2024

Building next-generation deep learning hardware using photonic computing

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

BUILDING NEXT-GENERATION DEEP LEARNING HARDWARE USING PHOTONIC COMPUTING

by

CANSU DEMIRKIRAN

B.Sc., Middle East Technical University, Ankara, Turkey, 2019

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2024

Approved by

First Reader

Ajay Joshi, PhD
Professor of Electrical and Computer Engineering

Second Reader

Ayse Coskun, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering

Third Reader

Darius Bunandar, PhD
Chief Scientist
Lightmatter

Fourth Reader

Martin Herbordt, PhD
Professor of Electrical and Computer Engineering

Life is short,
and the art long,
opportunity fleeting,
experiment treacherous,
and judgment difficult.

– Hippocrates

Acknowledgments

First, I would like to express my deepest appreciation and gratitude to my advisor, Prof. Ajay Joshi. His continuous support and guidance throughout my PhD journey have shaped my academic path and always pushed me to achieve more.

I am also deeply thankful to Dr. Darius Bunandar for his invaluable advice and mentorship over the years. I feel incredibly fortunate to have had the opportunity to work directly with him at Lightmatter from the early years of my PhD and for our continued collaboration.

In addition, I extend my gratitude to the rest of my thesis committee members, Prof. Ayse Coskun and Prof. Martin Herbordt, for their precious time, generous support, and insightful feedback.

I am very grateful to all my collaborators and friends from Boston University and Lightmatter. Their contributions have been invaluable to my work.

Furthermore, I greatly appreciate the invaluable friendships I have built over the years in Boston. These friendships have made me feel loved, appreciated, and at home in a foreign country, helping me navigate the hardships of my PhD journey. I would especially like to thank my amazing roommate and all-time cheerleader, Isabel, and my dearest friends Agnieszka, Aleksandra, Ania, Genevieve, and many others for their generous friendships.

I also want to thank Tahir, Emre, Ekin, Malik, Yekta, Deniz, and many other friends and family in Turkey who have kept in touch and supported me despite the distance. Finally, I would like to thank my parents, Aliye and Serdar, for their unconditional love and support.

BUILDING NEXT-GENERATION DEEP LEARNING HARDWARE USING PHOTONIC COMPUTING

CANSU DEMIRKIRAN

Boston University, College of Engineering, 2024

Major Professor: Ajay Joshi, PhD
Professor of Electrical and Computer Engineering

ABSTRACT

In recent years, the demand for computational power has skyrocketed due to the rapid advancement of artificial intelligence (AI). As we move past Moore's Law, the limitations of traditional digital computing are pushing the exploration of alternative computing paradigms. Among the emerging technologies, integrated photonics stands out as a highly promising candidate for the next generation of high-performance AI computing as it offers low latency, high bandwidth, and high parallelism. However, there still exist challenges associated with photonic hardware for AI acceleration including the need for slower and less efficient electronic circuits and memory units, lack of efficient nonlinearity in photonics, limited precision, analog noise, and various device non-idealities. In this thesis, we investigate the opportunities and challenges of photonics technology for accelerating state-of-the-art AI workloads from a realistic perspective, evaluate the performance benefits, and propose solutions to address the associated challenges.

First, we outline our strategy for designing and evaluating ADEPT, a complete electro-photonic accelerator for deep neural network (DNN) inference. ADEPT leverages a photonic computing unit for general matrix-matrix multiplication (GEMM) operations, a vectorized digital electronic application-specific integrated circuit (ASIC) for non-GEMM op-

erations, and static random-access memory (SRAM) arrays for storing DNN parameters and activations. Unlike previous photonic DNN accelerators, we adopt a system-level perspective to provide a more realistic assessment of the photonics technology and its applicability in accelerating state-of-the-art DNNs. We detail our design steps and introduce optimizations to minimize the overhead of electronic devices. Our evaluation shows that ADEPT achieves, on average, 5.73× higher throughput per watt compared to systolic arrays (SAs), and more than 6.8× and 2.5× better throughput per watt compared to state-of-the-art electronic and photonic accelerators, respectively.

