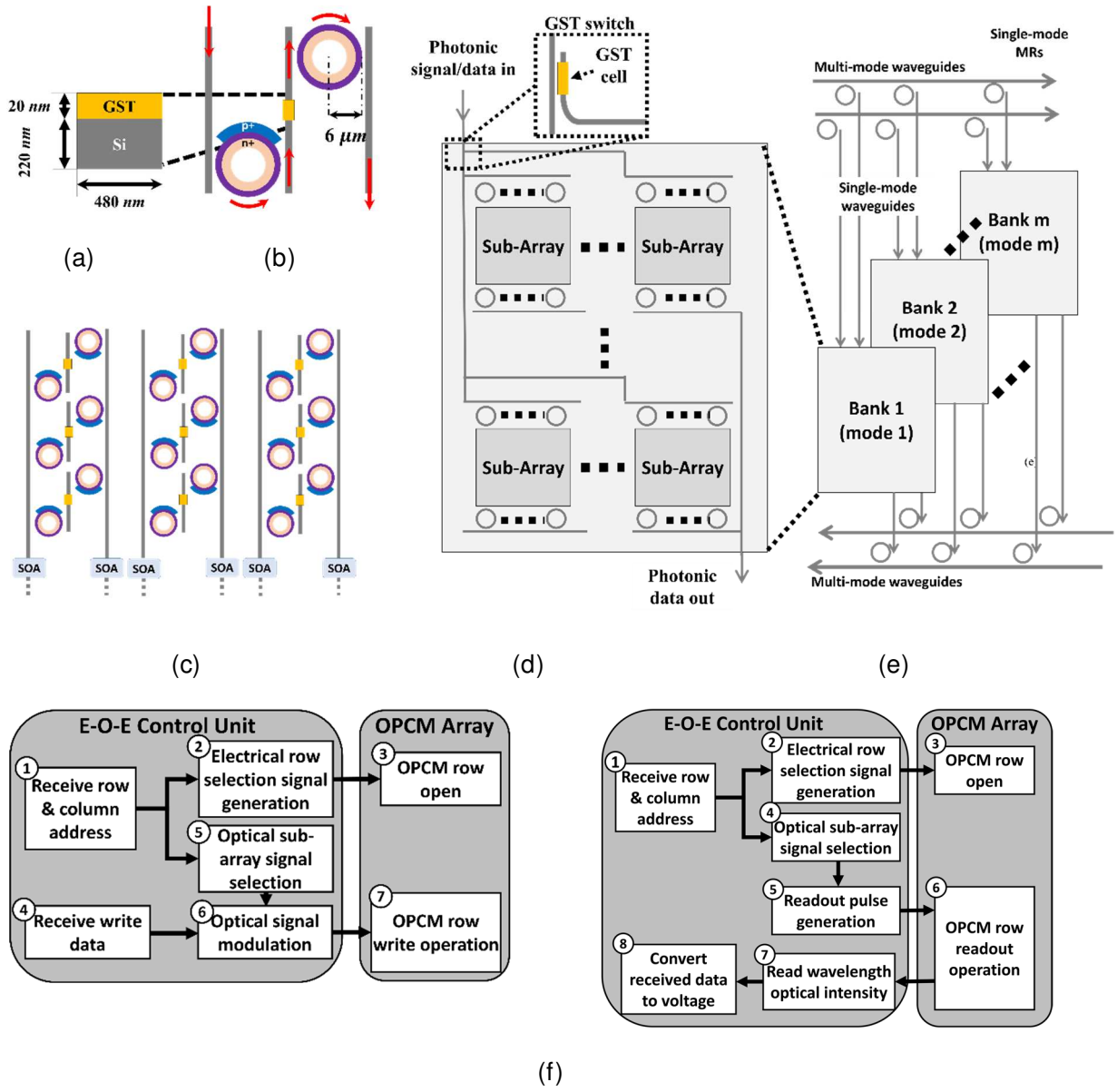


Figure 98. (a) GST cell designed and simulated for chapter. (b) Our proposed memory cell with MR-based access control. (*c*) OPCM memory array with intra-subarray semiconductor optical amplifiers (SOAs). (d) Single bank with multiple subarrays, with inset showing GST based signal switch. (e) Overall multi-bank architecture of *COMET*. (f) Steps during write (left) and read (right) operations in *COMET* architecture.

the optical transmission contrast stems from the optical power absorption which can be controlled by crystallization of the GST.

For a 2 µm long GST cell considered in our study (Figure 97), optical transmission and absorption are at 95% when the thickness of the cell is 20 nm. Note that the impact of PCM waveguide (WG) width on optical transmission and absorption is negligible. The SOI waveguide has a width of 480 nm (to ensure the single mode transmission of the light) and a thickness of 220 nm, where the GST deposited on it has the same width, but a thickness of 20 nm (Figure 98(a)). We have designed the GST cell with a small thickness, as a higher thickness makes heat transfer over the volume of the cell slower, hence leading to higher write and reset latencies. We also opted for a silicon (Si) waveguide instead of silicon nitride (SiN) waveguide, as Si offers higher transmission contrast between crystalline and amorphous states of the cell. Si also offers higher mode confinement over SiN waveguides, leading to lower propagation losses. Overall, us ing Si platforms can lead to a more compact GST cell footprint, lower amorphization time, and higher transmission contrast between amorphous and crystalline states [264].

To obtain the optical transmission characteristics of the GST cell we designed, we used FDTD simulations from Ansys Lumerical FDTD [216]. Since we are interested in OPCM MLCs, we obtain the refractive index profile of the intermediate states of phase transition using the Lorenz model [279]. The modeled refractive index profile was imported to the FDTD simulator. From these FDTD simulations, we were able to obtain the optical transmission levels of the intermediate states.

To obtain the energy and latency of transitions for intermediate states, Ansys/Lumerical HEAT [118] was used to solve transient unsteady-state heat transfer equations to capture the time-dependent temperature distribution over the OPCM's cell volume. In this simulation, a local uniform heat source was defined in the Si waveguide to mimic the power of the optical mode propagating in the waveguide. The regions of the GST cell which have a temperature between T_l

and T_g have a crystalline structure, whereas the regions with temperatures above T_l exist in an amorphous state because of the melt and quench mechanism [276]. We have considered two case studies, for two possible programming modes for an OPCM multi-level cell: (1) when the deposited state is crystalline, hence the reset state will be crystalline, and (2) when the deposited state is amorphous, hence the reset state is amorphous. Using the aforementioned methodology, a 4-bit OPCM GST cell with 16 distinctive and equally spaced transmission levels (with 6% spacing between transmission levels) was simulated. From these simulations, latency of transition, crystalline fraction in the transition level, and the optical transmission of the transition level were obtained, as shown in Figure 99. For case study (1), the reset pulse required 880 pJ of energy and for (2), the reset pulse required 280 pJ of energy.

As our architecture considers WDM-based data transfers, the performance of the GST cell, when exposed to different wavelengths, needs to be analyzed. Results for the C-band (1530–1565 nm; plots not shown for brevity) showed a linear decrease of loss from 0.073 dB/mm to 0.067 dB/mm for the wavelength range between 1530 nm to 1565 nm. In addition, the maximum wavelength-dependent transmission contrast between crystalline and amorphous states was calculated to be 1.4%. These results indicate that our GST cells can perform with low loss and low variation in transmission levels across the C-band.

After finalizing the GST cell design, we designed the OPCM memory cell which integrates the GST cell and regulates signal access to the cell. This access control provides cell isolation between adjacent GST cells and is necessary to avoid optical crosstalk and associated thermo-optic-effect-based data corruption, as discussed in Section 11.1.2. To ensure GST cell isolation, our memory cell has MR-based access control as shown in Figure 98(b). MRs are switched in and out of resonance to enable and disable the signal access to the GST cell. We use MR designs from [280]

with 6 μm radius, for low loss and efficient coupling during EO tuning to allow as much of the read/write signal to reach the GST cell from the Si waveguides. The signal enters the cell through the left waveguide and follows the path shown using red arrows in Figure 98(b) and exits through the rightmost waveguide. As discussed in Section 11.1.2, we have opted for electro-optic tuning for tuning the MRs in and out of resonance for the 2 ns access latencies this access mechanism provides [280].

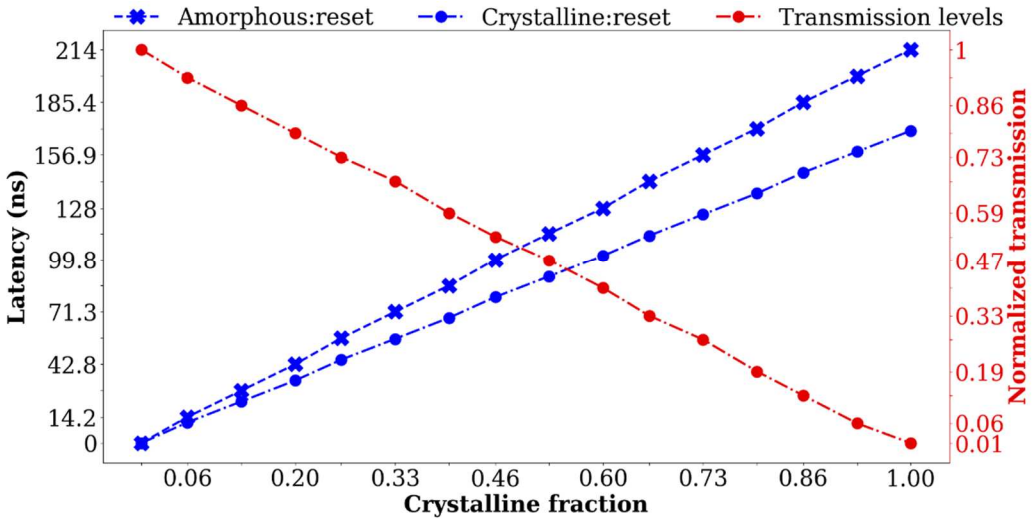


Figure 99. Latency and optical transmission for 16 crystalline-fraction levels representing the intermediate states in our designed GST cell.

11.2.3 *COMET* MEMORY BANK ARCHITECTURE DESIGN

The OPCM memory cell can be tiled to form an array, as shown in Figure 98(c). The array is comprised of columns of the designed OPCM memory cell, with each column assigned a single wavelength for accessing the GST cell. With this arrangement, the GST cells stay isolated from each other. The row access of the array is provided through simultaneous electrical control of the PN junctions of the MRs in the row, shifting them into resonance and, in doing so, allowing the wavelengths access to the GST cell.

The memory bank thus formed has $N_r \times N_c$ OPCM cells, with a total capacity of $N_r \times N_c \times b$, where b is the bit capacity of the OPCM MLC. This bank has to be further divided into subarrays to enable energy-efficient access. A bank will contain S subarrays, each containing $M_r \times M_c$ cells, such that $N_r = S_r \times M_r$ and $N_c = S_c \times M_c$. When a subarray has to be accessed, M_c wavelengths need to be modulated, and $2 \times M_c$ MRs per bank have to be tuned to be in resonance. The memory bank requires N_c wavelengths to operate. We consider an off-chip laser to provide the N_c wavelengths in this chapter. Consequently, N_c MRs are needed to allow access to the columns, and another N_c MRs are needed to allow readout from the columns.

To access such a subarray system, optical splitters and couplers are required, which essentially multiply the laser power needed. In *COMET*, we utilize electrically controlled GST-based waveguide switching [252] to allow efficient access to our subarrays. The GST cell is inserted at the waveguide coupler and is switched from crystalline (no coupling) to amorphous state (coupling) to allow access to the subarray (Figure 98(d)). This adds a loss of 0.2 dB (for amorphous GST) when the wavelengths are coupled, and a GST switching time of 100 ns, but significantly reduces overall laser power requirement. The wavelength modulation and the MR tuning signals are generated at the electrical control unit, which interfaces the OPCM memory banks to the memory controller and processor. As general-purpose computing is still only feasible using electrical circuits, this electrical interface is necessary.

Note that in an $M_r \times M_c$ subarray, to preserve the integrity of the data being read and to ensure that the data signals reach the electrical interface, several design choices must be made. Depending on b, the data being read from the cells can only suffer so much loss before the transmission level becomes similar to the expected transmission level for other data. For example: For $b = 2$, the transmitted signal can suffer up to 25% or 1.2 dB of losses before a readout of ‘10’ becomes the

same as the readout for '01'. For $b = 4$, this tolerance becomes even lower, with the signal only being able to suffer less than 6% losses or 0.26 dB before the readout becomes erroneous.

We integrate semiconductor optical amplifier (SOA) based gain tuning within and outside the banks and subarrays to address data integrity issues (see Figure 98(c)). At the electrical interface level, there needs to be row-wise loss-aware signal amplification so that the row-wise losses can be accounted for to preserve data integrity. Since these losses and the expected tolerances will be fixed once the design is finalized, for gain tuning at run-time, a look-up-table (LUT) based required gain storage is utilized. Based on the row address, the LUT can provide the gain necessary to tune the modulated wavelength, using SOAs. However, this gain may not be sufficient to survive the losses within the M rows, with the losses from the EO-tuned active MRs. This necessitates SOAs within the subarray at regular intervals to boost the readout signal. These intra-subarray SOAs need only provide the same amount of power as the input laser signal to the bank (1 mW for crystalline reset programming and 5 mW for the amorphous reset programming). These SOAs play an important role in compensating for the EO-tuned MR through losses, for both read and write signals. We consider the power and latency overheads of the SOA and LUTs in our analysis presented in Section 11.3.

COMET is architected as a multi-bank OPCM memory (see Figure 98(e)), enabled through the combination of mode-division multiplexing (MDM) and wavelength-division multiplexing (WDM). We enable parallel access across banks, and the interleaved cache lines using MDM. This requires designing silicon photonic links with an MDM degree of B to enable this access. There are several considerations to be made in selecting the MDM degree. The fundamental mode is the mode that gives the highest confinement of the electrical field in the waveguide and hence the lowest loss. As the number of modes increases, the confinement of the higher modes decreases,

since their effective index decreases. This causes the higher modes to suffer from higher losses. As higher order modes are excited, the field will become leakier with higher losses. In addition, to support higher order modes, the waveguide width of the links and devices also needs to increase. Prior works have shown an MDM degree of 4 is achievable on chip, without notable losses or area overhead [281]. So, for our architecture, we set the MDM degree (and hence B) to 4, to minimize overhead.

11.2.4 READ AND WRITE OPERATIONS IN *COMET*

For the read operation, we perform the steps as depicted in Figure 98(f), right. The isolated nature of our memory cells makes the read operation straightforward, with row access through EO tuning and column access through sending the readout pulses to the subarray. The row ID is obtained from the address and the EO tuning signals are sent to the MRs in that row. Depending on the subarray ID mapped from the physical address, the corresponding wavelengths are gain-tuned and sent to the OPCM banks. Once the data is received at the electrical interface it is demodulated using an MR bank and the received data can be passed on to the processor for further operation.

Similarly, for the write operation (see Figure 98(f), left), we follow a straightforward approach. The row ID is obtained from the address and row access is provided through tuning the MRs of the row into resonance. The column IDs are also obtained from the address and the corresponding wavelengths are gain-tuned and modulated to reflect the data to be written and sent to the OPCM subarray.

11.2.5 *COMET* POWER CONSUMPTION

The *COMET* architecture can achieve a capacity of $(B \times N_r \times N_c \times b)$ bits. With the architecture design, the power required to achieve this capacity can be calculated as follows. With our SOA-based loss mitigation strategy, we assume $M_c = N_c$, which implies, $S_c = 1$. The subarrays can be arranged in an array of $\sqrt{S_r} \times \sqrt{S_r}$ for layout and addressing. The M_r value would depend on the b value to ensure the cache line readout across the B banks. We consider intra-subarray SOAs which are able to provide a gain of 15.2 dB to their input signal [282]. Given that the EO-tuned MRs have a through loss of 0.33 dB, there needs to be an SOA array at every 46 rows. This necessitates a total SOA count of $\frac{B \times N_r \times N_c}{46}$. To minimize power overhead, we only enable the SOAs within the subarrays that are being accessed. Assuming 1.4 mW power consumption for 0 dBm i.e., 1 mW output [282], total SOA power consumption at any instance during *COMET* operation will be $\left(\frac{B \times M_r \times M_c}{46} \times 1.4\right)$ mW.

Apart from these power considerations, there is a need for considering power consumption of the photonic links. Our WDM-MDM link requires N_c wavelengths to operate and requires $2 \times B \times N_c$ MRs to access the OPCM arrays. These MRs can be completely passive as they need not perform any switching operations. The laser power consumption can be calculated based on the various losses the signal will experience on its way to and from the OPCM arrays. These losses and associated power overhead are discussed in Section 11.3.

Finally, we need to consider the power consumption by the EO tuning mechanism within the OPCM arrays. Given that the MRs need to be tuned only within the row of the subarray being accessed, the tuning mechanism will contribute to $(B \times 2 \times M_c \times P_{EO})$ W of power. Here, P_{EO} is power consumption for EO tuning a single MR. This is further discussed in Section 11.3.

11.2.6 ADDRESS MAPPING IN COMET

To access the data within the memory banks, we need to perform an address mapping to the cells in our architecture. *COMET* can consider cache lines of various sizes (e.g., 32, 64, and 128 bytes), to reflect popular last level cache line sizes, interleaved across the B banks. The mapping process should perform the following mapping:

$$\{Channel_{ID}, Row_{ID}, Bank_{ID}, Column_{ID}\} \rightarrow \{Channel_{ID}, Subarray_{ID}, Subarray_{ROW}, Bank_{ID}, Subarray_{COL}\} \quad (74)$$

For our architecture, as described in Section 11.1.2, the channel and bank IDs can remain the same. Row_{ID} must be mapped to $Subarray_{ID}$ and $Subarray_{ROW}$ and $Column_{ID}$ to $Subarray_{ID}$ and $Subarray_{COL}$. The values for these parameters can be calculated as follows:

$$ID_1 = int\left(\frac{Row_{ID}}{M_r}\right) \quad (75)$$

$$ID_2 = int\left(\frac{Column_{ID}}{M_C}\right) \quad (76)$$

$$Subarray_{ID} = ID_2 \times \sqrt{S_r} + ID_1 \quad (77)$$

$$Subarray_{ROW} = (Row_{ID} \% M_r) \quad (78)$$

$$Subarray_{COL} = (Column \% M_C) \quad (79)$$

11.3 EXPERIMENTS AND EVALUATION

In this section, we describe our simulation setup, various design parameters considered in the design of the *COMET* architecture, and our approach for comparison with other main memory architectures.

We use a modified version of NVMain 2.0 [283] a main memory simulator that we heavily modified to accommodate 4-bit/cell MLC operation, our photonic memory configurations, and the addressing scheme. Our experiments and evaluations are based on an 8 GB main memory chip capacity. Performance metrics encompass application latency, bandwidth, and energy-per-bit (EPB). We benchmark *COMET* against *COSMOS* [274], a prominent OPCM main memory architecture, EPCM-MM [269], a proposed EPCM main memory architecture, and 2D and 3D configurations of DDR3 and DDR4 DRAMs (labeled as 2D_DDR3, 3D_DDR3, 2D_DDR4, and 3D_DDR4). Memory traces from the SPEC benchmark suite [284] are utilized for architecture evaluation. The various parameters we have considered for power modeling of the *COMET* architecture are provided in Table 28.

Table 28 Optical loss and power parameters considered for *comet* power modeling.

Loss parameters	Values
Coupling loss	1 dB [285]
MR drop loss	0.5 dB [286]
MR through loss	0.02 dB [124]
EO tuned MR drop loss	1.6 dB [280]
EO tuned MR through loss	0.33 dB [280]
Propagation loss	0.1 dB/cm [287]
Bending loss	0.01 dB/90° [288]
SOA gain	20 dB
Laser wall plug efficiency	20%
Power parameters	**Values**
EO tuning power (P_{EO})	4 μW/nm [40]
Max. power at GST cell	1 mW
Intra-subarray SOA power	1.4 mW [282]

11.3.1 MODELING COMET ARCHITECTURE

As described in Section 11.2.3, *COMET* has a capacity of $B \times N_r \times N_c \times b$ bits, with the array of $N_r \times N_c$ GST memory cells divided into subarrays of size $M_r \times M_c$ memory cells for enabling parallel and energy-efficient access. For bit density b, there are works which show the GST cell

should be able to achieve up to 5 bits/cell [276]. However, considering data distribution across cells, it is practical to consider bit densities in the multiples of 2, which allows for an even distribution of data across cells in a row and equal loss and crosstalk considerations to be made across all columns.

But as discussed in Section 11.2.3, at an architecture level, to allow high bit density per cell, several considerations must be made. To determine the optimal b value in *COMET*, we analyze how power, latency, bandwidth, and EPB varies for $b = \{1,2,4\}$. For $b = 1$, our architecture for 8 GB would be $(4 \times 4096 \times 512 \times 1024 \times 1)$, reflecting $(B \times S_r \times M_r \times M_c \times b)$. For $b = 2$, this would be $(4 \times 4096 \times 512 \times 512 \times 2)$. Finally, for $b = 4$, this configuration would be $(4 \times 4096 \times 512 \times 256 \times 4)$. We have opted to reduce M_c $(= N_c)$ over N_r $(= S_r \times M_r)$ as b increases, as reduced N_c results in the reduction of both WDM-degree requirement and significant intra-subarray SOA power reduction. Modifying N_c also allows *COMET* to retain its cache line capacity and deliver the same bandwidth across the designs.

There is also a variable LUT size requirement at the electrical-optical interface as b changes. The LUT size depends on the row frequency at which the SOA gain tuning is required. For $b = 1$, the loss tolerance is less than 50%, as there are only two levels stored per GST cell. The signal exiting the GST cell can suffer up to 3.01 dB of losses before it becomes error-prone. With MR through loss of 0.33 dB, the signal can pass 9 rows other than the source row, without error. For M_r of 512, the LUT requires 52 entries. Given the intra-subarray gain tuning at every 46 rows, these entries can be repeating values. Hence making the entry requirement just 5 parameters, with the gain parameter selected as per the $ceil\left(\frac{rowID\%46}{10}\right)^{th}$ entry of the LUT. For $b = 2$, the loss tolerance is lower at 25% or 1.2 dB. Following the same process as for $b = 1$, the LUT requires

12 entries, with the gain parameter selected as per the $ceil\left(\frac{rowID\%46}{4}\right)^{th}$ entry of the LUT. For $b = 4$, the LUT requires 46 entries with the gain parameter selected as per the $(rowID\%46)^{th}$ entry of the LUT. Since the LUT is a part of the memory control interface, we do not consider this power consumption as part of the OPCM memory operation, even though it is negligible. The overall power requirements for the $b = \{1,2,4\}$ configurations can be calculated as per the discussion in Section 11.2.5.