Second, we focus on the precision limitations in analog computing and propose using the residue number system (RNS) to compose high-precision operations from multiple low-precision operations. This approach eliminates the need for high-precision data converters and avoids information loss. Our study shows that our technology-agnostic RNS-based approach can achieve ≥99% of 32-bit floating-point (FP32) accuracy for state-of-the-art DNN inference with only 6-bit and training with 7-bit fixed-point (FXP) arithmetic. This indicates that using RNS can significantly reduce the energy consumption of analog accelerators while maintaining the same throughput and precision. In addition, we present a fault-tolerant dataflow using redundant RNS (RRNS) to protect computations against noise and errors inherent in analog hardware.

At last, leveraging this RNS-based framework, we propose Mirage, a photonic DNN training accelerator. Mirage employs a novel micro-architecture to support modular arithmetic in the analog domain, achieving high energy efficiency without compromising precision. Our study shows that, on average, Mirage achieves FP32 accuracy with 23.8× lower training time and 32.1× lower energy-delay product (EDP) in an iso-energy scenario, and 42.8× less power consumption with comparable or better EDP in an iso-area scenario, compared to SAs.

Contents

List of Tables

List of Figures

List of Abbreviations

ADC	.....	Analog-to-Digital Converter
AI	.....	Artificial Intelligence
ASIC	.....	Application-Specific Integrated Circuit
A-to-D	.....	Analog-to-Digital
BFP	.....	Block Floating Point
BF16	.....	Brain Floating Point 16-bit
BMM	.....	Batched Matrix Multiplication
BNS	.....	Block Number System
CiM	.....	Compute in Memory
CMOS	.....	Complementary Metal-Oxide-Semiconductor
CNN	.....	Convolutional Neural Network
CPU	.....	Central Processing Unit
CRT	.....	Chinese Remainder Theorem
DAC	.....	Digital-to-Analog Converter
DAG	.....	Directed Acyclic Graph
DNN	.....	Deep Neural Network
D^2NN	.....	Diffractive Neural Networks
DOE	.....	Diffractive Optical Element
DPU	.....	Digital Processing Unit
DRAM	.....	Dynamic RAM
D-to-A	.....	Digital-to-Analog
ECRAM	.....	Electrochemical Random Access Memory
EDP	.....	Energy-Delay Product
EM	.....	Electromagnetic
E-to-O	.....	Electrical-to-Optical
FC	.....	Fully Connected
FeFET	.....	Ferroelectric Field-Effect Transistor
FHE	.....	Fully Homomorphic Encryption
FoM	.....	Figure of Merit
FP	.....	Floating Point
FP16	.....	16-bit Floating Point
FP32	.....	32-bit Floating Point

FPGA	.....	Field-Programmable Gate Array
FXP	.....	Fixed Point
GELU	.....	Gaussian Error Linear Unit
GEMM	.....	General Matrix Multiplication
Ge-on-Si	.....	Germanium-on-Silicon
GPU	.....	Graphics Processing Unit
GRU	.....	Gated Recurrent Unit
HP	.....	High Precision
IC	.....	Integrated Circuit
IM2COL	.....	Image to Column
INT8	.....	8-bit Integer
IPS	.....	Inferences Per Second
IPS/W	.....	Inferences Per Second per Watt
IS	.....	Input Stationary
LLM	.....	Large Language Model
LLVM	.....	Low-Level Virtual Machine
LP	.....	Low Precision
LSB	.....	Least Significant Bit
LSTM	.....	Long Short-Term Memory
LUT	.....	Look-Up Table
MAC	.....	Multiply-Accumulate
MDPU	.....	Modular Dot Product Unit
MLP	.....	Multi-Layer Perceptron
MMU	.....	Modular Multiplication Unit
MMVMU	.....	Modular MVM Unit
MOEMS	.....	Micro-Opto-Electro-Mechanical Systems
MRR	.....	Microring Resonator
MVM	.....	Matrix-Vector Multiplication
MZI	.....	Mach-Zehnder Interferometer
MZM	.....	Mach-Zehnder Modulator
NLP	.....	Natural Language Processing
NOEMS	.....	Nano-Opto-Electro-Mechanical Systems
NVM	.....	Non-Volatile Memory
OP/s	.....	Operations per Second
OS	.....	Output Stationary
O-to-E	.....	Optical-to-Electrical
PCI-e	.....	Peripheral Component Interconnect Express
PDK	.....	Process Design Kit
PDF	.....	Probability Distribution Function
PE	.....	Processing Element