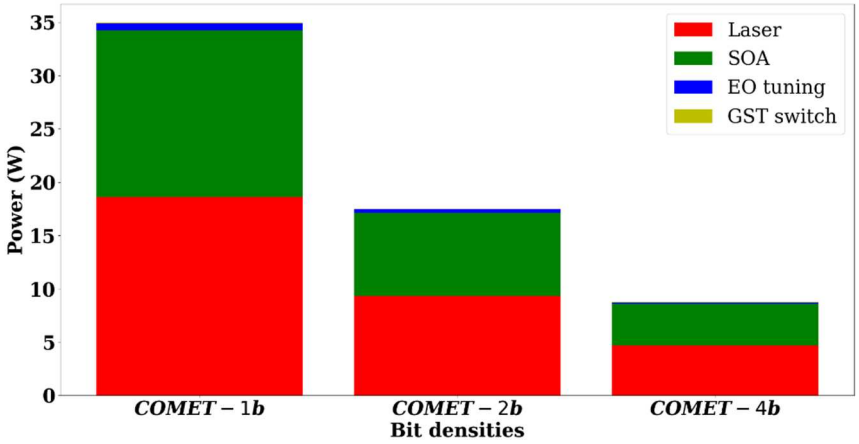


Figure 100. Power stack-plots for COMET with bit density (b) of 1 (*COMET*-1b), 2 (*COMET*-2b), and 4 (*COMET*-4b).

Table 29 Architectural details of photonic memory systems.

COMET	4 banks, 1 rank/channel, 1 device/rank bus width = 256 bits, burst length = 4 max. write time = 170 ns, erase time = 210 ns, read time = 10 ns, data burst time = 1 ns, electrical interface delay= 105 ns
COSMOS	8 banks, 1 rank/channel, 1 device/rank bus width = 128 bits, burst length = 8 write time = 1.6 μs, erase time = 250 ns, read time = 25 ns, data burst time = 1 ns, electrical interface delay= 105 ns

The resulting stacked power plots are shown in Figure 100. Based on these analyses, we have chosen $b = 4$ to keep the power overhead relatively low, and hence the memory configuration of $(4 \times 4096 \times 512 \times 256 \times 4)$ is considered in our subsequent comparative analysis.

11.3.2 MODELING COSMOS ARCHITECTURE

As *COSMOS* [274] is the only other photonic main memory architecture published in literature, we model it and compare our work against it. As discussed in Section 11.1.2, there are several challenges with obtaining the correct readout data and ensuring correct writes with a crossbar OPCM architecture as presented in *COSMOS*. We model *COSMOS* and update its design assumptions to ensure correct readouts, which is essential to realize a practical main memory architecture.

But before we address the crosstalk and data corruption issues in *COSMOS*, we must address the energy delivery assumptions. The GST cell design in *COSMOS* is taken from [276], which requires 5 mW laser pulses over 50 ns to 150 ns to deliver 250 pJ to 750 pJ in energy for phase transitions. *COSMOS* has retained the timing parameters from [276] but has reported the cells to require only 0.5 mW laser pulses. To ensure that the energy required for phase transitions is delivered to the memory cell, we have remodeled the timing constraints and assume 5 mW laser pulse power to ensure that the correct energy is delivered to the GST cells. We have opted to increase the timing parameters as increasing the power value would make the total power consumption entirely too high for an 8 GB main memory. The modified parameters for *COSMOS* (and also *COMET*) are provided in Table 29.

Data corruption in *COSMOS* is a result of the thermo-optic effect and the losses that the optical signals experience as they traverse the crossbar. As discussed in Section 11.1.2, the extraneous

energy is sufficient to trigger an 8% shift in crystalline fraction, due to how the cells are exposed to each other. We decided to not re-architect *COSMOS* to enable cell isolation as this will depart too far from the design in [276]. We instead opt to change the intermediate level count to allow for higher tolerance to this 8% shift in crystalline fraction. This necessitates dropping *COSMOS* bit density b from 4 to 2. This results in a memory architecture where $(B \times N_r \times N_c \times b)$ is $(16 \times 16384 \times 16384 \times 2)$ and $S_r \times M_r = S_c \times M_c = 512 \times 32$. For this version of *COSMOS*, we make the generous assumption that the MDM losses are negligible as designing 16 degree MDM silicon photonic links without losses is extremely challenging.

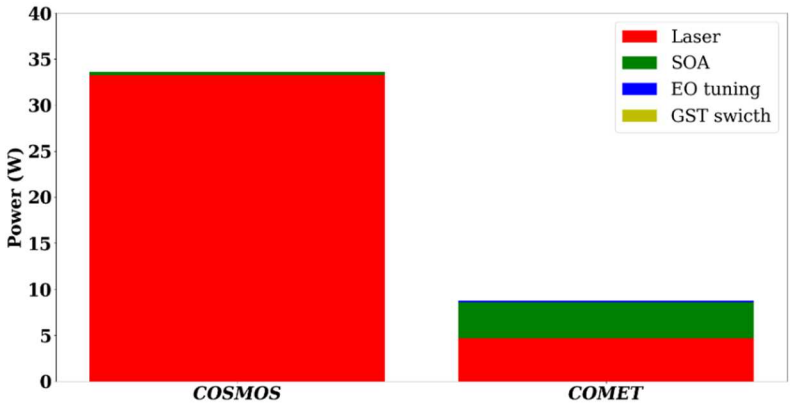


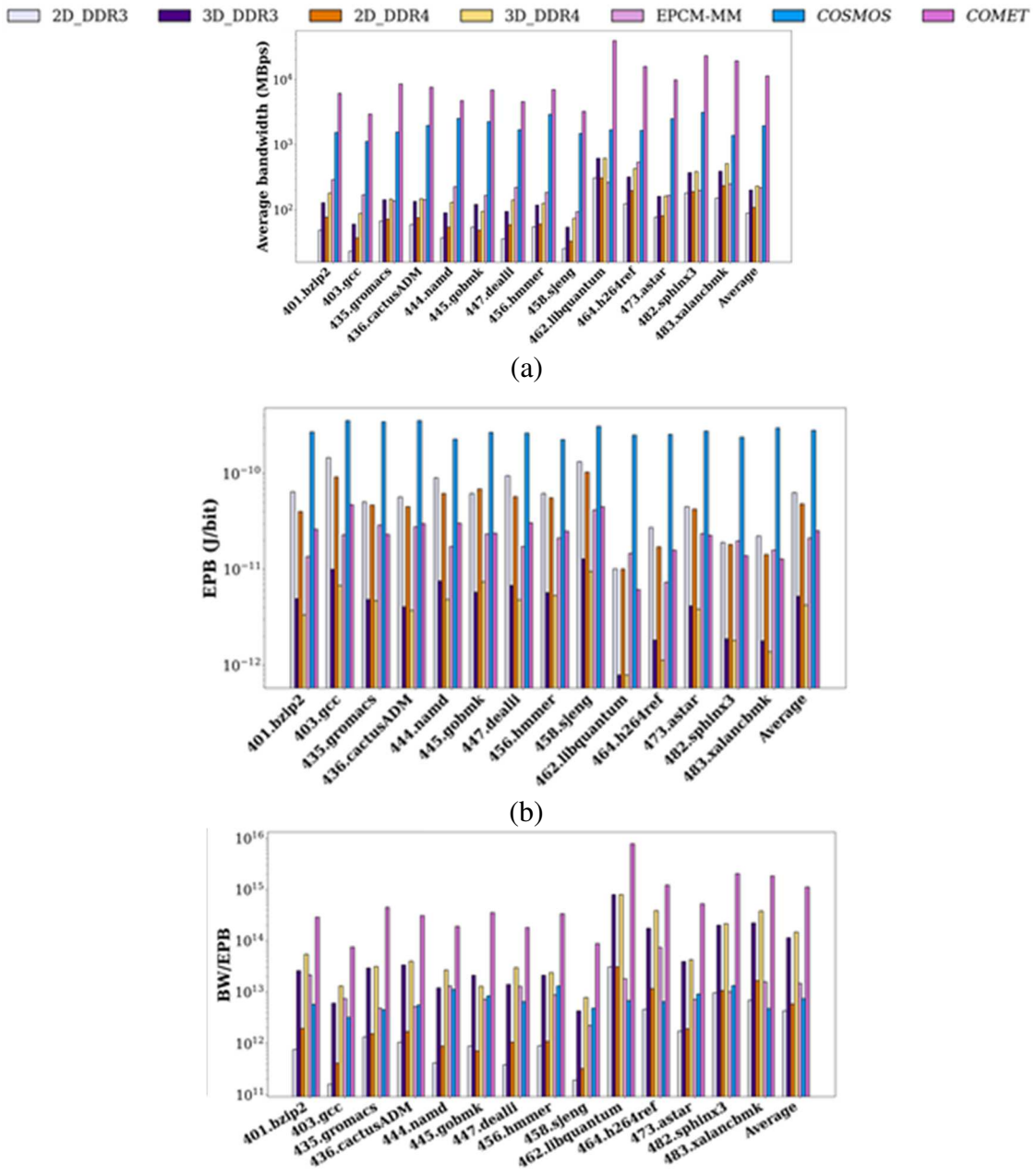
Figure 101. Power stack-plot for *COSMOS* and *COMET* architectures.

Additionally, we have to consider the losses that the read/write signals face as they traverse the GST cell array. We select 4 asymmetric transmission levels (0.99, 0.90, 0.81, 0.72) separated by 9% transmission to avoid the thermo-optic corruption, to represent the 4 levels necessary to represent 2 bit data, to avoid the high losses at high crystalline fractions. Keeping the subarray size of 32× 32 from COSMOS, the worst case loss of 1.4 dB (from transmission level 0.72) and the 15.2 dB gain for SOA [282], this also requires 6 SOA arrays (when considering both row and column losses) per subarray. There also needs to be dedicated data signal in and out ports for these

subarrays, to avoid further data corruption as the signal traverses the entirety of the 512× 32 GST cells per row/column. We assume these are passive MR-based ports. To avoid excessive splitter loss and hence laser power consumption, we also assume a PCM cell based subarray row access control in *COSMOS* (which we proposed for *COMET* in this chapter), separating the subarray rows. Overall, this adds 0.2 dB PCM switch loss to laser power calculations, some additional SOA power consumption, and a 100 ns delay while accessing the subarray row to *COSMOS*, which is similar to the overhead for such access control in *COMET*. Figure 101 shows the power stack comparison between *COSMOS* and *COMET*. The next subsection provides more detailed comparison results for latency, bandwidth, and EPB.

11.3.3 PERFORMANCE EVALUATION

We compare *COMET* in terms of bandwidth and EPB against 2D_DDR3, 3D_DDR3, 2D_DDR4, 3D_DDR4, the EPCM-MM, and *COSMOS* modified from [274], as discussed in the previous subsection. The absence of refreshes along with higher bandwidth provided by the silicon photonic interconnects allow both *COSMOS* and *COMET* to outperform their electronic counterparts in terms of bandwidth, (Figure 102(a)). *COMET* achieves 100.3×, 47.2×, 58.7×, 42.1×, 40.6×, and 5.1× higher bandwidth on average than 2D_DDR3, 3D_DDR3, 2D_DDR4, 3D_DDR4, EPCM-MM, and *COSMOS*, respectively.



(a)

(b)

(c)

Figure 102. a) Average bandwidth (BW); (b) energy-per-bit (EPB); and (c) BW/EPB, of applications across memory architectures.

In terms of EPB, it can be observed from Figure 102(b) that the 3D and PCM electronic counterparts are able to outperform both fully photonic memory systems. This can be attributed to the fact that the entire power consumption depicted in Figure 101 is utilized for orchestrating reads and writes for photonic memory. This is different from the case in an electronic memory where

only a very small portion of the overall power is required to orchestrate a read/write. In order to achieve comparable EPB values, more intelligent read/write orchestrations with dynamic power control are necessary for photonic memories. From Figure 101 it can be observed that laser power is a significant contributor to the overall power consumption of photonic memories. Enabling dynamic laser power management, such as that discussed in [191], could significantly improve photonic memory energy consumption. We leave the exploration of such techniques as future work. However, the high read/write bandwidth in comparison, enables *COMET* to outperform the 2D DRAM platforms. *COMET* is able to achieve 4.1×, 2.3×, 12.9× lower EPB than 2D_DDR3, 2D_DDR4, and *COSMOS*.

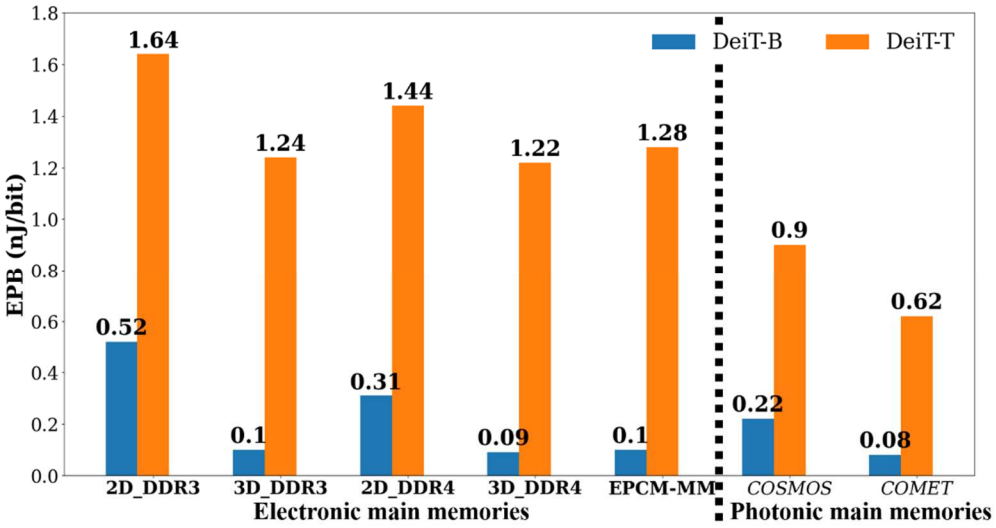


Figure 103. EPB of DOTA accelerator with different main memories.

COMET is able to contribute towards better overall energy efficiency of systems it is part of, owing to the combination of high bandwidth and comparable EPB performance to other main memory platforms. We showcase this using a BW/EPB metric (see Figure 102(c)), where *COMET* achieves 6.5× and 65.8× better BW/EPB over 3D_DDR4 (best electronic platform) and *COSMOS*, respectively.

11.3.4 PHOTONIC AI ACCELERATOR CASE STUDY

Photonic main memory architectures, such as our proposed *COMET*, are an excellent fit for emerging optical computing platforms. Over the recent years there has been several photonic computing platforms discussed [37]. However, to quantify the benefits of *COMET*, we consider DOTA [289], a photonic tensor engine-based transformer accelerator [10]. We analyzed how different electronic and photonic main memories impact the operation of. For this analysis, we considered the two transformer models DeiT-T and DeiT-B, as used in [289]. Figure 103 shows the EPB results for the various main memory architectures considered. Photonic memory architectures not only provide higher bandwidth (as discussed earlier and shown in Figure 102) than electronic memories, but also have the added benefit of being able to inject data directly into the photonic tensor engine, without the need for a energy-hungry electro-photonic conversion stages. *COMET*+DOTA achieves 1.3× and 2.06× lower EPB against 3D_DDR4+DOTA and 2.7× and 1.45× better EPB against *COSMOS*+DOTA. These results highlight the promise of photonic main memory for improving the performance of emerging optical computing platforms.

11.4 CONCLUSIONS

In this chapter, we presented *COMET*, a low-loss, low-latency, and high throughput OPCM-based main memory architecture that makes use of GST integrated with silicon-on-insulator strip waveguides. We described the cross-layer design and optimization of our *COMET* architecture from the material and device level to the architecture level. Our GST cell was designed and optimized by leveraging transient unsteady state heat transfer equations integrated with finite-difference time domain simulations. By using silicon, unlike silicon nitride platforms, our GST cell offered a high transmission contrast between crystalline and amorphous state of the cell (≈96%). In addition, the designed cell offers 16 distinctive transmission levels with 6% spacing

which makes *COMET* tolerant to transmission drift. Crosstalk-free, reliable memory operation was enabled with various loss-aware optimizations at the architecture level. Owing to these optimizations, *COMET* consumes only 26% of the power when compared to the best-known prior work on OPCM-based main memory architecture design. The low power consumption along with high-speed GST cell operations enable *COMET* to offer 7.1× better bandwidth, 15.5× lower EPB, and 3× lower latencies than the state-of-the-art OPCM-based main memory architecture.

CHAPTER 12: PROCESSING IN MEMORY USING NON-COHERENT PERSISTENT MAIN MEMORIES FOR MACHINE LEARNING INFERENCE ACCELERATION

For emerging machine learning (ML) models being used across application domains [290], [291], [292], the exponential growth in their computational demands has significantly outpaced the rate of advances in traditional computing architectures [17], [16]. The resulting "Von Neumann bottleneck" that alludes to the memory wall problem [293], is a critical challenge to overcome, to support modern ML workloads. In response to the limitations posed by the Von Neumann architecture, various alternative paradigms are being explored by industry and academia. A promising alternate computing paradigm involves in-memory computing or processing-in-memory (PIM) [294]. PIM architectures propose a departure from traditional designs by integrating processing capabilities within the memory subsystem. This integration aims to minimize data movement, reducing latency, and minimizing energy consumption associated with processing applications.

Given that dynamic random-access memory (DRAM) is the standard main memory technology today, it is an obvious candidate for PIM. Several prior efforts have focused on architecting DRAM-PIM [295], [296], [30]. However, conventional DRAM-based PIM systems have encountered challenges in achieving high throughput and energy efficiency. These challenges arise primarily due to internal data movement bottlenecks and the necessity for frequent memory refreshes. To address the energy and latency concerns associated with refreshes, non-volatile memory (NVM) technologies, such as ReRAM [297], [298], Spin-Transfer Torque RAM (STT-RAM) [299], and Phase Change Material (PCM) memories [300], have been considered. However, ReRAM and STT-RAM technologies face fabrication challenges and endurance issues [261],

[301]. ReRAM additionally suffers from resistance drift over time, which impacts data readout accuracy [261].

PCMs offer better energy efficiency, bit density, and bandwidth than other NVMs. They can switch between two physical states: amorphous and crystalline. This switch results in a contrast in electrical resistance, allowing these materials to encode information based on varying resistance levels. In the context of electrically controlled PCM (EPCM) devices, these phase changes are induced by applying current as part of microheaters. It is possible to precisely regulate the phase shift from amorphous to crystalline, enabling the creation of multi-level cells (MLCs) to store more data by adjusting the extent of the material's crystallization. However, utilizing the resistance in PCMs to encode data poses challenges as the resistance values that PCMs attain depend non-linearly on the applied write voltage [37].

To address these challenges, optically programmed PCM (OPCM) cells can be considered [51]. OPCM cells are fabricated with PCM deposited on top of a photonic waveguide and are programmed through laser pulses. Here, in place of resistance, the refractive index of the PCM is the physical property used to represent data. OPCMs can be programmed using laser pulses guided to them through on-chip waveguides. This makes them ideally suited for integration onto silicon photonic platforms. OPCMs are based on silicon photonics, which is an emerging field that integrates photonic systems with electronics. This platform offers several advantages over traditional electronic circuits, including high throughput and low energy consumption, for specialized computation tasks [37], [49], [48], [44]. Merging this computational capability with an OPCM main memory could allow for high-speed in-memory computation without the data movement and refresh bottlenecks seen in DRAM-PIM.

In this chapter, we explore how to architect a photonic main memory, to enable ML acceleration through PIM. We utilize the OPCM-based main memory from [51] as the backbone for our architecture and make several changes to it to support PIM. We have named our photonic PIM architecture for ML acceleration as *OPIMA*. We use convolutional neural networks (CNNs) to showcase the effectiveness of *OPIMA* for ML inference acceleration. The proposed PIM architecture is characterized by multi-bit density per cell enabling multiply and accumulate (MAC) operations to be performed directly within memory. This capability along with architecture-level innovations allows *OPIMA* to outperform the state-of-the-art in terms of ML inference throughput and energy efficiency. In summary, the novel contributions in this chapter include:

- Scattering and back reflection-aware OPCM cell design to maximize bit-density and minimize read errors per cell;
- Full system design of an OPCM-based PIM architecture that can operate as a main memory while performing PIM;
- Comprehensive comparison of operational efficiency of *OPIMA* with state-of-the-art accelerators.

12.1 BACKGROUND AND RELATED WORK

Before we discuss our PIM architecture and associated techniques, we review some fundamentals and background on PCMs, OPCM main memories, and photonic computing.

12.1.1 PHASE CHANGE MATERIALS (PCMS)

PCMs possess the ability to shift between amorphous and crystalline states, depending on the level of thermal energy applied. This energy must be sufficient to alter the material's temperature to either its melting temperature (T_i; for transitioning to the amorphous state) or its crystallization

temperature (T_g; for shifting to the crystalline state). Transitioning to the amorphous state consumes more energy because its required melting temperature exceeds the crystallization temperature. It should be noted that it is possible to induce partial phase changes within PCMs, creating intermediate states by converting only a fraction of the material to either state. These transitions can be initiated through electrical or optical means. Electrical heating can be provided through PN junctions whereas optically achieving phase changes requires a laser pulse, whose power and duration must be tailored to the material's specific transition energy needs. Common materials used for PCM applications include $Ge_2Sb_2Te_5$ (GST), $Ge_2Sb_2Se_4Te$ (GSST), and Sb_2Se_3 [264].