PiM	.....	Processing in Memory
PNS	.....	Positional Number System
QAT	.....	Quantization-Aware Training
RAM	.....	Random-Access Memory
ReLU	.....	Rectified Linear Unit
RNN	.....	Recurrent Neural Network
RNS	.....	Residue Number System
RRAM	.....	Resistive RAM
RRNS	.....	Redundant Residue Number System
RTL	.....	Register Transfer Level
SA	.....	Systolic Array
SAR	.....	Successive Approximation Register
SGD	.....	Stochastic Gradient Descent
SiN	.....	Silicon Nitride
SLM	.....	Spatial Light Modulator
SNR	.....	Signal-to-Noise Ratio
SOA	.....	Semiconductor Optical Amplifier
SOT-MRAM	.....	Spin-Orbit Torque Magnetic RAM
SRAM	.....	Static RAM
STT-MRAM	.....	Spin-Transfer Torque Magnetic RAM
SVD	.....	Singular Value Decomposition
TIA	.....	Transimpedance Amplifier
TOPS	.....	Tera Operations per Second
TPU	.....	Tensor Processing Unit
WDM	.....	Wavelength-Division Multiplexing
WS	.....	Weight Stationary

Chapter 1

Introduction

1.1 Motivation

1.1.1 AI Model Trends

Artificial intelligence (AI) has become pervasive in today's world, influencing numerous aspects of daily life and various industries. Deep neural networks (DNNs) are at the heart of this AI revolution, with a broad spectrum of applications, including language processing (Brown et al., 2020; Team et al., 2023; Touvron et al., 2023; Devlin et al., 2018; Zhang et al., 2022a), computer vision (He et al., 2016; Redmon et al., 2016), healthcare (Rajpurkar et al., 2017; Shen et al., 2019), coding (Chen et al., 2021; Feng et al., 2020b; Gupta et al., 2017), audio processing (Amodei et al., 2016; Van Den Oord et al., 2016), robotics (Levine et al., 2018; Mnih et al., 2016), and many others. Figure 1·1 illustrates the progress of AI systems relative to human baselines across nine benchmarks, each corresponding to a different task. This plot demonstrates that AI capabilities have already surpassed human performance, especially in visual tasks.

This progress, however, comes with a significant cost. While the progress trajectory of AI models is influenced by many factors, there is a strikingly consistent correlation between the amount of compute in AI models and their capabilities, which makes compute a quantifiable proxy for measuring progress in AI research (Kaplan et al., 2020; Hestness et al., 2017; Li et al., 2020).

A recent study (Sevilla et al., 2022) explored AI compute trends and divided the history of AI models into three distinct eras, as illustrated in Figure 1·2:

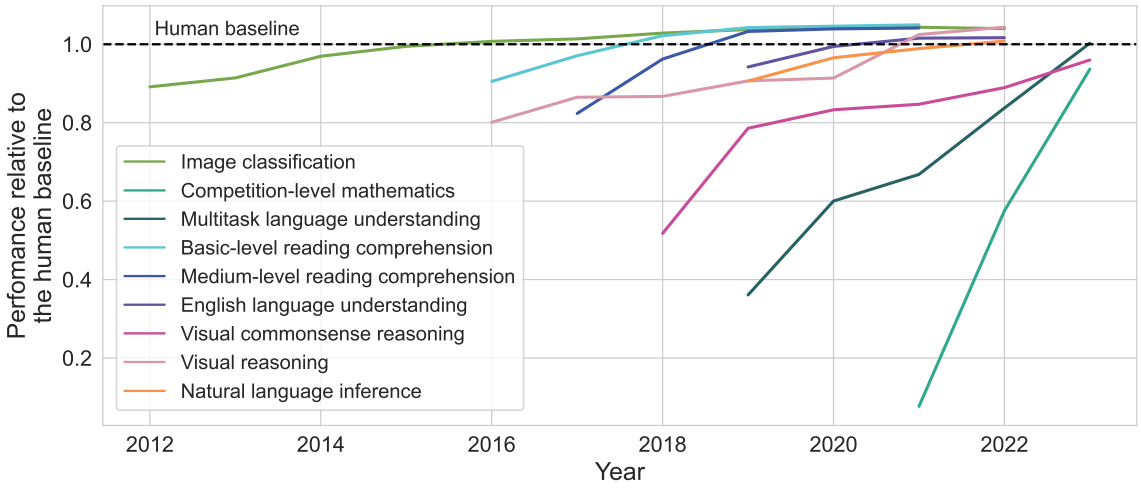

Figure 1·1: Select AI Index technical performance benchmarks vs. human performance, based on data from the 2024 Stanford AI Index Report (Maslej et al., 2024)

- **The Pre-Deep Learning Era:** Before the advent of deep learning, training compute requirements roughly followed Moore's Law, doubling approximately every 20 months.