The change in a PCM phase brings with it a change in the electrical and optical properties of the material. PCM's states have different electrical resistances and different optical refractive indices. These differences in characteristics can be leveraged for data representation, including multi-bit data representation, enabling dense PCM-based memories and, as discussed in this chapter, PIM architectures.

For EPCM applications, the high-resistance amorphous state is used to represent a binary 0, and the low-resistance crystalline state is used to represent a binary 1. This non-volatile change in resistance allows the PCM cell to be paired with an access transistor to form a 1T1R EPCM memory cell and a corresponding memory array of these cells, as described in many prior works (e.g., [269], [270], [271], [272]). However, EPCM memories face many challenges, such as asymmetric and high write latencies [273], non-linear response to write voltages, and resistance drift.

OPCM memories rely on shifts in the material's refractive index to modulate optical transmission, enabling data storage and retrieval [264]. A deep understanding of a PCM's optical

properties is crucial for the effective deployment of OPCM memories. A significant refractive index contrast, ensuring a clear distinction in optical transmission between phases, is vital for reducing optical signal losses and noise [278], which could otherwise lead to readout errors. Similar to the importance of resistance contrast in EPCM memories, a high refractive index contrast improves the signal-to-noise ratio (SNR) during data readout. This is extremely important not just from a data fidelity standpoint but also from a photonic PIM standpoint, as we must ensure error-free data readouts to ensure error-free calculations in the analog domain where photonic computations occur.

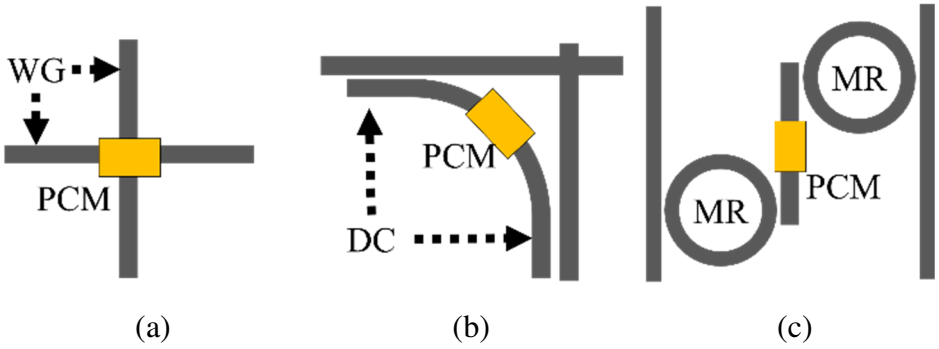


Figure 104. OPCM memory cells proposed in (a) COSMOS [274]; (b) Photonic tensor core [265]; (c) COMET [51]. WG: Waveguide; DC: Directional Coupler; MR: Microring Resonator.

12.1.2 OPCM MEMORY

A main memory architecture should have the ability to store large amounts of addressable data, which can be effectively retrieved and modified, whenever needed by the computing system. DRAMs achieve this by having row- and column-addressable memory cells, arranged into mats of cells, which in turn get organized into subarrays, and then banks. Collections of banks form memory chips, which are arranged into dual in-line memory modules (DIMMs) or 3D high bandwidth memory (HBM) architectures. Modern memory addressing schemes and memory controllers expect this style of data storage and management to be interfaced with them. So, it is

prudent to consider a similar style of data storage with OPCM memory as well. A few recent works have tackled the challenge of building an addressable OPCM memory [51], [274], which can be used for the DRAM-like memory organization described above.

The work in [274] introduced a straightforward design for a crossbar-based cell, illustrated in Figure 104(a), in which the OPCM is strategically positioned atop waveguide intersections. This cell design underpins the core of a main memory architecture called COSMOS. In this COSMOS OPCM memory, the mechanism for accessing data is facilitated through the use of specific row and column access signals that operate on distinct optical wavelengths. These signals are required to be activated simultaneously to enable successful write operations within the memory structure. COSMOS also adopts a subtractive read technique. This method involves initially performing a read operation across an entire subarray. Subsequently, a reset signal is dispatched specifically to the row selected for reading, which clears its contents. Following this reset, the subarray undergoes another read operation. By executing this sequential reading and resetting process, it is possible to extract the data from the intended row. The two obtained readouts are subsequently processed through subtraction at the memory controller (MC). This intricate process, when combined with the assumption that each cell can store up to 4 bits of information, significantly amplifies the bit density achievable by this architecture, presenting a substantial advancement in memory design aimed at enhancing data storage efficiency and capacity. However, this architecture is inherently susceptible to optical crosstalk as the data storage mechanisms end up interfering with one another. It is especially susceptible to thermal crosstalk from write operations from adjacent rows, especially when multi-bit storage is assumed, as shown in [51].

The work in [265] showcased an OPCM cell, originally devised for photonic tensor core operation, but deserves discussion as it has been used in [302] for their OPCM memory-based ML

acceleration work. The architecture has a crossbar structure to allow signals from orthogonal directions to interact with each other, enabling a wavelength-division multiplexing (WDM) based broadcast and weight computation technique [38]. The OPCM cell itself, however, is placed away from the waveguide crossing and can interact with a wavelength propagating along the horizontal waveguide. This interaction is enabled by the coupler on which the OPCM cell is fabricated. The coupler, as the name suggests, is a passive device designed to enable coupling between the WDM signal on the horizontal waveguide and the OPCM and does not offer wavelength selection like other photonic devices (e.g., a microring resonator (MR)). The work in [265] carefully designed this cell array to perform matrix-vector multiplications (MVM), where the matrix will be stored within the OPCM crossbar array, and individual elements of the vector are encoded per WDM signal batch in the horizontal waveguide. Each coupler has a splitting ratio associated with it, which is designed to ensure that the same fraction of signal from each wavelength reaches the GST OPCM, which is wavelength agnostic in the C-band of frequencies. So, in effect, each cell performs:

$$W_{cell} \times \left[\frac{\{A,\lambda_1\}}{n} + \frac{\{A,\lambda_2\}}{n} + \cdots + \frac{\{A,\lambda_n\}}{n}\right] = W_{cell} \times A \tag{80}$$

where, W_{cell} is the weight stored in the OPCM, A is the activation value imprinted onto the wavelength λ_i, n is the WDM degree (i.e., the number of wavelengths in the WDM batch) that corresponds to the number of cells per row. This operation makes it an excellent MVM engine, with low latency and energy-efficient operation. Additionally, this cell (Figure 104(b)) is compact and solves the interference and crosstalk issues that plague the COSMOS architecture [274] discussed earlier and would appear to be a good candidate for an OPCM-based PIM. However, the architecture is not column addressable, making it not a good choice for a memory architecture. To

consider this cell for a memory architecture and then a PIM architecture, column addressability to cells is essential.

To address these issues, the work in [51], COMET, designed a row and column addressable OPCM memory cell (Figure 104(c)), which is also isolated from other cells to avoid data corruption due to crosstalk. This memory cell makes use of GST for data storage, with two MRs acting as the access control, electro-optically. The MRs are electrically tunable using a PN junction and are hence active when they are in resonance (turned on). In this active state, they allow signals propagating through the vertical waveguide on the left to access the OPCM cell. The data is imprinted onto the signal and is passed to the readout waveguide on the right (Figure 104(c)). While the proposed cell is not as compact as the one suggested in [265], it offers more reliable data readouts, without crosstalk-induced errors. Further, the GST in the cell was designed to allow for improved energy efficiency in write operations. The subarray architecture also had provisions to ensure loss correction through intermittent semiconductor optical amplifier (SOA) arrays. There are several desirable characteristics that make COMET a suitable backbone for a PIM architecture, but there are also several challenges, as will be discussed in Section 12.2.

12.1.3 PHOTONIC COMPUTATION

The previous subsection discussed the characteristics required to realize an OPCM main memory. In this subsection we discuss principles of photonic computation, which are a precursor to realizing a PIM solution with OPCM memory.

Photonic computation can be performed through either coherent or noncoherent (aka incoherent) analog computation methods [37]. Coherent photonic computation utilizes the phase of light waves in a controlled manner, enabling the encoding and manipulation (e.g.,

multiplication) of data via interference patterns. This approach takes advantage of the coherent properties of light, such as phase coherence and superposition, to perform complex mathematical operations rapidly and with high precision. Computing architectures that leverage coherent computing often make use of Mach-Zehnder interferometers (MZIs) for data manipulation through constructive or destructive interference with a single wavelength.

Noncoherent photonic computation, on the other hand, does not rely on the phase information of light, conventionally [38]. Instead, it involves manipulation of the intensity or amplitude of light waves to perform computations, making it less sensitive to phase fluctuations and coherence issues that might affect coherent systems. Noncoherent approaches are simpler in terms of data encoding and more robust as they do not have as many noise sources to deal with. This makes them suitable for a wide range of applications that require optical signal processing, such as image processing and sensor data analysis, and fundamental arithmetic operations. Additionally, they allow performing arithmetic operations at a very large scale, through the usage of WDM, making noncoherent photonics an attractive option for MVM and general matrix multiply (GEMM) operations. To leverage WDM signals, the photonic device used in noncoherent computation systems must be wavelength sensitive, which makes wavelength selective MRs popular candidates for the fundamental devices in these architectures.

An MR is an on-chip optical resonator, which resonates when it encounters an optical wavelength that matches its resonant wavelength (λ_{MR}). Through tuning mechanisms, λ_{MR} can be altered, increasing losses to the encountered wavelength, thus enabling amplitude modulation, and hence forming the basis for noncoherent computation. There are two main tuning mechanisms used: thermo-optic tuning and electro-optic tuning. Both these mechanisms can change the effective refractive index (n_{eff}) of the bulk of the MR, thereby affecting ($\lambda_{MR}= 2\pi n_{eff} R$; R=MR

radius). Thermo-optic (TO) tuning achieves this by heating the MR through microheaters, and electro-optic (EO) tuning achieves the same through free carrier injection via a PN junction fabricated across the MR [37].

Several noncoherent computation architecture in prior work [49], [48], [44] rely on MR operation for high throughput, reliable, low energy ML inference acceleration, through the computation technique called broadcast and weight (B&W) [38]. Here, MRs are tuned to reflect a stationary matrix, and vectors are introduced either as amplitude-modulated wavelengths or via a subsequent array of tunable MRs downstream from the initial MR array's output. The interaction of light with the MRs modifies its amplitude to reflect a multiplication operation. Several of these light signals can be summed using a photodetector, achieving n multiply and accumulate (MAC) operations simultaneously. Here, n is the WDM degree of the signal and should correspond to the size of the MR array.

From the discussions in Sections 12.1.2, the OPCM memory cell in Figure 104(c) is a potential candidate to be part of noncoherent architectures that perform computation operations. The OPCM cells can represent the stationary matrix/vector element, while the incoming light signal or one of the access control MRs can represent the changing vector. At this point, performing a memory read operation through the OPCM cell will achieve a multiplication operation. However, to achieve effective large-scale noncoherent computation via PIM, several challenges must be addressed, as discussed in the next Section.

12.2 RE-ARCHITECTING OPCM MAIN MEMORY FOR PIM

In this section, we take a brief look at the COMET OPCM main memory architecture and why it cannot be used as is for effective noncoherent computation within a PIM solution.

The basic architectural component of the COMET main memory architecture is the OPCM memory cell depicted in Figure 104(c). This memory cell is tiled to form an array, where each cell can be isolated from each other, while access is enabled through a wavelength assigned per column of the memory cells in the array. Row access is provided by turning on the access control MRs through EO tuning, thereby allowing the light signals access to the OPCM cell. $N \times N$ of these cells can form a subarray and $S \times S$ of these subarrays form a memory bank. A collection of B memory banks constitutes the main memory.

There are four major challenges that must be overcome to adapt the COMET OPCM memory architecture for PIM:

- Accessing all the cells in the same row across subarrays and banks requires $B \times S \times N$ wavelengths, which would be too energy- and power-expensive for a main memory of any reasonable size. During data read/write operations, the light signals are given access only to the subarray in which the corresponding row resides. This is achieved through the usage of GST-based waveguide switching, rather than splitting the WDM signal into multiple subarrays unnecessarily. It should be noted that using optical splitters and couplers would essentially multiply the laser power needed, and this must be avoided.
- COMET was architected with specific power consumption constraints, and hence many architectural choices were made to enable a power consumption of under 10W for the main memory operation, as discussed earlier. This power constraint allows it to operate in a similar power point to electronic main memory architectures such as DDR5. However, from a PIM perspective, these choices pose a problem. Having limited access to subarrays, and hence OPCM cells, per read/ write operation severely limits the achievable

parallelization of computation operations. So, it is necessary to find a solution that enables multi-subarray access, without disrupting the optical main memory operation. Note that we cannot rely on increasing WDM degree or splitting signals from the source across multiple subarrays, as the power consumption this incurs will be many times higher than the 10 W constraint, reflecting the previous challenge.

- Optical signals can interact with each other in the readout waveguides. Increasing the WDM degree to avoid using splitters carries with it the risk of increased crosstalk and errors, especially when using OPCM cells at higher bit densities. So, careful orchestration of access and readout is necessary to achieve reliable and error-free computations.

- It is also important to consider the impact of bit density per cell on PIM operations. In COMET, a 4-bit per cell bit density was considered to ensure reliable memory operation. This limits possible neural network parameter sizes to 4-bit if there is a need to perform one-shot operations (e.g., multiplications) as discussed at the end of Section 12.1. Without careful architectural considerations, it will be impossible to handle higher parameter sizes for computation within COMET.

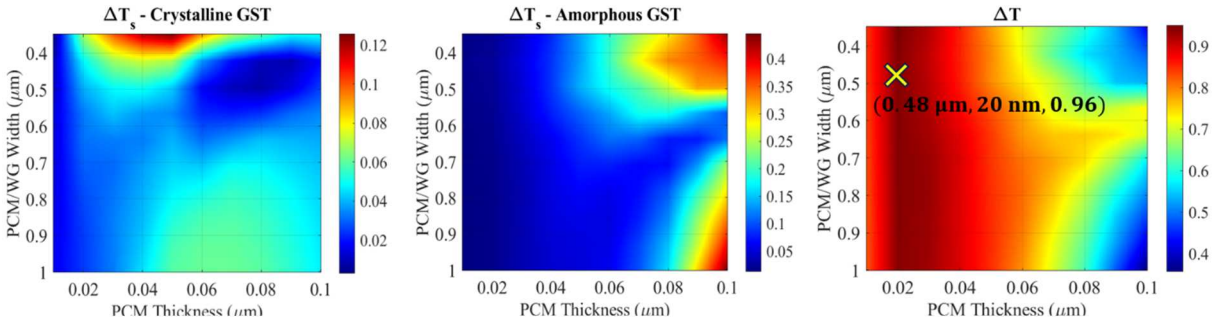
In summary, there are several challenges associated with enabling PIM within an OPCM main memory. In our proposed *OPIMA* architecture, described in the next section, we address all of these challenges via novel and significant alterations to an OPCM main memory architecture, to enable PIM within the memory platform, while still allowing it to retain its core functionality as a main memory solution.

12.3 *OPIMA* ARCHITECTURE

This section discusses the proposed *OPIMA* architecture and how it achieves PIM-based ML acceleration.

12.3.1 MAXIMIZING OPCM MEMORY CELL EFFICIENCY

The *OPIMA* architecture is a PIM architecture that significantly expands the capabilities of the COMET main memory architecture. COMET explored how effective refractive index (n_{eff}) and optical absorption (κ) can be optimized for maximum energy efficiency in OPCM cells. Based on this analysis, the authors had selected GST as the best-suited OPCM material for the C-band of frequencies.



(a) (b) (c)

Figure 105. Design-space exploration of GST-based OPCM memory cell, (a) Optical transmission changes due to scattering and back-reflections of the light (ΔT_s) in the crystalline state, (b) ΔT_s in the amorphous state, (c) Optical transmission contrast between amorphous and crystalline states (ΔT). Observe that for the chosen design point (highlighted with 'X'), the ΔT_s for both crystalline and amorphous states is less than 5% while the ΔT is at its maximum with 96%.

In this chapter, we consider more detailed factors influencing the behavior of OPCM-based memory cells, particularly the unwanted changes in the optical transmission of the cells because of the scattering and back-reflection of light when interacting with PCMs. The refractive index of the PCMs in crystalline and amorphous states is significantly higher than the refractive index of the waveguide material. Therefore, the propagating light can be scattered and reflected back and

forth in the waveguide when interacting with the PCM on top of the waveguide. Such a scattering effect leads to unwanted optical transmission changes at the output of the OPCM memory cell.

To tackle this limitation, we performed a design-space exploration using GST on top of silicon waveguide to select the most optimal geometry that offers minimal transmission change due to light scattering and maximum transmission contrast due to phase change. To capture the optimal design with minimized scattering of the light, we use the following model:

$$\Delta T_s = 1 - T_{out} - P_{abs}\,, \tag{81}$$

where ΔT_s is the optical transmission change due to light scattering and back-reflections, P_{abs} is the total fraction of the power that is absorbed in the PCM cell, and T_{out} is the output transmission of the cell (all in dB). This model is applicable to both amorphous and crystalline states of the cell. For ideal transmission control, T_{out} should follow an inverse trend with respect to P_{abs}, i.e.:

$$P_{abs} = (1 - T_{out}) \rightarrow \Delta T_s = 0. \tag{82}$$

In addition, the desired OPCM memory cell should offer 1) high optical transmission which originates from the low power absorption in the amorphous state, and 2) high absorption and hence low optical transmission in the crystalline state. Consequently, the optimum design point should offer minimized light scattering and back-reflections at both crystalline and amorphous states while leveraging the high controlled optical transmission contrast. Therefore, the ΔT_s and the total optical transmission contrast between amorphous and crystalline states ($\Delta T = T_a - T_c$) can be used as a figure-of-merit to find the optimal design for the GST-based OPCM memory cell. This optimal design should offer a low ΔT_s in the amorphous and crystalline state and a high optical transmission contrast (ΔT) between amorphous and crystalline states.

The results for a 2-□m long GST cell that we designed are reported in Figure 105. Observe that for the design point which offers the highest optical transmission contrast (ΔT) highlighted in Figure 105(c), the transmission change due to light scattering and back-reflections is always less than 5% in the crystalline state (Figure 105(a)) and the amorphous state (Figure 105(b)). In addition, GST offers a high controlled optical transmission contrast (~96%) for the optimal design point shown in Figure 105(c) which corresponds to a width of 0.48 μm and thickness of 20 nm. This higher contrast in transmission also allows us to program 16 levels of transmission per cell, allowing a bit density of 4 bits/cell.

The OPCM memory cell that we designed and optimized forms the building block of the *OPIMA* architecture that is designed for efficient data storage and access, as well as for performing in-situ multiplication operations. For the sake of maintaining row and column addressability, and hence main memory operation, we combine this OPCM memory cell with double MRs for optical access control.

12.3.2 OPCM MEMORY OPERATION

An overview of how *OPIMA* is designed to operate as a memory interfaced with an external general-purpose electronic CPU is shown in Figure 106. A controller unit that handles the electro-optical interfacing requirements must reside between the CPU and *OPIMA*, as depicted in the figure. This controller unit interprets memory commands from the host CPU, enabling main memory operation. It also supports data caching for read data to be sent to the CPU or data to be written to the OPCM memory. In the latter case, the data is encoded via optical signals derived from the laser source.

The isolated OPCM cells within *OPIMA* make read/write operations quite straightforward. For both operations, the row ID and subarray ID must be deciphered from the physical address. Once this has been done, laser signals are sent to the corresponding OPCM bank. The read process (Figure 107(b)) happens as the signal passes through the memory cell and is modulated by the OPCM's optical transmission. The read data is sent back to the E-O-E controller where it is demodulated using an MR array. Then this data can be translated to the electronic domain and passed on to the CPU. The write process (Figure 107(a)) requires much higher energy as it requires inducing partial phase transition in the OPCM memory cells. This necessitates more laser power to achieve the phase transition across multiple OPCM cells, based on the data to be written.

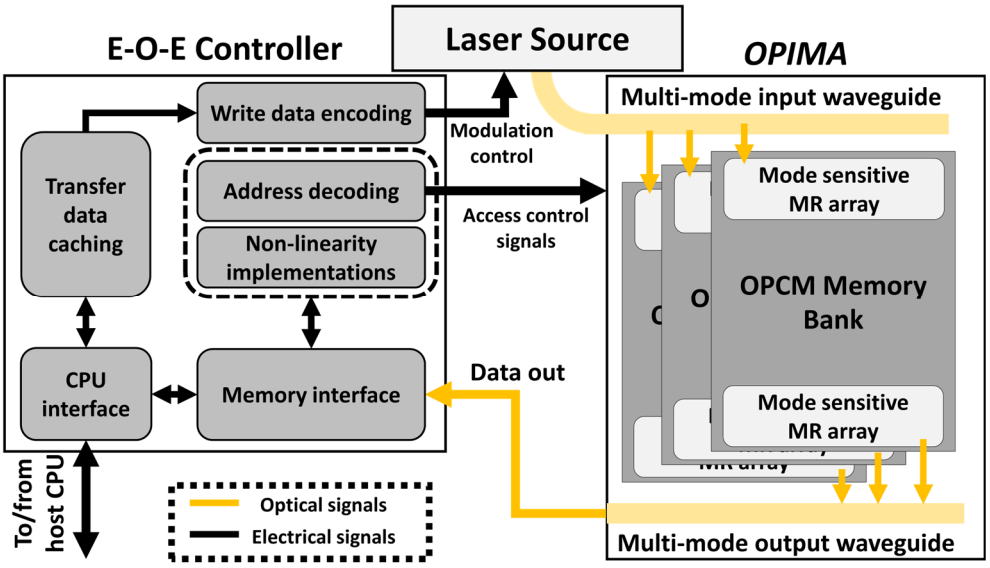


Figure 106. Architectural overview of *OPIMA*

During the read and write operations, data integrity is a critical concern, especially considering the loss tolerance in signal transmission. *OPIMA* incorporates semiconductor optical amplifiers (SOAs) within and outside the banks and subarrays to maintain signal quality. We employ row-

wise loss-aware signal amplification to counteract potential degradation. The banks and subarrays, once designed, have constant losses, facilitating this correction approach.

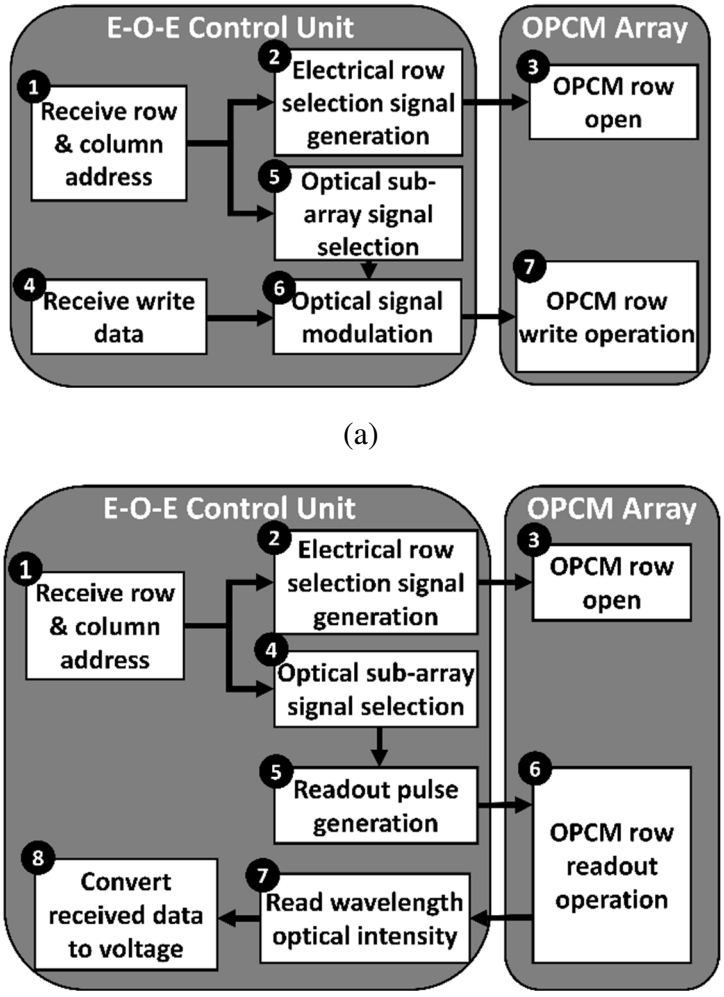


Figure 107. Memory (a) write and (b) read operation in *OPIMA*.

12.3.3 OPIMA PIM ARCHITECTURE

As discussed earlier, the optical transmission of an OPCM cell modulates the optical signal passing through it. If the access control MR is tuned to represent the second parameter, the successive modulations from the MR and the OPCM can achieve a multiplication operation. However, since we need all the MRs in a row to behave identically to facilitate row access, it is

better to tune the incoming laser signal to represent the second parameter. To achieve an accumulate operation, we must let two signals of the same wavelength, modulated to reflect products, interact with each other. To perform this step, we need to involve products from another subarray sharing the same readout waveguide. Within the readout waveguide, these signals interfering with each other generates the sums. This is desirable from a PIM perspective, but will lead to erroneous readouts from a main memory perspective. Hence, for achieving this goal and thus realizing the PIM operations for ML inference acceleration, we need several architectural changes to the main memory architecture, as discussed next.

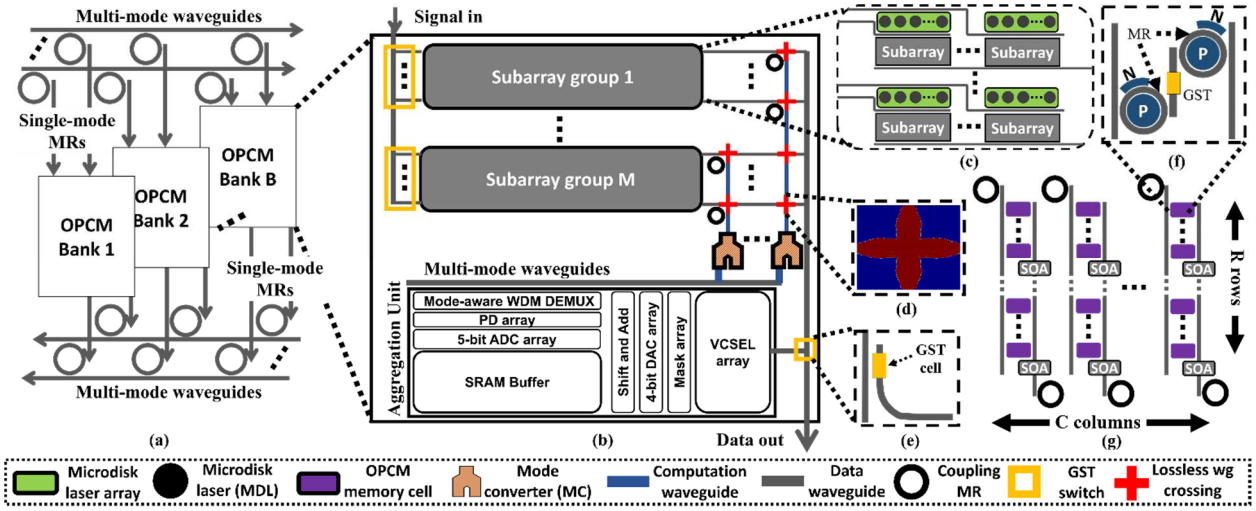


Figure 108. *OPIMA*'s PIM-specific architecture; (a) OPCM bank organization; (b) Subarray organization within the bank, showcasing grouping, aggregation unit, and computation specific waveguides, coupling MRs, and mode converters (MC); (c) Subarray group internals; each subarray is equipped with a microdisk laser (MDL) array for PIM operation independent of main memory operation; (d) low loss waveguide (wg) crossings designed using inverse design; (e) GST cells used for subarray access control during OPCM main memory operation; (f) OPCM memory cell with EO tuned MRs showcased; (g) OPCM memory array within subarrays, with $R \times C$ OPCM cells within it.

To realize high throughput and error-free PIM operation in *OPIMA*, we need to address four major challenges: (1) We need to leverage additional mechanisms to increase memory access and computation parallelism beyond those offered by WDM; (2) reads should be supported from a

selected subarray or a group of subarrays as needed, without interrupting the main memory operation; (3) When simultaneously read out, the data from computation outputs and main memory accesses must not interfere with each other in an undesirable manner; and (4) the architecture should support PIM operations between parameters (e.g., CNN weights and activations) of any size, irrespective of the specific bit density used in the OPCM cells.

12.3.3.1 IMPLEMENTING MDM FOR IMPROVED PARALLELISM

To address challenge (1), within *OPIMA*, we design the multi-bank OPCM memory organization to go beyond WDM and additionally use mode-division multiplexing (MDM) to enable parallel access across banks (Figure 108(a)). MDM involves exciting higher order modes in a waveguide, where each of the modes of a wavelength can then be used for supporting parallel data transfers and computations. Note that multiple wavelengths co-existing in the waveguide (WDM) provide further parallelism for data transfers and computations. Increasing the number of modes comes at a cost of increased width of the waveguide to allow the higher order modes to be excited and propagated, as well as increased crosstalk. Thus, determining the optimal number of modes (MDM degree) requires a careful trade-off analysis.

We inverse designed photonic mode convertors based on [303] to exploit the first four modes of TE polarization. Compared to conventional mode convertors based on tapered structures or thickness changes to induce the required index changed, the inverse designed mode convertors offer a compact footprint and minimal loss. Note that exciting more than four modes in the waveguide at the same time is physically challenging as it requires extremely wide waveguides that significantly increase memory area. In addition, higher order modes suffer from intermodal crosstalk due to the overlap of the modes [304], [305]. Based on our MDM propagation analyses, we decided to keep the MDM degree to four, which limits the number of banks in the architecture

to four. These MDM signals can be filtered by mode-sensitive MRs to their respective banks and be routed to their respective subarrays through GST switches, enabling parallel read/write operations across banks. However, there is a need to improve parallelism further to achieve higher PIM throughput. Additionally, while it is technically possible to perform a MAC operation by reading from two OPCM cells, this operation will be limited to 4-bit parameters under the configuration discussed here.

12.3.3.2 REDESIGNING BANKS FOR CONCURRENT PIM AND MEMORY ACCESS

A memory bank within the *OPIMA* architecture is composed of $R \times C$ OPCM cells (Figure 108(g)), offering a total capacity determined by the product of the number of cells and the bit density of each OPCM Multi-Level Cell (MLC). To enhance energy efficiency, banks are divided into subarrays. The *OPIMA* architecture employs electrically controlled GST-based waveguide switching to facilitate efficient subarray access (Figure 108(e)), markedly reducing the laser power requirements. The GST switch introduces minimal losses and is pivotal for the energy-efficient operation of the system. We need to make changes to this organizational structure and provide additional access mechanisms to address challenge (2).

Data within OPCMs cannot be sensed in the same manner as charge-based storage in DRAM. Accessing data in OPCM cells necessitates external laser signals, which must overcome several losses in propagation, to be rerouted to the subarrays within which the OPCM cell resides. But this will lead to high power consumption to overcome the losses and the signals being split into several destinations. To circumvent this, we propose the addition of local laser sources to subarrays, which can be triggered as needed for reads. Fortunately, unlike OPCM write operations, OPCM read operations are not energy intensive [51] and hence we can employ low-power lasers.

For *OPIMA* we opted for low-power microdisk laser (MDL) arrays (Figure 108(c)), which can be integrated with every subarray. Each subarray uses *C* MDLs in its subarray, reflecting the column number per subarray. The laser output from the MDL array can be coupled onto the signal input waveguide of the corresponding subarray, using directional couplers. Using the MDL arrays, we can access any row within a subarray, without the involvement of the external laser source which drives the main memory operation. Additionally, since the MDL arrays are independent of each other, multiple of them can be activated simultaneously to read from multiple subarrays without having to reroute or incur additional losses.

Moreover, to ensure that we can read for PIM while main memory operations happen in parallel, the subarrays are divided into several groups (Figure 108(b)). One row of subarrays per group can be employed for PIM at a time, while the rest of the subarrays can be used for main memory read/write operations. This ensures significant parallelism in MAC operations that can be executed simultaneously per bank, offering simultaneous solutions to challenges (1) and (2).

12.3.3.3 REDUCING OUTPUT INTERFERENCE

Now that we have several MAC operations being supported simultaneously, we must ensure that their results can be aggregated without interfering with each other or the main memory readout operations, to address challenge (3). It should be noted that the subarrays make use of WDM signals which can interfere with each other constructively or destructively.

To avoid computation signals interfering with memory read operations, we employ a series of computation-specific waveguides. Computed data is rerouted to the computation waveguides rather than the data-out waveguide using coupling MRs which can be activated alongside the MDL array (Figure 108(c)). The computation waveguide is used to move the data to the aggregation unit

in the bank. To prevent losses and the computed signal from interfering with orthogonally traveling data signals, all the waveguide crossings in the computation waveguide have been carefully designed to be as leakage-free as possible (Figure 108(d)).

To achieve the optimized waveguide crossing design, we used a photonic inverse design technique to minimize the loss and crosstalk of the waveguide crossings. The Lumerical FDTD solver [216] with the LumOpt [306] inverse design library was used to perform the geometry optimization of the waveguide crossings. The optimized geometry of the waveguide crossing is shown in Figure 109. Note that the transmission of the fundamental TE mode was used as a figure-of-merit in our inverse design optimization of waveguide crossing. We can observe from the figure that the inverse-designed waveguide crossing offers the maximum transmission at the C-band with less than 0.001% of the input optical signal being lost due to optical insertion loss. Note that the optimized waveguide crossing offers minimal -40 dB of the crosstalk in the C-band.

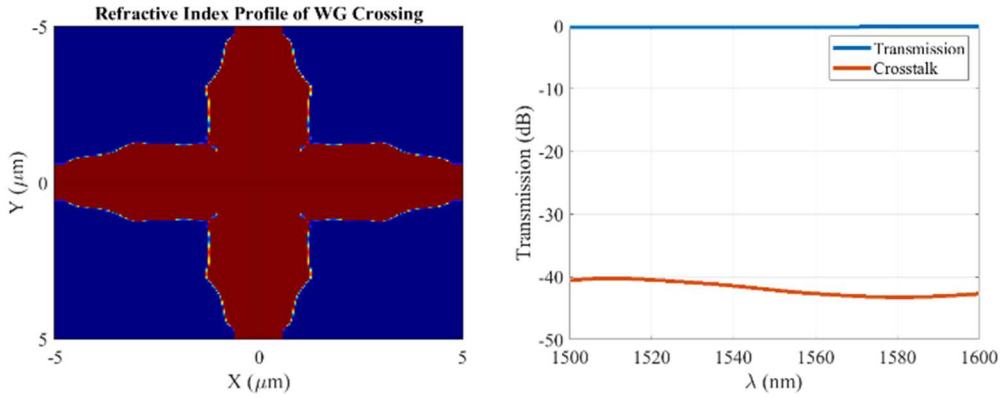


Figure 109. Low-loss waveguide crossing designed with inverse design methodology (left) and its loss profile for C-band (right).

As the data reaches the aggregation unit, they have to be merged. Here again, interference between signals can be an issue. As discussed earlier in this subsection, we can make use of up to four modes without significant crosstalk between the signals. We can reuse the orthogonality of

modes here again. Each subarray group can be assigned a mode using a mode converter (MC), before it merges with the waveguide carrying the signals to the aggregation unit's demultiplexer (demux). These changes to the architecture solve challenge (3).

12.3.3.4 ADDRESSING BIT SIZE MISMATCHES

OPCM cells within the photonic memory can be designed to have different bit densities, e.g., 1 bit/cell, 2 bit/cell, 4 bit/cell, etc. However, the parameters in an ML model like a CNN can be 32 bits in size without quantization. They can also be quantized to lower bitwidths such as 16 bits, 8 bits, or 4 bits to reduce storage requirements and also to reduce computation latency and energy. In scenarios where there is a mismatch between OPCM cell bit density and the CNN parameter size (e.g., 4 bits/cell bit density with 8 bit CNN parameters), the one-shot multiplication operation achieved by reading the OPCM cell, as discussed earlier, is not feasible.

To support different bitwidth scenarios and tackle challenge (4), we make use of a time division multiplexing (TDM) based approach. For higher bit densities per cell than 4-bits (i.e., a nibble), each nibble will have to interact with every nibble of the other parameter. This can be achieved without significant loss in throughput because of solutions for challenges (1)-(3) which offer high parallelism in MAC operations, while the signals can stay disentangled from each other. However, we still have to perform shift-and-add operations to obtain the true results for these operations [46]. These necessary operations are facilitated within the aggregation unit (Figure 108(b)). This results in an overall drop in throughput, but facilitates flexibility in operation, unconstrained by the OPCM MLC bit-density.

The aggregation unit is essential to address challenge (4), but it also provides some additional benefits. The photodetector (PD) based conversion to the electrical domain acts as a noise filtering

mechanism. The wavelength-specific PDs offer disentanglement from crosstalk between wavelengths, improving SNR before the longer transmission to the E-O-E control unit. Additionally, the parameters can be stored within the SRAM cache within the aggregation unit, for additional accumulation operations if needed. We also consider 5-bit ADCs so that the data can be translated to the electrical domain with any carries from the operations. Finally, the readout signals for the MAC operations which were generated using low-power MDLs will be regenerated through DACs and vertical cavity surface emission lasers (VCSELs) for better fidelity before they reach the E-O-E controller which handles further aggregation and applies non-linear activation functions (see Figure 106) for ML inference operations.

12.3.4 CNN MAPPING AND INFERENCE IN OPIMA

The architectural design choices discussed in the previous subsection allow the *OPIMA* architecture to realize high power consumption efficiency and high integrity large-scale parallel MAC operations and main memory accesses in the optical domain. From a CNN inference perspective, this offers two-fold benefits. Firstly, MAC operations are fundamental operations in CNNs and *OPIMA* is capable of performing them with high degrees of parallelism. Secondly, CNNs in general require significant storage and data movement between layers, but this can be significantly reduced as the processing occurs within the memory where model parameters and activation feature maps are stored.

To leverage the parallelism offered by the PIM substrate in *OPIMA* for CNN inference, we need to efficiently map CNNs onto the OPCM arrays. For CNNs, this involves mapping the parameters from both convolutional layers and fully connected layers. Operations for both of these types of layers can be mapped into MVM operations. For convolutional layers, we adopt an input stationary dataflow approach, where the input data can stay in its native storage location while we

drive the smaller weight matrices (decomposed as vectors) through them. Because of the large row sizes within the subarrays, we will be able to drive several kernels simultaneously. The feature map must be divided across subarrays, so that we can access subsequent rows of the map from neighboring subarrays. The kernels rows which must operate on the feature map can be encoded into laser signals through MDL tuning and be introduced into the subarrays. Additionally, we can achieve several parallel MAC operations through in-waveguide interference of WDM signals, from multiple subarrays within the same subarray group.

Let us consider a simple example with a 2×2 kernel, a feature map (F) with a row size of 4 elements, and MDL array generating wavelengths $\{\lambda_1, \lambda_2, \ldots, \lambda_C\}$ (C=number of columns per subarray). The kernel can be broken down into two vectors and mapped to MDL wavelengths: $k_1 = \{k_{00}, k_{01}\} \rightarrow \{\lambda_1, \lambda_2\}$ and $k_2 = \{k_{10}, k_{11}\} \rightarrow \{\lambda_1, \lambda_2\}$. Similarly the rows in F can be broken down into vectors and mapped to subarrays: $\{f_{00}, f_{01}, f_{02}, f_{03}\} \rightarrow Subarray_1$ and $\{f_{10}, f_{11}, f_{12}, f_{13}\} \rightarrow Subarray_2$. Both subarrays must be within the same subarray group to facilitate the MAC operation. If we now enable access to the rows containing these vectors and simultaneously send the k_1 and k_2 signals from the MDLs through the subarrays, we shall obtain the following in the common readout waveguide $\{(k_{00} \times f_{00}, k_{10} \times f_{10}), \lambda_1\}, \{(k_{01} \times f_{01}, k_{11} \times f_{11}), \lambda_2\}$.

Because signals of the same λ_i interfere with each other, this in turn generates: $(k_{00} \times f_{00} + k_{10} \times f_{10})$, $(k_{01} \times f_{01} + k_{11} \times f_{11})$, which is one addition away from generating the first element of an output feature map. This addition can be performed at the aggregation unit. The kernel can be moved across the MDL array to reflect the stride operation and further outputs can be obtained. Additionally, multiple kernels can be deployed simultaneously over F, across different

wavelengths, reducing overall processing time requirement. This mapping process scales easily with kernel sizes as well, as long as the kernel sizes do not exceed the subarray row size.

For fully connected layers we opt for a weight-stationary approach. In both cases, the stationary matrix has to be distributed across subarrays to ensure parallelism in operations. Once this mapping process is done, *OPIMA*'s PIM-specific architecture (Figure 108), as described in this section can be utilized effectively to achieve inference operation.

12.4 EXPERIMENTS

In this section, we discuss the evaluation of the performance of *OPIMA* for PIM-based CNN inference acceleration. *OPIMA* adopts a main memory configuration of 4 banks, 64× 64 subarrays per bank, with 256× 512 OPCM elements and 256 MDLs per subarray. For evaluating *OPIMA* we rely on a Python-based simulator, which makes use of the loss and energy parameters summarized in Table 30.

We compare *OPIMA* against several electronic and optical acceleration platforms along with the current state-of-the-art in photonic PIM. For our photonic accelerator systems, we consider the work in [302], named PhPIM in our comparison studies, which proposed a PIM adjacent system, and CrossLight [43], a photonic CNN accelerator. CrossLight and PhPIM are modeled using the parameters in Table 30, and considering 8GB DDR5 DRAM, with 4800 megatransfers per second (MTS) data transfer rate as its main memory.

We also consider Nvidia P100 GPU (referred to as NP100 in results), AMD EPYC 7742 CPU (referred to as E7742 in results), and Nvidia Jetson ORIN (a low-power embedded GPU for edge AI applications; referred to as ORIN in results), as our electronic platform comparison points.

Table 30 Optical loss and power parameters considered for *OPIMA*

Loss parameters	Values
Directional coupler loss	0.02 dB [307]
MR drop loss	0.5 dB [286]
MR through loss	0.02 dB [124]
Propagation loss	0.1 dB/cm [287]
Bending loss	0.01 dB/90^{o} [151]
EO tuned MR drop loss	1.6 dB [280]
EO tuned MR through loss	0.33 dB [280]
SOA gain	20 dB

Energy parameters	Values
OPCM read	5 pJ [51]
OPCM write	250 pJ [51]
EPCM write	860 nJ [308]
DRAM access	20 pJ/bit [309]
ADC	24.4 fJ/step [310]
DAC	2.0 pJ/bit [311]

12.4.1 SUBARRAY GROUPING

The first experiment explores the *OPIMA* design space to determine the number of subarray groups, which in turn will determine the number of operations that can be performed per cycle, in *OPIMA*. This increase in parallelism trades off with the power consumption of the architecture. As the number of groups increases, the complexity of the interface required at the aggregation unit also increases, along with the laser power requirement to perform the operations. Simultaneously, we would like the maximum number of subarray rows to be accessible for main memory operations.

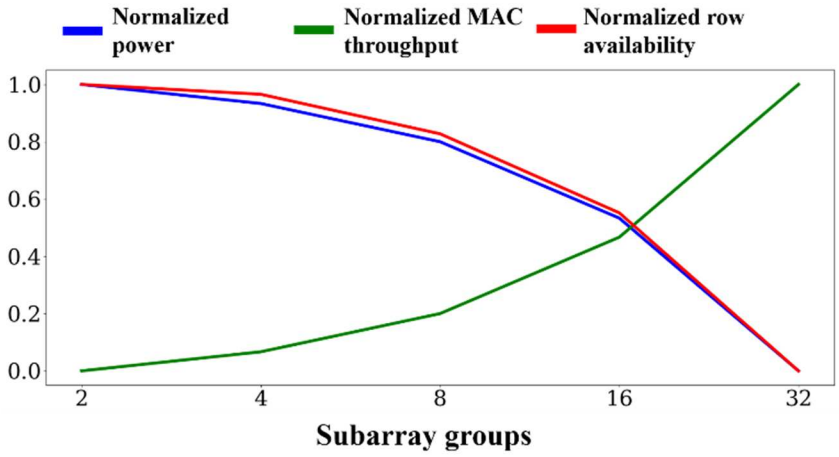


Figure 110. Subarray group selection for *OPIMA* architecture.

The *OPIMA* memory organization has 64 rows of subarrays per bank as mentioned earlier, which must be grouped as per the criteria discussed above. While considering the groups, we would like to avoid the extremes i.e., the case with a single group or the case with each subarray row belonging to an individual group, resulting in 64 groups. A single group severely limits parallelism and 64 groups imply that all 64 rows will be engaged in PIM operations, essentially preventing any main memory read/write operations.

Figure 110 shows the normalized power, MAC throughput, and rows available for main memory operation, with changing number of subarray groups (x axis). It can be observed that a configuration with 16 groups strikes a balance between achievable compute parallelism with reasonable power consumption and sufficient memory access without starvation. Additionally, 16 subarray groups enables the maximum throughput efficiency (MAC/Watt) from *OPIMA*.

Table 31 Various models considered for *opima* evaluation and their accuracy across quantization levels for classifying the specified datasets.

Model	Dataset	Accuracy (fp32)	Accuracy (int8)	Accuracy (int4)	Parameter count
Resnet18	CIFAR100	75.3%	74.2%	72.6%	11584865 (11.6 M)
InceptionV2	SVHN	81.5%	80.8%	75.9%	2661960 (2.6 M)
MobileNet	CIFAR10	88.2%	87.5%	83.5%	4209088 (4.2 M)
SqueezeNet	STL-10	92.5%	90.3%	86.5%	1159848 (1.1 M)

Our earlier analysis on mode conversion pointed to the fact that we can only have a maximum of four modes in our waveguide at the aggregation unit. Since we have to rely on four modes only, to meet the demand of 16 groups, the modes can be reused. For enabling mode reuse we use the same mode converter designs along the computation waveguides (Figure 108(b)). Additionally, to prevent the same modes from interacting with each other, each four modes is assigned a separate multimode waveguide for transferring to the demux unit within the aggregation unit.

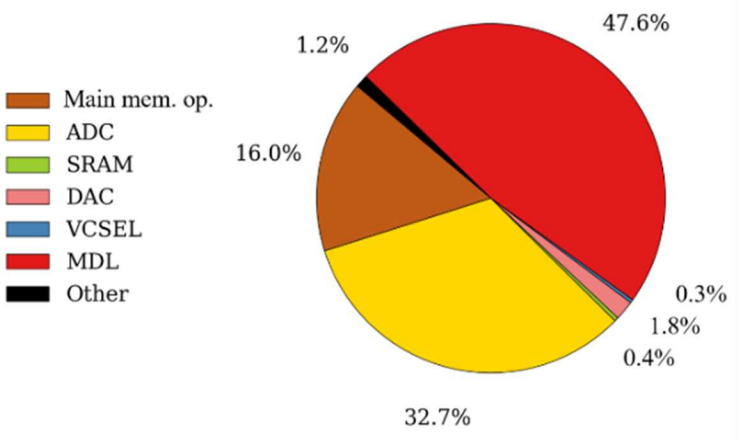


Figure 111. Power breakdown for *OPIMA* architecture.

12.4.2 OPIMA POWER BREAKDOWN

The power consumption breakdown of the resulting version *OPIMA* is shown in Figure 111. From this plot we can observe that the maximum power consumption is contributed by the MDL array and the electrical-optical interface, leading to a maximum power consumption of 55.9 W, for both main memory and PIM operations running simultaneously.

12.4.3 CNN WORKLOAD ACCURACY AND LATENCY ANALYSES

For workloads we considered four CNN models: ResNet18, InceptionV2, MobileNet, and SqueezeNet. The inference is performed for image classification of datasets, details of which are provided in Table 31. We have considered 4-bit integer quantization using TensorRT, as this is the baseline MLC capacity. As the table shows this level of quantization results in at most 6% loss in accuracy, in the considered models. But this accuracy drop is model architecture-dependent, as can be seen in Table 31. To showcase *OPIMA's* flexibility in handling parameter sizes, we have also considered 8-bit variants of the same models (Table 31).

Before we go into further comparisons, we first analyze the performance of *OPIMA* using both the 4-bit and 8-bit quantized variants of the CNN models. A breakdown of *OPIMA*'s latency in ms, as it processes these models, is provided in Figure 112. Processing latency is the total time for processing the necessary MAC operations and the aggregation unit operation, i.e. all in-memory processing operations. The writeback latency refers to the latency incurred while applying the non-linearities and writing back the final results, i.e. output feature maps, back into *OPIMA*'s main memory architecture.

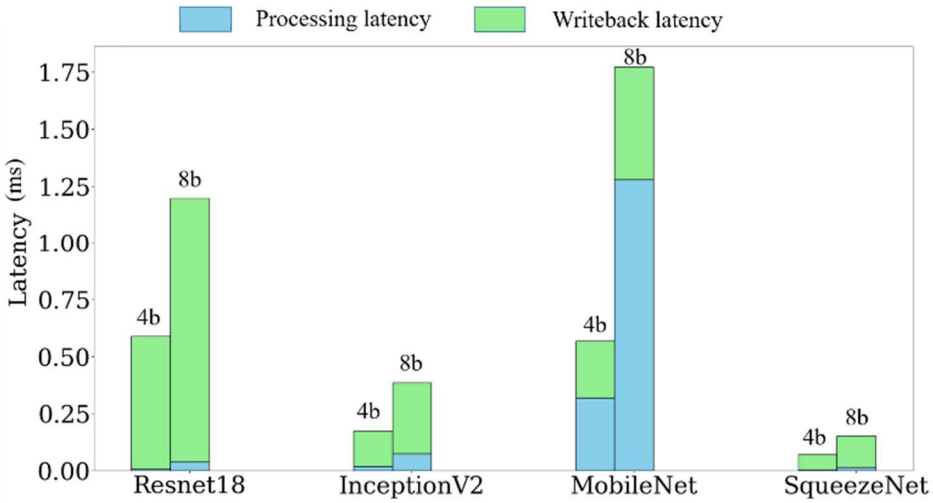


Figure 112. Latency breakdown for *OPIMA*'s 4-bit (4b) and 8-bit (8b) variants across the models from Table 31.

It can be observed that writeback is a significant contributor to latency in *OPIMA*. The PIM operations can leverage data within the memory and the high processing parallelism, leading to remarkably low processing times. However, the latency for the OPCM write operations needed to make the output feature maps available within the memory for further processing far outweigh the latency savings from the PIM operations. So, even though *OPIMA* can handle a variety of parameter sizes, given the OPCM write latencies, it is prudent to rely on 4-bit quantized models, while suffering some loss in accuracy, if throughput is significantly more important.

It can also be observed that *OPIMA* does not perform as one would expect for the far smaller InceptionV2 and MobileNet models when compared to ResNet18. Both of these models have higher processing latencies, with MobileNet having significantly higher processing latency than ResNet18. This is attributed to the 1×1 kernels in these models, which pose problems for the WDM-based MAC parallelization within *OPIMA*. Since the results from these operations do not have any further accumulation to be performed on them, they prevent the totality of the subarray row from being used. If more operations are performed, they will interfere with the results from

the 1× 1 kernels, leading to erroneous results. So, when these are encountered, *OPIMA* loses a significant portion of its parallel processing capabilities, especially when they are sequential in the CNN execution graph, like in the case of InceptionV2. MobileNet, though a larger model, offers higher parallelization opportunities, and hence performs at a similar latency, despite being ~4× the size of InceptionV2.

Similarly, writeback is a significant contributor to overall latency as discussed earlier. However, this is proportional to the sizes of the output feature maps generated by the model and not the computational complexity of the model. This is the reason for MobileNet having lower writeback latency than processing latency, in comparison, and why InceptionV2 has an overall lower latency than ResNet18.

To further characterize the latency benefits of *OPIMA*, we compare it against the latency for the other photonic computing architectures we have considered, as shown in Figure 113. The OPCM-based architectures (*OPIMA*, PhPIM) have better performance than CrossLight, because of the higher parallelism achievable in these architectures. PhPIM leverages the photonic tensor core operation from [265], along with an external DRAM acting as the actual main memory. PhPIM has opted for the faster yet energy-intensive electrical PCM programming mechanism, but the tensor core operation is still in the optical domain. The reprogramming, or writeback as we call it for an OPCM PIM, is significantly faster for PhPIM. However, *OPIMA* leverages much higher parallelism inherent to a main memory, and available to a PIM architecture, enabling faster processing times. Additionally, *OPIMA* does not have to access an external DRAM to access data needed for processing hence it does not have any external data movement latencies associated with its operation.

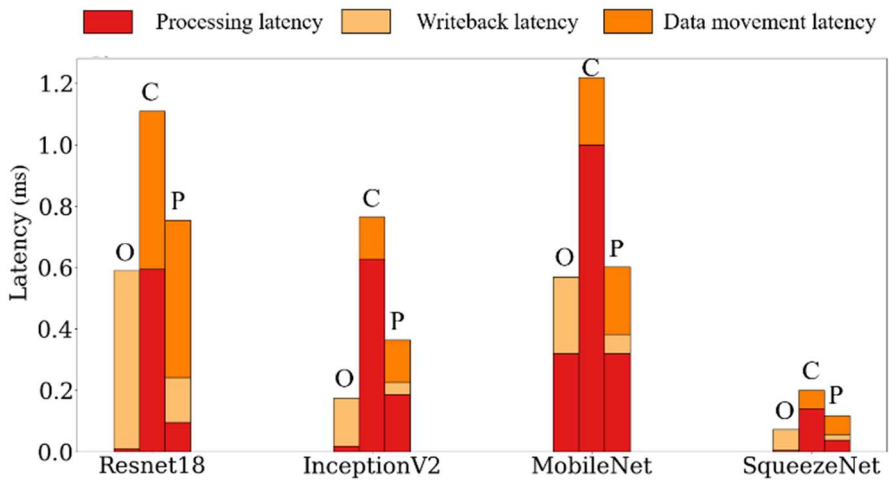


Figure 113. Latency breakdown of CNN model inference across photonic architectures *OPIMA* (O), CrossLight (C), and PhPIM (P), for model-dataset pairs from Table 31.

12.4.4 COMPARISON STUDIES

In this section, we compare *OPIMA* against the various photonic and electronic acceleration platforms in terms of energy per bit (EPB) and throughput efficiency (FPS/W; FPS=frames per second).

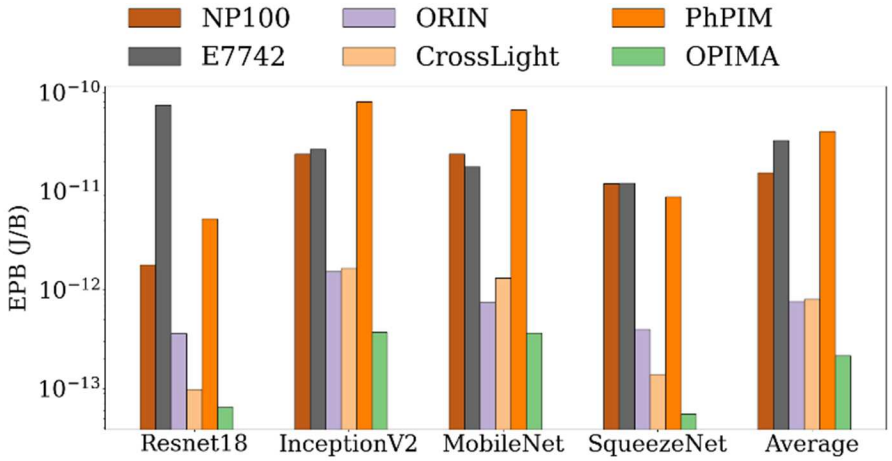


Figure 114. EPB comparison across architectures.

On average *OPIMA* achieves 71.2×, 151.2×, 3.5×, 3.7×, and 186× better performance in terms of EPB over NP100, E7742, ORIN, CrossLight, and PhPIM respectively (Figure 114). It

should be noted that P100 can outperform *OPIMA* in terms of raw throughput, especially in the case of InceptionV2 and MobileNet, where the GPU threads are not constrained by the interference limitations of our WDM-based parallelization of operations.

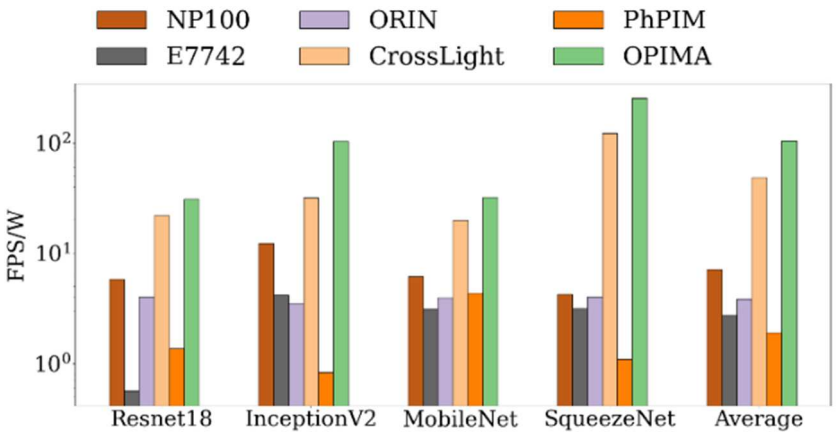


Figure 115. FPS/W comparison across architectures.

But *OPIMA* consumes significantly lower power, which also leads to overall better throughput efficiency. In terms of FPS/W, *OPIMA* achieves 14.7× , 38.2× , 27.2× , 2.1× , and 55.3× better performance over NP100, E7742, ORIN, CrossLight, and PhPIM respectively (Figure 115).

It can also be noted that though *OPIMA* and PhPIM had comparable latencies (Figure 113), *OPIMA* is able to outperform PhPIM in these metrics. This is because of the energy-intensive EPCM write processes that accompany PhPIM operation (nJ), as opposed to *OPIMA*'s OPCM reprogramming process (pJ).

12.5 CONCLUSIONS

In this chapter, we presented *OPIMA*, a high throughput, low latency, highly energy efficient OPCM PIM architecture. OPIMA showcases how an OPCM main memory architecture can be rearchitected to achieve photonic PIM. Through device-level design to enhance efficiency and various architectural innovations, *OPIMA* compares remarkably against electronic and photonic

ML acceleration platforms. On average *OPIMA* outperforms the considered architectures by 83.1× in terms of EPB and 27.5× in terms of FPS/W. In particular, it outperforms the state-of-the-art photonic PIM architecture PhPIM by 186× and 55.3× in these metrics, while achieving lower average latency, across several CNN models. *OPIMA* also opens the door for possible system-level integration of photonic PIM with dedicated photonic accelerators such as those described in [43]- [50]. Such a system can benefit from both the higher bandwidth that *OPIMA*'s main memory can provide along with computation support through PIM.

CHAPTER 13: CONCLUSION AND FUTURE WORK SUGGESTIONS

13.1. RESEARCH CONCLUSIONS

We addressed several challenges faced by noncoherent photonic DNN accelerators through cross-layer design approaches, in this thesis. Our proposed codesign framework employs layer-specific solutions and cross-layer solutions that combine enhancements at the software-level, architecture-level, circuit-level, and device-level. Our framework utilizes (i) hardware and device aware quantization and compression strategies at the software level; (ii) neural network specific hardware pipelines at the architecture level, along with NoC and memory enhancements to bring us closer to a fully photonic system; (iii) device and circuit level enhancements to reduce crosstalk, thermal noise and variations, and address fabrication process variation. The works that employed this framework validate this cross-layer design approach, through their experimental results, showcasing the framework's potential as a strategy to ensure high throughput, energy efficient, and reliable noncoherent photonic DNN accelerator designs.

Our first contribution was *ARXON*, a loss-aware approximation framework for data communicated over PNoC architectures, which improved on its predecessor LORAX. ARXON explored multi-level signaling, MR tuning, and crosstalk mitigation strategies as avenues to save energy along with the loss-aware laser tuning startegy. Our results indicated *ARXON* can reduce laser-power consumption by up to 57.2% over a baseline PNoC architecture like Clos and SwiftNoC. Our framework also shows up to 56.4% lower laser power and up to 23.8% better energy-efficiency compared to the best-known prior work on approximating communication in PNoCs, at the time.

Our next contribution is the photonic CNN accelerator architecture *CrossLight*, which features CNN specific VDU design, combined with architecture, circuit, and device level optimizations to combat laser power, reliability, and tuning challenges. *CrossLight* experimentally demonstrated 9.5× lower energy-per-bit and 15.9× higher performance-per-watt compared to state-of-the-art photonic DNN accelerators. *CrossLight* also shows improvements in these metrics over several CPU, GPU, and custom electronic accelerator platforms considered in our analyses.

By driving accelerator design from the software level, we arrived at high throughput and energy efficiency exhibited by our *ROBIN* architecture. *ROBIN* is an optical-domain BNN accelerator which utilizes device-level, circuit-level and architecture-level optimizations to save on energy and area while improving overall throughput. Through our optimization efforts, we identified two variants of *ROBIN*: *ROBIN-EO*, which is optimized for energy and area efficiency, and *ROBIN-PO*, which exhibits higher throughput performance, at the expense of comparatively higher power consumption. *ROBIN-EO* exhibits EPB values ~4× lower than electronic BNN accelerators and ~933× lower than the photonic BNN accelerator considered, while *ROBIN-PO* shows ~3x and ~25x better FPS than the electronic and photonic BNN accelerators respectively.

Similarly, tackling unstructured sparsity in CNNs using our crosslayer design framework yielded *SONIC* architecture. *SONIC* integrated several architecture and software level optimizations and exhibits up to 5.8× better power efficiency, and 8.4× lower EPB than state-of-the-art sparse electronic neural network accelerators; and up to 13.8× better power efficiency and 27.6× lower EPB than state-of-the-art dense photonic neural network accelerators.

To efficiently accelerate heterogeneously quantized CNNs, we proposed *HQNNA*, which used WDM and TDM simultaneously to achieve this goal. By pinpointing the ideal quantization profiles for CNNs and making tailored optimizations for the hardware, HQNNA managed to secure

enhanced performance metrics, specifically in the realms of energy and throughput efficiency. It achieved improvements reaching up to 73.8× better EPB and 159.5× better throughput-energy efficiency compared to traditional photonic CNN accelerators.

RecLight which was designed for being capable of accelerating several RNN variants, exhibited EPB improvements that range from 37× to 1730× when compared with six state-of-the-art electronic RNN accelerators. *RecLight* also demonstrated up to 2631.6× better throughput than these electronic RNN accelerators.

The GNN accelerator designed through our framework, *GHOST*, exhibited throughput improvements of at least 10.2× and energy-efficiency improvements of at least 3.8×, when compared against to nine computing platforms and state-of-the-art GNN accelerators.

The transformer accelerator *TRON*, exhibited throughput improvements of at least 14× and energy-efficiency improvements of at least 8× when compared to eight different processing platforms and state-of-the-art transformer accelerators. *TRON*'s flexible and energy efficienct architecture was also found to be ideal for deployment in different scenarios, including edge computing environments.

We also explored 2.5D chiplet platform-based photonic DNN accelerators, where both communication on the interposer and computation on the chiplets were performed using photonic devices. The 2.5D platform outperformed its monolithic counterpart with ~7× better throughput and ~6× better EPB values, owing to the high-bandwidth photonic communication across chiplets.

Seeing how we were able to tackle computation and communication challenges in DNN acceleration using our framework, we applied it on photonic main memory to yield the *COMET* main memory architecture. *COMET* followed the crosslayer design principle of our framework,

by designing a cohesive main memory system from PCM storage cell design to access controlled PCM memory cell and architecture enhancements for laser power management, loss and crosstalk management, and high bandwidth data transfer from memory. Thanks to these enhancements, *COMET* operates at a mere 26% of the power usage relative to the most efficient previously documented OPCM-based main memory architecture designs. This reduction in power consumption, coupled with the rapid operation of GST cells, allows *COMET* to deliver a bandwidth that is 7.1× superior, an EPB that is 15.5× lower, and latencies that are 3× shorter than the most advanced OPCM-based main memory architectures currently known.

Using *COMET* as a backbone we proposed the photonic PIM architecture *OPIMA*. *OPIMA* employs several PIM-specific architectural enhancements, including: (i) local microdisk arrays to facilitate low-energy PIM operations without stressing the main memory laser source; (ii) computation-specific waveguides separating PIM outputs from main memory read outputs for reliability in PIM and memory operations; (iii) memory operation aware grouping of subarrays to ensure high throughput PIM operation along with main memory operation. On average, *OPIMA* surpasses the architectures we compared it against with an 83.1× improvement in EPB and a 27.5× enhancement in frames per second per watt (FPS/W). Specifically, it outperformed the best prior known photonic Processing-in-Memory (PIM) architecture, which we named PhPIM, by 186× in EPB and 55.3× in FPS/W, all the while maintaining lower average latency.

With these contributions, we have the main components for a photonic ML acceleration system (accelerator, communication network, and main memory) which only requires minimal control interface from the electronic domain. These contributions and the the cross layer design framework we proposed may help enable such systems in the future.

13.2. SUGGESTION FOR FUTURE WORK

As the DNN architecture landscape continues to grow in scale and complexity, there are several directions our future works may take. Some of the key directions are as follows:

- *Hyperdimensional Computing using noncoherent photonics:* Hyperdimensional computing (HDC) is an emerging computational paradigm inspired by the brain's ability to process information through high-dimensional, dynamic representations. Unlike traditional computing models that rely on binary logic and linear data processing, HDC operates in spaces of thousands of dimensions, using vectors to represent data and perform computations. This approach enables HDC to handle complex patterns, recognize similarities, and efficiently manage large-scale, noisy datasets. Its applications span areas such as natural language processing, bioinformatics, and robotics, offering a robust and energy-efficient alternative to conventional computing methods. HDC's capacity for parallel processing and noise tolerance makes it particularly appealing for developing advanced artificial intelligence systems, and makes it an ideal candidate to tackle using our framework.
- *Data representation techniques in noncoherent photonics:* The works discussed in this thesis relied on integer quantization for running DNNs on noncoherent photonic systems. Though high accuracy can be obtained from integer quantization, converting data and weights associated with a model to integer from fp32, and then obtaining similar accuracy to the original model is possible, but is a difficult task. Having the ability to use floating point or fixed point data directly in the photonic domain will make the architectures yielded from the domain attractive to a wider audience. However, this datatype representation capability is limited by the broadcast and weight

approach which relies solely on amplitude modulation. More research on how to represent and operate on other datatypes effectively in photonic analog domain is a necessity.

- *Cohesive photonic acceleration systems:* As mentioned at the end of Section 13.1, we have the components to enable a photonic system which will only require minimal control interaction with the electronic domain. However, enabling such a system requires careful architecture design to ensure reliability, energy and power efficiency, and high throughput. The accelerators discussed in this thesis still rely on an electronic-photonic interface for data movement and aggregating results. A cohesive photonic system can do away with the interface circuitry and rely on photonic memory, data movement, and computation, bringing its throughput capabilities closer to the theoretical maximum of photodetection rate.
- *System and software integration of photonic systems to digital systems:* As mentioned, even though photonics can be used effectively for application-specific processing, it still needs a general purpose processor to interact with surrounding systems and the user. The analog computation technique lacks the flexibility to be used for general purpose computing, currently, hence the photonic system will have to inevitably interact with the rest of the systems in, for example, a data center through a digital CPU. Considering the photonic system as a subsystem like this opens up avenues for several research directions: (i) developing methods to seamlessly integrate photonics with digital electronics, ensuring they operate synchronously and efficiently; (ii) interface hardware and protocol design to enable analog-digital communication and data transfer with minimal loss of fidelity and latency; (iii) System architecture

optimization considering both photonic and digital components; (iv) investigating driver and middleware development for interfacing with real-world programs like the OS of the host and higher-level applications (E.g. PyTorch).

- *Photonic accelerator hardware security:* Security in ML accelerators is crucial primarily to prevent malicious exploits such as model theft, adversarial attacks, and data leakage, which can have dire consequences, including financial loss, privacy violations, and safety hazards. ML accelerators are often deployed in environments where they process highly sensitive information, making them attractive targets for cyber-attacks. All of this is true for photonic accelerators as well, making this a crucial aspect of architectural design to be considered in future works.
- *Photonic neuromorphic computing through PIM:* Neuromorphic computing is a computing paradigm that mimics human brain operation. By definition, neuromorphic computing systems implement synapse and neuron analogs, and mimic the spike-based activation and operation of neurons, for computation. There is active research on both hardware (E.g. IBM TrueNorth, Brainchip Akida) and software realization (spiking neural networks or SNNs) of this technology. As discussed in chapter 12, photonic PIM is a viable strategy for ML acceleration. Given how the photonic main memories from chapters 11 and 12 already have the inherent ability to mimic synapses through their PCM memory cells, an investigation into how to realize neuromorphic systems using these memory platforms could yield SNN-compatible, low-energy systems.

BIBLIOGRAPHY

[1] S. V. Mahadevkar, B. Khemani, S. Patil, K. Kotecha, D. R. Vora, A. Abraham and L. A. Gabralla, "A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions," *IEEE Access,* vol. 10, no. 2169-3536, pp. 107293-107329, 2022.

[2] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre and D. R. I. Heintz, "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey," *ACM Computing Surveys,* vol. 56, no. 2, pp. 1-40, 2023.

[3] M. Soori, B. Arezoo and R. Dastres, "Artificial intelligence, machine learning and deep learning in advanced robotics, a review," *Cognitive Robotics,* vol. 3, pp. 54-70, 2023.

[4] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng and S. Z. Li, "A Survey on Generative Diffusion Models," *IEEE Transactions on Knowledge and Data Engineering,* pp. 1-20, 2024.

[5] Y. Bengio, A. Courville and P. vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 8, pp. 1798-1828, 2013.

[6] Z. Li, W. Yang, S. Peng and F. Liu, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in *arXiv:2004.02806 [cs.CV]*, 2020.

[7] C. Gao, D. Neil, E. Ceolini, S.-C. Liu and T. Delburk, "DeltaRNN: A Power-efficient Recurrent Neural Network Accelerator," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2018.

[8] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *arXiv:1609.02907 [cs.LG]*, 2016.

[9] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph Attention Networks," in *arXiv:1710.10903 [stat.ML]*, 2017.

[10] A. Vaswani, N. Shazeer, N. Pamar, J. Uszkoreit, L. Jones, A. N. Gomez, L. kaiser and I. Polosukhin, "Attention is All You Need," in *arXiv:1706.03762 [cs.CL]*, 2014.

[11] S. Markidis, S. W. D. Chien, E. Laure, I. B. Peng and J. S. Vetter, "Nvidia Tensor Core Programmability, Performance & Precision," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018.

[12] J. Reinders, "Intel Xeon Phi Coprocessor Architecture and Tools: The Guide for Application Developers," Apress, 2013.

[13] Nvidia, "nvidia.com," [Online]. Available: https://www.nvidia.com/en-us/data-center/h100/. [Accessed 18 4 2024].

[14] AMD, "amd.com," [Online]. Available: https://www.amd.com/en/newsroom/press-releases/2023-12-6-amd-delivers-leadership-portfolio-of-data-center-a.html. [Accessed 18 4 2024].

[15] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Computer Architecture News,* vol. 23, no. 1, pp. 20-24, 1995.

[16] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney and K. Keutzer, "AI and Memory Wall," *IEEE Micro,* 2024.

[17] S. Han, H. Mao and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *International Conference on Learning Representations (ICLR)*, 2016.

[18] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier and J. dean, "Carbon Emissions and Large Neural Network Training," in *arXiv preprint arXiv:2104.10350*, 2021.

[19] AMD Inc., "AMD CDNA 3 Architecture," https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf, 2023.

[20] Nvidia, "NVIDIA Blackwell Platform Arrives to Power a New Era of Computing," https://nvidianews.nvidia.com/news/nvidia-blackwell-platform-arrives-to-power-a-new-era-of-computing, 2024.

[21] Datacenterdynamics, "Upcoming Nvidia Blackwell GPU will consume 1kW of power," https://www.datacenterdynamics.com/, 2024.

[22] J. Hamilton, "Datacenter Networks Are in My Way," *ACM SIGOPS Operating Systems Review,* vol. 44, no. 4, pp. 2-13, 2010.

[23] Venturebeat, "AMD addressing the challenge of energy efficient computing," 2022.

[24] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agarwal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Bochers, R. Boyle, P.-L. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland and e. al, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *International Symposium on Computer Architecture (ISCA)*, 2017.

[25] Apple, "Deploying Transformers on the Apple Neural Engine," https://machinelearning.apple.com/research/neural-engine-transformers, 2022.

[26] Intel, "Habana Goya HL-100 Datasheet," https://habana.ai/wp-content/uploads/pdf/2020/Habana%20GOYA%20HL-100%20Datasheet.pdf.

[27] S. Amborgio, P. Narayan, H. Tsai, R. M. Shelby, I. Boybat, C. d. Noifo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi and G. W. Burr, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature,* vol. 558, pp. 60-67, 2018.

[28] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla and Y. N. P. D. G.-J. N. B. T. M. B. B. B. J. B. A. B. B. C. M. C. D. D. Nabil Imam, "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 34, no. 10, pp. 1537-1557, 2015.

[29] M. V. Ornhag, P. Guler, D. Knyagini and M. Borg, "Accelerating AI Using Next-Generation Hardware: Possibilities and Challenges With Analog In-Memory Computing," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

[30] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan and Y. Xie, "DRISA: A DRAM-based Reconfigurable In-Situ Accelerator," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017.

[31] M. Imani, A. Rahimi and T. S. Rosing, "Low Power Hyperdimensional Computing using Processing-In-Memory," in *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[32] S. Pasricha and M. Nikdast, "A Survey of Silicon Photonics for Energy Efficient Manycore Computing," *IEEE Design and Test,* vol. 37, no. 4, pp. 60-81, 2020.

[33] L. Chrostowski, H. Shoman, M. Hammood, H. Yun, J. Jhoja, E. Luan, S. Lin, A. Mistry, D. Witt, N. A. Jaeger and e. al, "Silicon photonic circuit design using rapid prototyping foundry process design kits," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 25, no. 5, pp. 1-26, 2019.

[34] D. A. Miller, "Silicon Photonics: Meshing Optics with Applications," *Nature Photonics,* vol. 11, no. 7, 2017.

[35] A. R. Totovic, G. Dabos, N. Passalis, A. Tefas and N. Pleros, "Femtojoule per MAC neuromorphic photonics: An energy and technology roadmap," *IEEE Selected Topics in Quantum Electronics,* vol. 26, no. 5, pp. 1-15, 2020.

[36] Y. Shen, N. C. harris, S. Skirlo, M. Prabhu, T. B.-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund and e. al, "Deep Learning with Coherent Nanophotonic Circuits," *Nature Photonics,* vol. 11, no. 7, 2017.

[37] F. Sunny, E. Taheri, M. Nikdast and S. Pasricha, "A survey on silicon photonics for deep learning," *ACM Journal of Emerging Technologies in Computing Systems,* vol. 17, no. 4, pp. 1-57, 2021.

[38] A. N. Tait, M. A. Nahmias, B. J. Shastri and P. R. Pruncal, "Broadcast-and-weight interconnects for integrated distributed processing systems," in *Optical Interconnects Conference*, 2014.

[39] P. Pintus, M. Hofbauer, C. L. Manganelli, M. Fournier, S. Gundavarapu, O. Lemonnier, F. Gambini, L. Adelmini, C. Meinhart, C. Kopp, F. Testa, H. Zimmermann and C. J. Oton, "PWM-Driven Thermally Tunable Silcon Micronring Resonators: Design, Fabrication, and Characterization," *Laser & Photonic Reviews,* vol. 13, no. 9, 2019.

[40] S. Abel, T. Stoferle, C. Marchiori, D. Caimi, L. Czornomaz, M. Stuckelberger, M. Sousa, B. J. Offrein and J. Fompeyrine, "A Hybrid Barium Titanate–Silicon Photonics Platform for Ultraefficient Electro-Optic Tuning," *IEEE Journal of Lightwave Technology,* vol. 34, no. 8, pp. 1688-1693, 2015.

[41] M. Buria, X. Wang, M. Li, L. Chrostowski and J. Azana, "Wideband dynamic microwave frequency identification system using a low-power ultracompact silicon photonic chip," *Nature Communications,* vol. 7, no. 13004, 2016.

[42] F. Sunny, A. Mirza, M. Nikdast and S. Pasricha, "ARXON: A Framework for Approximate Communication Over Photonic Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration Systems,* vol. 29, no. 6, pp. 1206-1219, 2021.

[43] F. Sunny, A. Mirza, M. Nikdast and S. Pasricha, "CrossLight: A Cross-layer optimized silicon photonic neural network accelerator," in *IEEE/ACM Design Automation Conference*, 2021.

[44] F. Sunny, A. Mirza, M. Nikdast and S. Pasricha, "ROBIN: A robust optical binary neural network accelerator," *ACM Transactions on Embedded Computing Systems (TECS),* vol. 20, no. 5s, pp. 1-24, 2021.

[45] F. Sunny, M. Nikdast and S. Pasricha, "SONIC: A sparse neural network inference acceelrator with silicon photonics for energy efficient deep learning," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022.

[46] F. Sunny, M. Nikdast and S. Pasricha, "A silicon photonic accelerator for convolutional neural networks with heterogeneous quantization," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2022.

[47] F. Sunny, M. Nikdast and S. Pasricha, "RecLight: A recurrent neural network accelerator with integrated silicon photonics," in *IEEE Computer Soceity Annual Symposium on VLSI (ISVLSI)*, 2022.

[48] S. Afifi, F. Sunny, A. Shafiee, M. Nikdast and S. Pasricha, "GHOST: A Graph Neural Network Accelerator using Silicon Photonics," *ACM transactions on Embedded Computing Systems (TECS),* vol. 22, no. 5s, pp. 1-25, 2023.

[49] S. Afifi, F. Sunny, M. Nikdast and S. Pasricha, "TRON: transformer neural network acceleration with non-coherent silicon photonics," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2023.

[50] F. Sunny, E. Taheri, M. Nikdast and S. Pasricha, "Machine Learning Accelerators in 2.5D Chiplet Platforms with Silicon Photonics," in *IEEE/ACM Design, Automation, & Test in Europe (DATE)*, 2023.

[51] F. Sunny, A. Shafiee, B. Charbonnier, M. Nikdast and S. Pasricha, "COMET: A Cross-Layer Optimized Optical Phase Change Main Memory Architecture," in *IEEE/ACM Design, Automation, & Test in Europe (DATE)*, 2023.

[52] F. Sunny, A. Shafiee, A. Balasubramaniam, M. Nikdast and S. Pasricha, "OPIMA: Optical Processing-In-Memory for Convolutional Neural Network Acceleration," *IEEE Transactions on Computer Aided Design (TCAD),* (under review).

[53] S. V. R. Chittamuru and S. Pasricha, "Crosstalk mitigation for highradix and low-diameter photonic NoC architectures," *IEEE Design and Test,* vol. 32, no. 3, pp. 29-39, 2015.

[54] M. Milanizadeh, D. Aguiar, A. Melloni and F. Morichetti, "Canceling Thermal Cross-Talk Effects in Photonic Integrated Circuits," *IEEE Journal of Lightwave Technology (JLT),* vol. 37, no. 4, pp. 1325-1332, 2019.

[55] Nvidia, "Nvidia Ampere Architecture," Nvidia.

[56] Cerebras, "Cerebras Wafer-scale Engine," www.cerebras.net.

[57] S. Pasricha and N. Dutt, "On-Chip Communication Architectures," Morgan Kauffman, ISBN 978-0-12-373892-9, 2008.

[58] T. Alexoudi, N. Terzinidis, S. Pitris, M. Moralis-Pegios, P. Maniotis, C. Vagionas, C. Mitsolidou, G. Mourgias-Alexandris, G. T. Kanelios, A. Miliou, K. Vyrsokinos and N. Pleros, "Optics in Computing: From Photonic Network-on-Chip to Chip-to-Chip Interconnects and Disintegrated Architectures," *IEEE Journal of Lightwave Technology,* vol. 37, no. 2, 2019.

[59] Q. Xu, T. Mytkowicz and N. S. Kim, "Approximate Computing: A Survey," *IEEE Design & Test,* vol. 33, no. 1, 2016.

[60] F. Qiao, N. Zhou, Y. Chen and H. Yang, "Approximate Computing in Chrominance Cache for Image/Video Processing," in *IEEE International Conference on Multimedia Big Data*, 2015.

[61] H. Ahmadvand, M. Goudarzi and F. Foroutan, "Gapprox: using Gallup approach for approximation in Big Data processing," *Journal of Big Data,* vol. 6, no. 20, 2019.

[62] D. T. Nguyen, H. Kim, H.-J. Lee and I.-J. Chang, "An Approximate Memory Architecture for a Reduction of Refresh Power Consumption in Deep Learning Applications," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.

[63] H. Younes, A. Ibrahim, M. Rizk and M. Valle, "Algorithmic Level Approximate Computing for Machine Learning Classifiers," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019.

[64] S. Sen and A. Raghunathan, "Approximate Computing for Long Short Term Memory (LSTM) Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems ,* vol. 37, no. 11, pp. 2266-2276, 2018.

[65] A. Ibrahim, M. Osta, M. Alameh, M. saleh, H. Chible and M. Valle, "Approximate Computing Methods for Embedded Machine Learning," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018.

[66] P. Yellu, N. Boskov, M. A. Kinsy and Q. Yu, "Security Threats in Approximate Computing Systems," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2019.

[67] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm for energy-efficient design,," in *IEEE European Test Symposium*, 2013.

[68] V. K. Chippa, S. Venkataramani, S. T. Chakradhar, K. Roy and A. Raghunathan, "Approximate computing: An integrated hardware approach," in *Asilomar Conference on Signals, Systems and Computers*, 2013.

[69] Z. Yang, A. Jain, J. Liang, J. han and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in *IEEE International Conference on Nanotechnology (NANO)*, 2013.

[70] M. Ramasamy, G. Narmadha and S. Deivasigamani, "Carry based approximate full adder for low power approximate computing," in *International Conference on Smart Computing & Communications (ICSCC)*, 2019.

[71] A. Raha, S. Sahir, H. Jayakumar and V. Raghunathan, "Quality Configurable Approximate DRAM," *IEEE Transactions on Computers,* vol. 66, no. 7, pp. 1172-1187, 2017.

[72] A. Sampson, A. Baixo, B. Ransford, T. Moreau, J. Yip, L. Ceze and M. Oskin, "ACCEPT: A Programmer-Guided Compiler Framework for Practical Approximate Computing," University of Washington, 2014.

[73] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasm, L. Ceze and D. Grossman, "EnerJ: Approximate Data Types for Safe and General Low-Power Computation," in *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLD)*, 2011.

[74] J. Park, H. Esmaelizadeh, X. Zhang, M. Naik and W. Harris, "FlexJava: Language support for safe and modular approximate programming," in *Joint Meeting on Foundations of Software Engineering (FSE)*, 2015.

[75] Y. Raparti and S. Pasricha, "DAPPER: Data Aware Approximate NoC for GPGPU Architectures," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2018.

[76] R. Boyapati, J. huang, P. Majumdar, K. H. Yum and E. J. Kim, "APPROX-NoC: A data approximation framework for Network-on-Chip architectures," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2017.

[77] L. Wang, X. Wang and Y. Wang, "ABDTR: Approximation-Based Dynamic Traffic Regulation for Networks-on-Chip Systems," in *IEEE International Conference on Computer Design (ICCD)*, 2017.

[78] A. B. Ahmed, D. Fujiki, H. Matsutani, M. Koibuchi and H. Amano, "AxNoC: Low-power Approximate Network-on-Chips using Critical-Path Isolation," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2018.

[79] J. Lee, C. killian, S. L. Beux and D. Chillet, "Approximate Nanophotonic Interconnects," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCs)*, 2019.

[80] F. Sunny, A. Mirza, I. Thakkar, S. pasricha and M. Nikdast, "LORAX: Loss-Aware Approximations for Energy-Efficient Silicon Photonic Networks-on-Chip," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2020.

[81] C. Bieneia, "Benchmarking Modern Multiprocessors," Princeton University, 2011.

[82] "TinyDNN," https://github.com/tiny-dnn/tiny-dnn.

[83] N. Binkert, "The Gem5 Simulator," *Comp. Arch. News,* 2011.

[84] C. K. Luk, "Pin: building customized program analysis tools with dynamic instrumentation," in *SIGPLAN*, 2005.

[85] R. Soref, "Electrooptical Effects in Silicon," *IEEE JSTQE,* pp. 123-129, 1987.

[86] S. V. R. Chittamuru, "HYDRA: Heterodyne Crosstalk Mitigation with Double Microring Resonators and Data Encoding for Photonic NoCs," *IEEE TVLSI,* 2018.

[87] M. Nikdast, "Crosstalk Noise in WDM-based Optical Netwroks-on-Chip," *IEEE TVLSI,* 2015.

[88] I. Thakkar, S. V. R. Chittamuru and S. Pasricha, "Improving the reliability and energy-efficiency of high bandwidth photonic NoC architectures with multi-level signalling," in *IEEE/ACM NOCS*, 2017.

[89] H. Li, A. Fourmigue, S. L. Beux, X. Letarte, I. O. Conner and G. Nicolescu, "Thermal Aware Design Method for VCSEL-based On-Chip Optical Interconnect," in *Design, Automation, and Test in Europe*, 2015.

[90] X. Wu, "SUOR: Sectioned Unidirectional Optical Ring for Chip Multiprocessor," *ACM JETC,* 2014.

[91] M. Mohamed, "Reliability-aware design flow for silicon photonics on-chip interconnect," *IEEE TVLSI,* 2014.

[92] K. Padmaraju, "Resolving the tehrmal challenges for silicon microring resonator devices," *Nanophotonics,* 2013.

[93] C. Sun, "Single-chip microprocessor that communicates directly using light," *Nature,* vol. 528, pp. 24-31, 2015.

[94] T. Kajo, "Optical Multilevel Signalling for High bandwidth and Power-Efficient On-Chip Interconnects," *PTL,* vol. 27, no. 19, 2015.

[95] A. Roshan-Zamir, "A 40Gbps PAM4 silicon microring resonator modulator transmitter in 65nm CMOS," in *OIC*, 2016.

[96] X. Wu, "A 20 Gb/s NRZ/PAM-4 1V transmitter in 40nm CMOS driving a Si-photonic modulator in 0.13um CMOS," in *ISSCC*, 2013.

[97] R. D.-.. Demers, "Ultrafast pulse-amplitude modulation with a femtojoule silicon photonic modulator," *Optica,* vol. 3, no. 6, 2016.

[98] A. Joshi, "Silicon-photonic Clos networks for global on-ship communication," in *IEEE/ACM NOCS*, 2009.

[99] S. V. R. Chittamuru, S. Desai and S. Pasricha, "SwiftNoC: A reconfigurable silicon photonic network with multicast enabled channel sharing for multicore architectures," *ACM Journal on Emerging Technology in Computing Systems (JETC),* vol. 13, no. 4, pp. 1-27, 2017.

[100] C. Sun, C.-H. O. Chen, G. Kurien, L. Wei, J. Miller, A. Agarwal, L.-S. Peh and V. Stojanovic, "DSENT A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2012.

[101] K. Chen, "CACTI-3D: Architecture-level modeling for 3D diestacked DRAM main memory," in *IEEE/ACM DATE*, 2012.

[102] A. Stillmaker and B. Bass, "Scaling equations for the accurate predictions of CMOS device performance from 180 nm to 7 nm," *Integrations,* vol. 58, pp. 74-81, 2017.

[103] S. V. R. Chittamuru, I. Thakkar and S. Pasricha, "LIBRA: Thermal and Process Variation Aware Reliability Management in Photonic Networks-on-Chip," *IEEE Transactions on Multi-Scale Computing Systems (TMSCS),* vol. 4, no. 1, pp. 758-772, 2018.

[104] A. Mirza, S. pasricha and M. Nikdast, "Variation-Aware Inter-Device Matching in Silicon Photonic Microring Resonator Demultiplexers," in *IEEE Photonics Conference*, 2020.

[105] E. Totoni, B. Behzad, S. Ghike and J. Torrellas, "Comparing the power and performance of Intel's SCC to state-of-the-art CPUs and GPUs," in *IEEE International Symposium on Performance Analysis of Systems & Software*, 2012.

[106] M. M. Waldrop, "The chips are down for Moore's Law," *Nature News,* vol. 530, no. 7589, 2016.

[107] A. K. K. Ziabari, J. L. Abellan, R. Ubal, C. Chen, A. Joshi and D. Kaeli, "Leveraging Silicon-Photonic NoC for Designing Scalable GPUs," in *ACM on International Conference on Supercomputing*, 2015.

[108] W. A. Zortman, D. C. Trotter and M. R. Watts, "Silicon photonics manufacturing," *Optics Express,* vol. 18, no. 23, pp. 23598-23607, 2010.

[109] A. N. Tait, T. F. d. Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri and P. R. Prucnal, "Neuromorphic photonic networks using silicon photonic weight banks," *Scientific Reports,* vol. 7, 2017.

[110] V. Bangari, B. A. Marquez, H. Miller, A. N. Tait, M. A. Nahmias, T. F. d. Lima, H.-T. Peng, P. R. Pruncal and B. J. Shastri, "Digital Electronics and Analog Photonics for Convolutional Neural Networks (DEAP-CNNs)," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 26, no. 1, pp. 7701213-7701226, 2020.

[111] W. Liu, W. Liu, Y. Ye, Q. Lou, Y. Xie and L. Jiang, "HolyLight: A Nanophotonic Accelerator for Deep Learning in Data Centers," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[112] K. Shiflett, D. Wright, A. Karanth and A. Louri, "PIXEL: Photonic Neural Network Accelerator," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020.

[113] Z. Zhao, D. Liu, M. Li, Z. Ying, L. Zhang, B. Xu, B. Yu, R. T. Chen and D. Z. Pan, "Hardware-software co-design of slimmed optical neural networks," in *ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2019.

[114] G. Mourgias-Alexandris, A. Totovic, A. Tsakyridis, N. Passalis, K. Vyrsokinos, A. Tefas and N. Pleros, "Neuromorphic photonics with coherent linear neurons using dual-IQ modulation cells," *IEEE Journal of Lightwave Technology (JLT),* vol. 38, no. 4, pp. 811-819, 2019.

[115] C. Pask, "Generalized parameters for tunneling ray attenuation in optical fibers," *Journal of Optical Soceity of America,* vol. 68, no. 1, 1978.

[116] W. Bogaerts, P. D. Heyn, T. v. Vaerenbergh, K. D. Vos, S. K. Selvaraja, T. Claes, P. Dumon, P. Bienstman, D. V. Thourhout and R. Baets, "Silicon microring resonators," *Laser and Photonics Reviews,* vol. 6, no. 1, pp. 47-73, 2012.

[117] L. Lu, X. Li, W. Gao, X. Li, L. Zhou and J. Chen, "Silicon Non-Blocking 4 × 4 Optical Switch Chip Integrated With Both Thermal and Electro-Optic Tuners," *IEEE Photonics Journal,* vol. 11, no. 6, 2019.

[118] Ansys Lumerical, "Lumerical HEAT," https://courses.ansys.com/index.php/courses/lumerical-heat-intro/.

[119] S. De, R. Das, R. K. Varshney and T. Schneider, "Design and Simulation of Thermo-Optic Phase Shifters with Low Thermal Crosstalk for Dense Photonic Integration," *IEEE Access,* vol. 8, pp. 141632 - 141640, 2020.

[120] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedigs of the IEEE,* vol. 86, no. 11, pp. 2278 - 2324, 1998.

[121] QKeras, "QKeras: a quantization deep learning library for Tensorflow Keras," https://github.com/google/qkeras.

[122] L. H. Frandsen, P. I. Borel, Y. X. Zhuang, A. Harpoth, M. Thorhauge, M. Kristensen, W. Bogaerts, P. Dumon, R. Baets, V. Wiaux, J. Wouters and S. Beckx, "Ultralow-loss 3-dB photonic crystal waveguide splitter," *Optics Letters,* vol. 29, no. 14, pp. 1623-1625, 2004.

[123] Y.-C. Tu, P.-H. Fu and D.-W. Huang, "High-Efficiency ultra-broadband multi-tip edge couplers for integration of distributed feedback laser with silicon-on-insulator waveguide," *IEEE Photonics Journal,* vol. 11, no. 4, 2019.

[124] S. Bahirat and S. Pasricha, "OPAL: A Multi-layer Hybrid Photonic NoC for 3D ICs," in *IEEE/ACM Asia South Pacific Design Automation Conference (ASP-DAC)*, 2011.

[125] H. Jayatilleka, M. Caverley, N. A. F. Jaeger, S. Shekhar and L. Chrostowski, "Crosstalk limitations of Microring-Resonator based WDM Demultiplexers on SOI," in *IEEE Optical Interconnects Conference*, 2015.

[126] E. Timurdogan, C. M. Sorace-Agaskar, E. S. Hosseini, G. Leake, D. D. Coolbaugh and M. R. Watts, "Vertical junction silicon microdisk modulator with integrated thermal tuner," in *CLEO*, 2013.

[127] M. Pisatti, F. D. Bernardinis, P. Pascale, C. Nani, M. Sosio, E. Pozzati, N. Ghittori, F. Magni, M. Garampazzi, G. Bollati, A. Milani, F. Glunco, P. Uggetti, I. Fabiano, N. Codega, A. Bosi, N. Carta, D. Pellicone, G. Spelgatti, M. Cutrupi and A. Rossini, "A sub-250 mW 1-to-56Gb/s continuous-range PAM-4 42.5 dB IL ADC/DAC-based transceiver in 7 nm FinFET," in *IEEE International Solid-State Circuits Conference - (ISSCC)*, 2019.

[128] Z. Ruan, Y. Zhu, P. Chen, Y. Shi, S. He, X. Cai and L. Liu, "Efficient Hybrid Integration of Long-Wavelength VCSELs on Silicon Photonic Circuits," *IEEE Journal of Lightwave Technology,* vol. 38, no. 18, pp. 5100 - 5106, 2020.

[129] A. D. Gungordu, G. Dundar and M. B. Yelten, "A high performance TIA design in 40 nm CMOS," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020.

[130] B. Wang, Z. Huang, W. V. Sorin, X. Zeng, D. Lang, M. Fiorentino and R. G. Beausoleil, "A low-voltage Si-Ge avalanche photodiode for high-speed and energy efficient silicon photonic links," *IEEE Journal of Lightwave Technology (JLT),* vol. 38, no. 12, pp. 3156-3163, 2020.

[131] 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 and Test (D&T),* vol. 31, no. 5, pp. 55-65, 2014.

[132] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masero and M. Martina, "An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks," *Future Internet,* vol. 12, no. 7, 2020.

[133] M. Coubariaux, I. Hubara, D. Soudry, R. El-Yaniv and Y. Bengio, "BinaryNet: Training Deep Neural Networks with Weights and Activations constrained to +1 or -1," in *arXiv 2016, arXiv:1602.02830.*, 2016.

[134] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv and Y. Bengio, "Binarized Neural Networks," in *NeurIPS*, 2016.

[135] S. Bahirat and S. Pasricha, "METEOR: Hybrid Photonic Ring-Mesh Network-on-Chip for Multicore Architectures," *ACM Transactions on Embedded Computing Systems (TECS),* vol. 13, no. 3, 2014.

[136] S. Bahirat and S. Pasricha, "HELIX: Design and Synthesis of Hybrid Nanophotonic Application-Specific Network-on-Chip Architectures," in *IEEE International Symposium on Quality Electronic Design (ISQED)*, 2014.

[137] J. Anderson, S. Sun, Y. Alkabani, V. Sorger and T. El-Ghazawi, "Photonic Processor for Fully Discretized Neural Networks," in *IEEE International Conference on Application-specific Systems, Architectures and Processors*, 2019.

[138] F. Zokae, Q. Lou, N. Youngblood, W. Liu, Y. Xie and L. Jiang, "LightBulb: A Photonic-Nonvolatile-Memory-based Accelerator for Binarized Convolutional Neural Networks," in *IEEE/ACM Design and Test in Europe*, 2020.

[139] M. Nikdast, G. Nicolescu, J. Trajkovic and O. Liboiron-Ladouceur, "Chip-scale silicon photonic interconnects: A formal study on fabrication non-uniformity," *IEEE Journal of Lightwave Technology (JLT),* vol. 34, no. 16, 2016.

[140] Y. Bengio, N. Léonard and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," in *arXiv:13126199.*, 2013.

[141] Ansys Lumerical, "Ansys Lumerical MODE," https://www.ansys.com/products/optics/mode.

[142] Y. Liu, W. Sun, H. Xie, N. Zhang, K. Xu, Y. Yao, S. Xiao and Q. Song, "Adiabatic and Ultracompact Waveguide Tapers Based on Digital Metamaterials," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 25, no. 3, pp. 1-6, 2019.

[143] Z. Su, E. S. Hosseini, E. Timurdogan, J. Sun, G. Leake, D. D. Coolbaugh and M. R. Watts, "Reduced wafer-scale frequency variation in adiabatic microring resonators," in *Optical Fiber Conference (OFC)*, 2014.

[144] Q. Xu, D. Fattal and R. G. Beausoleil, "Silicon microring resonators with 1.5-μm radius," *Optics express,* vol. 16, no. 6, pp. 4309-4315, 2008.

[145] B. E. Little, S. T. Chu, H. A. Haus, J. Foresi and J.-P. Laine, "Microring resonator channel dropping filters," *IEEE Journal of Lightwave Technology (JLT),* vol. 15, no. 6, pp. 998-1005, 1997.

[146] J. Xia, A. Bianco, E. Bonetto and R. Gaudino, "On the design of microring resonator devices for switching applications in flexible-grid networks," in *IEEE International Conference on Communications (ICC)*, 2014.

[147] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen and O. Temam, "DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning," in *ACM ASPLOS*, 2014.

[148] J. Shen, A. Shikata, L. D. Fernando, N. Guthrie, B. Chen, M. Maddox, N. Mascarenhas, R. Kapusta and M. C. W. Coln, "A 16-bit 16-MS/s SAR ADC With On-Chip Calibration in 55-nm CMOS," *IEEE Journal of Solid-State Circuits,* vol. 53, no. 4, pp. 1149-1160, 2018.

[149] B. Wu, S. Zhu, B. We and Y. Chiu, "A 24.7 mW 65 nm CMOS SAR assisted CT modulator with second-order noise coupling achieving 45 MHz bandwidth and 75.3 dB SNDR," *IEEE Journal of Solid-State Circuits,* vol. 51, no. 12, p. 2893–2905, 2016.

[150] A. Mirza, F. Sunny, S. Pasricha and M. Nikdast, "Silicon Photonic Microring Resonators: Design Optimization under Fabrication Non-uniformity," in *IEEE/ACM Design, Automation and Test in Europe (DATE)*, 2020.

[151] M. Bahadori, M. Nikdast, S. Rumley, L. Y. Dai, N. Janosik, T. V. Vaerenbergh, A. Gazman, Q. Cheng, R. Polster and K. Bergman, "Design space exploration of microring resonators in silicon photonic interconnects: Impact of the ring curvature," *IEEE Journal of Lightwave Technology,* vol. 36, no. 13, pp. 2767-2782, 2018.

[152] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul and T. Krishna, "SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training," in *IEEE International Symposium on High Performance Computing Architecture*, 2020.

[153] S. Cass, "Taking AI to the edge: Google's TPU now comes in a maker-friendly package," *IEEE Spectrum,* vol. 56, no. 5, pp. 16-17, 2019.

[154] T. Luo, S. Liu, L. Li, Y. Wang, S. Zhang, T. Chen, Z. Xu, O. Temam and Y. Chen, "DaDianNao: A Neural Network Supercomputer," *IEEE Transactions on Computers,* vol. 66, no. 1, pp. 73-88, 2017.

[155] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I. A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S. C. Liu and T. Delbruck, "NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representation of Feature Maps," *IEEE Transactions on Neural Network Learning Systems,* vol. 30, no. 3, pp. 644-656, 2016.

[156] P. Guo, H. Ma, R. Chen, P. Li, S. Xie and D. Wang, "FBNA: A fully Binarized Neural Network Accelerator," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*, 2018.

[157] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre and K. Vissers, "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in *ACM/SIGDA FPGA*, 2017.

[158] K. Simonyan and A. Zisserman, "Very deep convolutional neyworks for large-scale image recognition," in *arxiv:1409.1556*, 2014.

[159] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen and P. Dubey, "Faster CNNs with Direct Sparse Convolutions and Guided Pruning," in *arXiv:1608.01409 [cs.CV]*, 2016.

[160] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen and Y. Chen, "Cambricon-X: An accelerator for sparse neural networks," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016.

[161] W. You and C. Wu, "RSNN: A software/hardware co-optimized framework for sparse convolutional neural networks on FPGAs," *IEEE Access,* vol. 9, pp. 949-960, 2020.

[162] J. Gu, Z. Zhao, C. Feng, M. Liu, R. T. Chen and D. Z. Pan, "Towards Area-Efficient Optical Neural Networks: An FFT-based Architecture," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.

[163] D. Dang, S. V. R. Chittamuru, S. Pasricha, R. Mahapatra and D. Sahoo, "BPLight-CNN: A Photonics-based Backpropagation Accelerator for Deep Learning," *ACM Journal on Emerging Technologies in Computing Systems (JETC),* vol. 17, no. 4, 2021.

[164] S. Banerjee, M. Nikdast and K. Chakrabarty, "Modeling silicon-photonic neural networks under uncertainties," in *Design, Automation, and Test in Europe (DATE)*, 2021.

[165] M. H. Zhu and S. Gupta, "To prune or not to prune: Exploring efficacy of pruning for model compresion," in *arXiv:1710.01878v2*, 2017.

[166] S. Han, H. Mao and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *arXiv:1510.00149v5 [cs.CV]*, 2015.

[167] R. Inti, M. Mansuri, J. Kennedy, J. Qiu, C.-M. Hsu, J. Sharma, H. Li, B. Casper and J. Jaussi, "A scalable 32-to-56Gb/s 0.56-to-1.28pJ/b voltage-mode VCSEL-based optical transmitter in 28nm CMOS," in *IEEE Custome Integrated Cicuits Conference (CICC)*, 2021.

[168] C. M. Yang and T. H. kuo, "A 3 mW 6-bit 4 GS/s subranging ADC with subrange-dependent embedded references," *IEEE Transactions on Circuits and Systems (TCAS) !!: Express Briefs,* vol. 68, no. 7, pp. 2312-2316, 2021.

[169] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda and K. Keutzer, "Mixed Precision Quantization of Convnets via Differentiable Neural Architecture Search," in *arXiv:1812.00090 [cs.CV]*, 2018.

[170] M. Nikolic, G. B. Hence, C. Bannon, A. D. Lascroz, M. Courbariax, Y. Bengio, V. Gripon and A. Moshovos, "BitPruning: Learning Bitlengths for Aggressive and Accurate Quantization," in *arXiv:2002.03090 [cs.LG]*, 2020.

[171] E. Soufleri and K. Roy, "Network Compression via Mixed Precision Quantization Using a Multi-Layer Perceptron for the Bit-Width Allocation," *IEEE Access,* vol. 9, 2021.

[172] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu and P. Micikevicius, "Accelerating Sparse Deep Neural Networks," in *arXiv:2104.08378 [cs.LG]*, 2021.

[173] J. Zhu, Q. Chao, H. Huang, Y. Zhao, Y. Li, L. Tao, X. She, H. Liao, R. Huang, Z. Zhu, X. Liu, Z. Sheng and F. Gan, "Compact, broadband, and low-loss silicon photonic arbitrary ratio power splitter using adiabatic taper," *Applied Optics,* vol. 60, no. 2, pp. 413-416, 2021.

[174] S. Kaur and R.-S. Kaler, "Ultrahigh Speed Reconfigurable Logic Operations Based on Single Semiconductor Optical Amplifier," *Journal of the Optical Soceity of Korea,* vol. 16, no. 1, pp. 13-16, 2012.

[175] L. Kull, T. Toifi, M. Schmatz, P. A. Francese, C. Menolfi, M. Brandli, M. Kossel, T. Morf, T. M. Andersen and Y. Leblebici, "A 3.1 mW 8b 1.2 GS/s Single-Channel Asynchronous SAR ADC With Alternate Comparators for Enhanced Speed in 32 nm Digital SOI CMOS," *IEEE Journal of Solid-State Circuits,* vol. 48, no. 12, pp. 3049-3058, 2013.

[176] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[177] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[178] V. S. Lalapura, J. Amudha and H. S. Satheesh, "Recurrent Neural Networks for Edge Intelligence: A Survey," *ACM Computing Surveys,* vol. 54, no. 4, pp. 1-38, 2021.

[179] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 1986.

[180] K. Cho, B. v. Merrienboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8*, 2014.

[181] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[182] S. Cao, C. Zhang, Z. Yao, W. Xiao, L. Nie, D. Zhan, Y. Liu, M. wu and L. Zhang, "Efficient and effective sparse LSTM on FPGA with bank-balanced sparsity," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019.

[183] S. Wang, Z. Li, C. Ding, B. Yuan, Y. Wang, Q. Qiu and Y. Liang, "C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2018.

[184] E. Azari and S. Vrudhula, "ELSA: A throughput-optimized design of an LSTM accelerator for energy-constrained devices," *ACM Transactions on Embedded Computing Systems,* vol. 19, no. 1, pp. 1-21, 2020.

[185] F. Conti, L. Cavigelli, G. Paulin, I. Susmelj and L. Benini, "Chipmunk: A systolically scalable 0.9 mm2,3.08Gop/s/mW @ 1.2 mW accelerator for near-sensor recurrent neural network inference," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2018.

[186] C. Gao, A. Rios-Navarro, X. Chen, S.-C. Liu and T. Delbruck, "EdgeDRNN: Recurrent Neural Network Accelerator for Edge Inference," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems ,* vol. 10, no. 4, pp. 419-432, 2020.

[187] Q. Cheng, J. Kwon, M. Glick, M. Bahadori, L. P. Carloni and K. Bergman, "Silicon Photonics Codesign for Deep Learning," *Proceedings of the IEEE,* vol. 108, no. 8, pp. 1261-1282, 2020.

[188] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *ACM Design, Automation, and Test in Europe (DATE)*, 2017.

[189] C. Wang, C. Li, J. Dai and Z. Wang, "Study on in-chip phase locked high brightness bottom emitting Talbot-VCSELs array," in *AOPC*, 2020.

[190] A. N. Tait, T. F. d. Lima, M. A. Nahmias, H. B. Miller, H.-T. Peng, B. J. Shastri and P. R. Pruncal, "Silicon photonic modulator neuron," *Physical Review Applied,* vol. 11, no. 6, 2019.

[191] I. Thakkar, S. V. R. Chittamuru and S. Pasricha, "Run-Time Laser Power Management in Photonic NoCs with On-Chip Semiconductor Optical Amplifiers," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2016.

[192] W. S. Beutenberg. [Online]. Available: https://www.bgc-jena.mpg.de/wetter/. [Accessed 12 4 2024].

[193] X. Li and L. Zhou, "A survey of high-speed high-resolution current steering DACs," *Journal of Semiconductors,* vol. 41, no. 11, 2020.

[194] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open,* vol. 1, pp. 57-81, 2020.

[195] Z. Wu, S. Pan, F. Chen, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS,* vol. 32, no. 1, pp. 4-25, 2021.

[196] S. Abadal, A. Jain, R. Guirado, J. Lopez-Alonso and E. Alarcon, "Computing graph neural networks: A survey from algorithms to accelerators," *ACM Computing Surveys,* vol. 54, no. 9, 2021.

[197] J. Shalf, "The future of Computing beyond Moore's law," *Philosophical Transactions of the Royal Soceity A,* vol. 378, no. 2166, 2020.

[198] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks,* vol. 20, no. 1, pp. 61-80, 2008.

[199] W. Hamilton, Z. Yang and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing,* vol. 30, pp. 1024-1034, 2017.

[200] K. Xu, W. Hu, J. Leskovec and S. Jegelka, "How powerful are graph neural networks?," in *arXiv:1810.00826*, 2018.

[201] M. Fey and J. E. Lenssen, "Fast graph representation learning with Pytorch Geometric," in *arXiv:1903.02428*, 2019.

[202] M. Y. Wang, "Deep graph library: Towards efficient and scalable deep learning on graphs," in *ICLR*, 2019.

[203] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou and Y. Dai, "NeuGraph: Parallel Deep Neural Network Computation on Large Graphs," in *USENIX*, 2019.

[204] K. Kinningham, P. Levis and C. Re, "Greta: Hardware optimized graph processing for gnns," in *ReCoML*, 2020.

[205] A. Auten, M. Tomei and R. Kumar, "Hardware accelerator for large graph neural networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[206] S. Liang, Y. Wang, C. Liu, L. He, H. Li, D. Xu and X. Li, "EnGN: A high-throughput and energy-efficient accelerator for large graph neural networks," *IEEE Transactions on Computers,* vol. 70, no. 9, pp. 1511-1525, 2021.

[207] M. Yan, L. Deng, X. Hu, L. Liang, Y. Feng, X. Ye, Z. Zhang, D. Fan and Y. Xie, "Hygcn: A gcn accelerator with hybrid architecture," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019.

[208] K. Kinningham, P. Levis and C. Re, "GRIP: A graph neural network accelerator architecture," *IEEE Transactions on Computers (TC),* vol. 72, no. 4, pp. 914-925, 2022.

[209] C. Liu, H. Liu, H. Jin, X. Liao, Y. Zhang, Z. Duan, J. Xu and H. Li, "ReGNN: a ReRAM-based heterogeneous architecture for general graph neural networks," in *IEEE/ACM Design Automation Conference*, 2022.

[210] A. I. Arka, J. R. Doppa, P. P. Pande, B. K. Joardar and K. Chakrabarti, "ReGraphX: NoC-enabled 3D heterogeneous ReRAM architecture for training graph neural networks," in *IEEE Design Automation Conference*, 2021.

[211] A. Mirza, F. Sunny, S. Pasricha and M. Nikdast, "Silicon Photonic Microring Resonators: A Comprehensive Design-Space Exploration and Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 41, no. 10, pp. 3359-3372, 2022.

[212] K. Vandoorne, J. Dambre, D. Verstraten, B. Schrauwen and P. Bienstman, "Parallel reservoir computing using optical amplifiers. IEEE transactions on neural networks," *IEEE Transactions on Neural Networks,* vol. 22, no. 9, pp. 1469-1481, 2011.

[213] Z. Wei, A. Arora, P. Patel and L. John, "Design space exploration for Softmax implementation," in *IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP)*, 2020.

[214] HP Labs, "Cacti," [Online]. Available: https://www.hpl.hp.com/research/cacti. [Accessed 12 4 2024].

[215] S. Li, Z. Yang, D. Reddy, A. Srivastava and B. Jacob, "DRAMSim3: A cycle-accurate, thermal-capable DRAM simulator," *IEEE Computer Architecture Letters,* vol. 19, no. 2, pp. 106-109, 2020.

[216] Ansys, "Ansys Producst Photonics," [Online]. Available: https://www.ansys.com/products/photonics.. [Accessed 12 4 2024].

[217] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS*, 2014.

[218] J. Devlin, M.-W. Cheng, K. Lee and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[219] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *ICLR*, 2020.

[220] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[221] M. Zhou, W. Xu, J. Kang and T. Rosing, "TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformer," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022.

[222] S. Lu, M. Wang, S. Liang, J. Lin and Z. Wang, "Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer," in *IEEE International System-on-Chip Conference*, 2020.

[223] M. Sun, H. Ma, G. Kang, Y. Jiang, T. Chen, X. Ma, Z. Wang and Y. Wang, "VAQF: Fully automatic software-hardware co-design framework for low-bit vision transformer," in *arXiv:2201.06618v2*, 2022.

[224] P. Qi, E. H.-M. Sha, Q. Zhuge, H. Peng, S. Huang, Z. Kong, Y. Song and B. Li, "Accelerating framework of transformer by hardware design and model compression co-optimization," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021.

[225] C. Li, C. E. Graves, X. Sheng, D. Miller, M. Foltin, G. Pedretti and J. P. Strachan, "Analog content-addressable memories with memristors," *Nature Communications,* vol. 11, pp. 1-8, 2020.

[226] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," in *arXiv:1606.08415*, 2016.

[227] V. K. Kukkala, J. Tunnell, S. Pasricha and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronic Magazine,* vol. 7, no. 5, pp. 18-25, 2018.

[228] Q. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015.

[229] V. K. Kukkala, S. V. Thiruloga and S. Pasricha, "INDRA: Intrusion Detection using Recurrent Autoencoders in Automotive Embedded Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems,* vol. 39, no. 11, 2020.

[230] S. Jaiswal and T. Jaiswal, "Survey on recommender system using deep learning networks," *Artificial Intelligence Evolution,* vol. 1, no. 2, pp. 63-144, 2020.

[231] S. Pasricha, V. Ugave, Q. Han and C. Anderson, "LearnLoc: A Framework for Smart Indoor Localization with Embedded Mobile Devices," in *ACM/IEEE CODES+ISSS*, 2015.

[232] N. Jouppi, C. Young, N. Patil and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro,* vol. 38, no. 3, pp. 10-19, 2018.

[233] V. Sundaram, Q. Chen, T. Wang, H. Lu, Y. Suzuki, V. Smet, M. Kobayashi, R. Pulugurtha and R. Tummala, "Low cost, high performance, and high reliability 2.5 D silicon interposer," in *IEEE Electronic Components and Technology Conference*, 2013.

[234] J. Kim, G. Murali, H. Park, E. Qin, H. Kwon, V. C. K. Chekuri, N. M. Rahman, N. Dasari, A. Singh, M. Lee, H. M. Torun, K. Roy, M. Swaminathan, S. Mukhopadhyay, T. Krishna and S. K. Lim, "Architecture, chip, and package codesign flow for interposer-based 2.5-D chiplet integration enabling heterogeneous IP reuse," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 2020, no. 11, pp. 2424-2437, 2020.

[235] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanasundharam, G. Donley, A. Smith, L. Hewitt and S. Naffziger, "Zeppelin: An SoC for multichip architectures," *IEEE Journal of Solid-State Circuits,* vol. 54, no. 1, pp. 133-143, 2019.

[236] P. Fotouhi, S. Werner, R. Proietti, X. Xiao and S. J. B. Yoo, "Enabling scalable disintegrated computing systems with AWGR-based 2.5D interconnection networks," *IEEE Journal of Optical Communications and Networking ,* vol. 11, no. 7, pp. 333-346, 2019.

[237] A. Narayan, Y. Thonnart, P. Vivet and A. K. Coskun, "PROWAVES: Proactive Runtime Wavelength Selection for Energy-Efficient Photonic NoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 40, no. 10, pp. 2156-2169, 2021.

[238] S. Bahirat and S. Pasricha, "Exploring hybrid photonic networks-on-chip for emerging chip multiprocessors," in *IEEE/ACM CODES+ISSS*, 2009.

[239] D. A. B. Miller, "Device Requirements for Optical Interconnects to Silicon Chips," *Proceedings of the IEEE,* vol. 97, no. 7, pp. 1166-1185, 2009.

[240] H. Kwon, A. Samajdar and T. Krishna, "MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects," in *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2018.

[241] Y. Li, A. Louri and A. Karanth, "Scaling deep-learning inference with chiplet-based architecture and photonic interconnects," in *ACM/IEEE Design Automation Conference (DAC)*, 2021.

[242] Y. Li, A. Louri and A. Karanth, "SPRINT: A high-performance, energy-efficient, and scalable chiplet-based accelerator with photonic interconnects for CNN inference," *IEEE Transactions on Parallel and Distributed Systems,* vol. 33, no. 10, pp. 2332-2345, 2022.

[243] S. Banerjee, M. Nikdast, S. Pasricha and K. Chakrabarty, "Pruning Coherent Integrated Photonic Neural Networks," *IEEE Journal of Selected Topics in Quantum Electronics (JSTQE),* vol. 29, no. 2, 2023.

[244] C. Demirkiran, F. Eris, G. Wang, J. Elmhurst, N. Moore, N. C. Harris, A. Basumalik, V. J. Reddi, A. Joshi and D. Bunander, "An electro-photonic system for accelerating deep neural networks," *ACM Journal on Emerging Technologies in Computing Systems,* vol. 19, no. 4, pp. 1-31, 2023.

[245] F. K. Law, M. R. Uddin, H. Hashim and Y. H. Won, "Demonstration of photonic micro-ring resonator based digital bit magnitude comparator," *Optical and Quantum Electronics,* vol. 51, 2019.

[246] J. Huang, H. Ma, D. Chen, H. Yuan, J. Zhang, Z. Li, J. Han, J. Wu and J. Yang, "Digital nanophotonics: The highway to the integration of subwavelength-scale photonics," *Nanophotonics,* vol. 10, no. 3, 2020.

[247] M. A. Tran, D. huang, T. Komljenovic, J. Peters, A. Malik and J. E. Bowers, "Ultra-low-loss silicon waveguides for hetero-geneously integrated silicon/III-V photonics," *Applied Sciences,* vol. 8, no. 7, 2018.

[248] Z. Zhou, B. Yin and J. Michel, "On-chip light sources for silicon photonics," *Light: Science and Applications,* vol. 4, no. e358, 2015.

[249] S. Nambiar, P. Sethi and S. K. Selvaraja, "Grating-assisted fiber to chip coupling for SOI photonic circuits," *Applied Sciences,* vol. 8, no. 7, 2018.

[250] P. Ehrett, T. Austin and V. Bertacco, "SiPterposer: A fault-tolerant substrate for flexible system-in-package design," in *Design, Automation & Test in Europe (DATE)*, 2019.

[251] E. Taheri, S. Pasricha and M. Nikdast, "DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5D chiplet-based networks," in *IEEE/ACM Design, Automation & Test in Europe (DATE)*, 2022.

[252] E. Taheri, S. Pasricha and M. Nikdast, "ReSiPI: A reconfigurable silicon-photonic 2.5D chiplet network with PCMs for energy-efficient interposer communication," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022.

[253] T. Y. Teo, M. Krbal, J. Mistrik, J. Priryl, L. Lu and R. E. Simpson, "Comparison and analysis of phase change materials-based reconfigurable silicon photonic directional couplers," *Optical Materials Express,* vol. 12, no. 2, pp. 606-621, 2022.

[254] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie and J.-R. Wen, "A Survey of Large Language Models," in *arXiv:2303.18223 [cs.CL]*, 2023.

[255] X. Gao, C. Shuan, C. Hu, Z. Niu and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access,* vol. 7, pp. 82512-52512, 2019.

[256] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson and C. Guestrin, "PowerGraph: Distributed graph-parallel computation on natural graphs," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012.

[257] S. Salihoglu and J. Widom, "GPS: A graph processing system," in *International Conference on Scientific and Statistical Database Management (SSDBM)*, 2013.

[258] S. K. Kim and M. Popovici, "Future of dynamic random-access memory as main memory," *MRS Bulletin,* vol. 43, pp. 334-339, 2018.

[259] A. V. Krishnamoorthy, H. Schwetman, X. Zheng and R. Ho, "Energy-Efficient Photonics in Future High-Connectivity Computing Systems," *IEEE Journal of Lightwave Technology (JLT),* vol. 33, no. 4, pp. 889-900, 2015.

[260] H. Ishiwara, "Ferroelectric random access memories," *Journal of Nanoscience and Nanotechnology,* vol. 12, no. 10, pp. 7619-7627, 2012.

[261] Y. Chen, "ReRAM: History, Status, and Future," *IEEE Transactions on Electron Devices,* vol. 67, no. 4, pp. 1420-1433, 2020.

[262] N. S. Kim, C. Song, W. Y. Cho, J. Huang and M. Jung, "LL-PCM: Low-latency phase change memory architecture," in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[263] S. Song and A. Das, "Design Methodologies for Reliable and Energy-efficient PCM Systems," in *IEEE International Green and Sustainable Computing Workshops (IGSC)*, 2020.

[264] A. Shafiee, S. Pasricha and M. Nikdast, "A Survey on Optical Phase-Change Memory: The Promise and Challenges," *IEEE Access,* vol. 11, pp. 11781-11803, 2023.

[265] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. L. Gallo, X. Fu, A. Lukaschuk, A. S. Raja, J. Liu, C. D. Wright, A. Sebastian, T. J. Kippenberg, W. H. P. Pernice and H. Bhaskaran, "Parallel convolutional processing using an integrated photonic tensor core," *Nature,* vol. 589, pp. 52-58, 2021.

[266] M. Wuttig, H. Bhaskaran and T. Taubner, "Phase-change materials for non-volatile photonic applications," *Nature Photonics,* vol. 11, pp. 465-476, 2017.

[267] A. Cabrini, S. Braga, A. Manetto and G. Torelli, "Voltage-driven multilevel programming in phase change memories," in *IEEE International Workshop on Memory Technology, Design, and Testing*, 2009.

[268] F. Bedeschi, R. Fackenthal, C. Resta, E. M. Donze, M. Jagasivamani, E. Buda, F. Pellizzer, D. Chow, A. Cabrini, G. M. A. Cavi, R. Faravelli, A. Fantini, G. Torelli, D. Mills, R. Gastaldi and G. Casagrande, "A multi-level-cell bipolar selected phase-change memory," in *IEEE International Solid-State Circuits Conference*, 2008.

[269] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y.-J. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee and Y.-T. Lee, "A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth," in *IEEE International Solid-State Circuits Conference*, 2012.

[270] D. Loke, T. H. Lee, W. J. Wang, L. P. Shri, R. Zhao, Y. C. Yeo, T. C. Chong and S. R. Elliott, "Breaking the speed limits of phase-change memory," *Science,* vol. 336, no. 6088, pp. 1566-1569, 2012.

[271] A. Chen, "A review of emerging non-volatile memory (NVM) technologies and applications," *Solid-State Electronics,* vol. 125, pp. 25-38, 2016.

[272] A. Pirovano, A. L. Lacaita, A. Benvenuti, F. Pellizzer and R. Bez, "Electronic switching in phase-change memories," *IEEE Transactions on Electron Devices ,* vol. 51, no. 3, pp. 452-459, 2004.

[273] I. Thakkar and S. Pasricha, "DyPhase: A Dynamic Phase Change Memory Architecture with Symmetric Write Latency and Restorable Endurance," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 37, no. 9, pp. 1760-1774, 2018.

[274] A. Narayan, Y. Thonnart, P. Vivet, A. Coskun and A. Joshi, "Architecting Optically Controlled Phase Change Memory," *ACM Transactions on Architecture and Code Optimization,* vol. 19, no. 4, pp. 1-26, 2022.

[275] C. Rios, M. Stegmaier, P. Hosseini, D. Wang, T. Scherer, C. D. Wright, H. Bhaskaran and W. H. P. Pernice, "Integrated all-photonic non-volatile multi-level memory," *Nature Photonics,* vol. 9, pp. 725-732, 2015.

[276] X. Li, N. Youngblood, C. Rios, Z. Chen, C. D. Wright, W. H. P. Penrice and H. Bhaskaran, "Fast and reliable storage using a 5 bit, nonvolatile photonic memory cell," *Optica,* vol. 6, no. 1, pp. 1-6, 2019.

[277] J. Feldmann, N. Youngblood, X. Li, C. D. Wright, H. Bhaskaran and W. H. P. Pernice, "Integrated 256 cell photonic phase-change memory with 512-bit capacity," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 26, no. 2, 2020.

[278] A. Shafiee, B. Charbonnier, S. Pasricha and M. Nikdast, "Design Space Exploration for PCM-based Photonic Memory," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2023.

[279] Y. Wang, J. Ning, L. Lu, M. Bosman and R. E. Simpson, "A scheme for simulating multi-level phase change photonics materials," *NPJ Computational Materials,* vol. 7, 2021.

[280] A. W. Poon, X. Luo, F. Xu and H. Chen, "Cascaded microresonator-based matrix switch for silicon on-chip optical interconnection," *Proceedings of the IEEE,* vol. 97, no. 7, pp. 1216-1238, 2009.

[281] P. Guo, N. Zhou, W. Hou and L. Guo, "StarLight: a photonic neural network accelerator featuring a hybrid mode-wavelength division multiplexing and photonic nonvolatile memory," *Optics Express,* vol. 30, no. 20, pp. 37051-37065, 2022.

[282] T. Lin, K. A. Williams, R. V. Penty, I. H. White, M. Glick and D. AcAuley, "Performance and scalability of a single-stage SOA switch for 10/spl times/10 gb/s wavelength striped packet routing," *IEEE Photonics Technology Letters,* vol. 18, no. 5, pp. 691-693, 2006.

[283] M. Poremba, T. Zhang and Y. Xie, "NVMain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems," *IEEE Computer Architecture Letters,* vol. 14, no. 2, pp. 140-143, 2015.

[284] The Standard Performance Evaluation Corporation (SPEC®), "SPEC's Benchmarks and Tools," 2017. [Online]. Available: https://www.spec.org/benchmarks.html. [Accessed 13 4 2024].

[285] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. W. Holzwarth, M. A. Popovic, H. Li, H. I. Smith, J. L. Hoyt, F. X. Kartner, R. J. Ram, V. Stojanovic and K. Asanovic, "Building many-core processor-to- DRAM networks with monolithic CMOS silicon photonics," *IEEE Micro,* vol. 29, no. 4, pp. 8-21, 2009.

[286] M. R. Yahya, N. Wu, Z. Fang, F. Ge and M. S. Shah, "A Low Insertion Loss 5x5 Optical Router for Mesh Photonic Network-on-Chip Topology," in *IEEE Conference on Sustainable Utilization and Development in Engineering and Technologies (CSUDET)*, 2019.

[287] L. Zhang, S. Hong, Y. Xie and D. Dai, "New-generation silicon photonics beyond the singlemode regime," in *Asia Communications and Photonics Conference*, 2021.

[288] M. Bahadori, M. Nikdast, Q. Cheng and K. Bergman, "Universal design of waveguide bends in silicon-on-insulator photonics platform," *IEEE Journal of Lightwave Technology,* vol. 37, no. 13, pp. 3044-3054, 2019.

[289] H. Zhu, J. Gu, H. Wang, Z. Jiang, Z. Zhang, R. Tang, C. Feng, S. Han, R. T. Chen and D. Z. Pan, "DOTA: A Dynamically-operated Optically-interconnected Photonic Transformer Accelerator," in *arXiv:2305.19533*, 2023.

[290] N. Patwardhan, S. marrone and C. Sonsone, "Transformers in the Real World: A Survey on NLP Applications," *Information,* vol. 14, no. 4, 2023.

[291] Q. An, S. Rahman, J. Zhou and J. J. Kang, "A Comprehensive Review on Machine Learning in Healthcare Industry: Classification, Restrictions, Opportunities and Challenges," *Sensors,* vol. 23, no. 9, 2023.

[292] Z. Cao, K. Jiang, W. Zhou, S. Xu, H. Peng and D. Yang, "Continuous improvement of self-driving cars using dynamic confidence-aware reinforcement learning," *Nature Machine Intelligence,* vol. 5, pp. 145-158, 2023.

[293] S.-L. Lu, T. Karnik, G. Srinivasa, K.-Y. Chao, D. Carmean and J. Held, "Scaling the Memory Wall," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012.

[294] K. Khan, S. Pasricha and R. G. Kim, "A Survey of Resource Management for Processing-in-Memory and Near-Memory Processing Architectures," *Journal of Low Power Electronics Applications,* vol. 10, no. 4, 2020.

[295] M. He, C. Song, I. Kim, C. Jeong, S. kim, I. Park, M. Thottethodi and T. N. Vijayakumar, "Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020.

[296] S. Roy, M. Ali and A. Raghunathan, "PIM-DRAM: Accelerating Machine Learning Workloads Using Processing in Commodity DRAM," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS),* vol. 11, no. 4, pp. 701-710, 2021.

[297] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang and Y. Xie, "Prime: a novel processing-in-memory architecture for neural network computation in ReRam-based main memory," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016.

[298] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachen, M. Hu and R. S. Williams, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016.

[299] S. Jain, A. Ranjan, K. Roy and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic ram," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI),* vol. 26, no. 3, pp. 470-483, 2018.

[300] B. C. Lee, E. Ipek, O. Mutlu and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *ACM international symposium on Computer architecture (ISCA)*, 2009.

[301] P. Chi, S. Li, Y. Cheng, Y. Lu, S. H. Kang and Y. Xie, "Architecture design with STT-RAM: opportunities and challenges," in *IEEE Asia and South Pacific Design and Automation Conference (ASPDAC)*, 2016.

[302] G. Yang, C. Demirkiran, Z. E. Kizilates, C. A. R. Ocampo, A. K. Coskun and A. Joshi, "Processing-in-Memory Using Optically-Addressed Phase Change Memory," in *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2023.

[303] M. M. Masnad, G. Zhang, D.-X. Xu, Y. Grinberg and O. Liboiron-Ladouceur, "Fabrication error tolerant broadband mode converters and their working principles," *Optics Express,* vol. 30, no. 14, pp. 25817-25829, 2022.

[304] H. Xu, D. Dai and Y. Shi, "Silicon integrated nanophotonic devices for on-chip multi-mode interconnects," *Applied Sciences,* vol. 10, no. 8, 2020.

[305] C. Li, D. Liu and D. Dai, "Multimode silicon photonics," *Nanophotonics,* vol. 8, no. 2, 2018.

[306] LumOpt, [Online]. Available: https://github.com/chriskeraly/lumopt.git. [Accessed 13 4 2024].

[307] Z. Lu, D. Celo, P. Dumias, E. Bernier and L. Chrostowski, "Comparison of Photonic 2×2 3-dB Couplers for 220 nm Silicon-on-Insulator Platforms," in *IEEE International Conference on Group IV Photonics (GFP)*, 2015.

[308] Z. Fang, R. Chen, J. Zhang, A. I. Khan, K. M. Neilson, S. J. Geiger, D. M. Callahan, M. G. Moebius, A. Saxena, M. E. Chen, C. Rios, J. Hu, E. pop and A. Majumdar, "Ultra-low-energy programmable non-volatile silicon photonics based on phase-change materials with graphene heaters," *Nature Nanotechnology,* vol. 17, pp. 842-848, 2022.

[309] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014.

[310] D. Li, X. Zhao, Y. Shen, S. Liu and Z. Zhu, "A 7-bit 3.8-GS/s 2-Way Time-Interleaved 4-bit/Cycle SAR ADC 16× Time-Domain Interpolation in 28-nm CMOS," *IEEE Transactions on Circuits and Systems,* vol. 70, no. 9, pp. 3557-3566, 2023.

[311] T. O. Dickson, Z. T. Deniz, M. Cochet, T. J. Beukema, M. Kossel, T. Morf, Y.-H. Choi, P. A. Francese, M. Brandli, C. W. Baks, J. E. Proesel, J. F. Bulzacchelli, M. P. Beakes, B.-J. Yoo, H. Ahn, D.-H. Lim, G. Kang, S.-H. Park, M. Meghelli and H. G. Rhew, "A 72-GS/s, 8-Bit DAC-Based Wireline Transmitter in 4-nm FinFET CMOS for 200+ Gb/s Serial Links," *IEEE Journal of Solid-State Circuits,* vol. 58, no. 4, pp. 1074-1086, 2023.

[312] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan and Y. Xie, "DRISA: a DRAM-based Reconfigurable In-Situ Accelerator," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017.