- **The Deep Learning Era:** Beginning around 2010, there has been a dramatic acceleration in compute requirements, with a doubling time of roughly 6 months.

- **The Large-Scale Era:** Starting from 2015, this era is marked by the development of models that used two-to-three orders of magnitude more compute than those in the Deep Learning Era. However, the compute growth for these large-scale models has a relatively slower doubling time of about 10 months.

Although the doubling time in the Large-Scale Era appears relatively slower, it remains extraordinarily rapid. Figure 1·2 shows the amount of training compute in notable AI models over time. Since 2010, the amount of training compute for AI models has increased by a factor of 10 billion, far outpacing Moore's Law (Sevilla et al., 2022).

The number of parameters is another important metric correlated with AI capabilities,

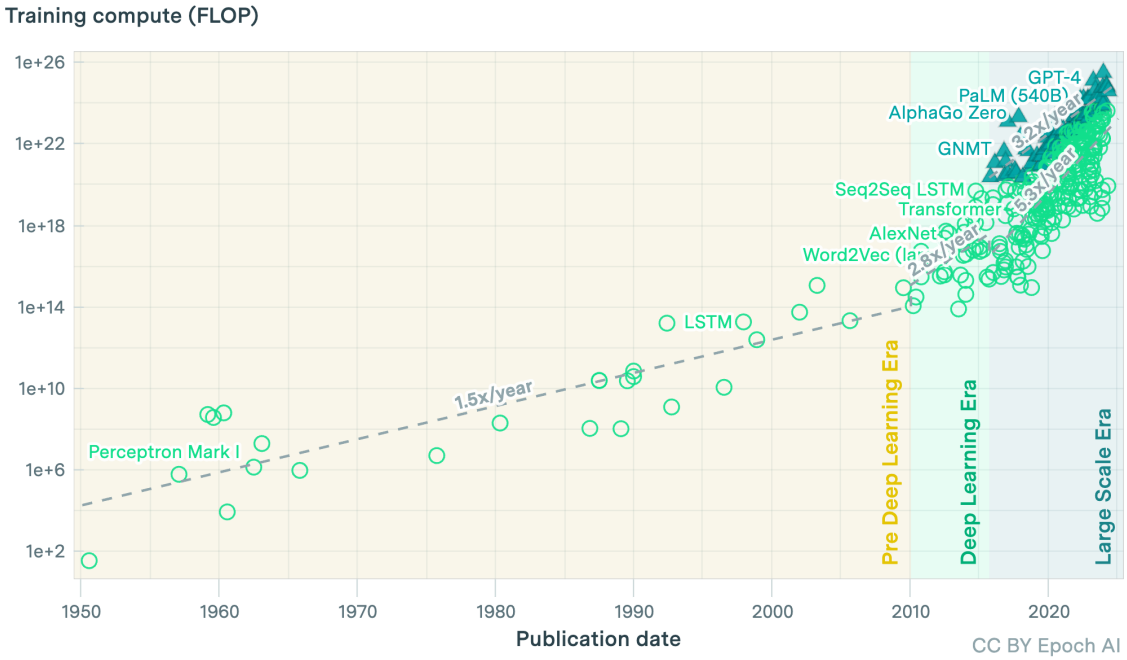

Figure 1·2: Training compute of notable AI models over time. The figure is taken from Epoch AI's 2023 study (Epoch AI, 2023).

which greatly impacts the compute and memory requirements in AI systems. Figure 1·3 shows the increase in the number of trainable parameters in notable AI models over time for the three eras of AI compute. The largest models available today include more than a trillion trainable parameters.

This increasing trend in compute requirements and model sizes of models places significant stress on today's AI systems, necessitating faster compute cores, higher data bandwidths, and improved interconnect speeds. Supporting these large models also results in very high power consumption, significantly escalating their environmental footprint. For instance, Meta's Llama 2 model with 70 billion parameters is reported to release 291.2 tonnes of carbon during training while consuming 400 MWh of power. This carbon emission is nearly 291× more than the emissions from a round-trip flight from New York to San Francisco for one traveler, and roughly 16× more than the annual carbon footprint of an average American (Touvron et al., 2023). Similarly, GPT-3 is reported to consume 1,287

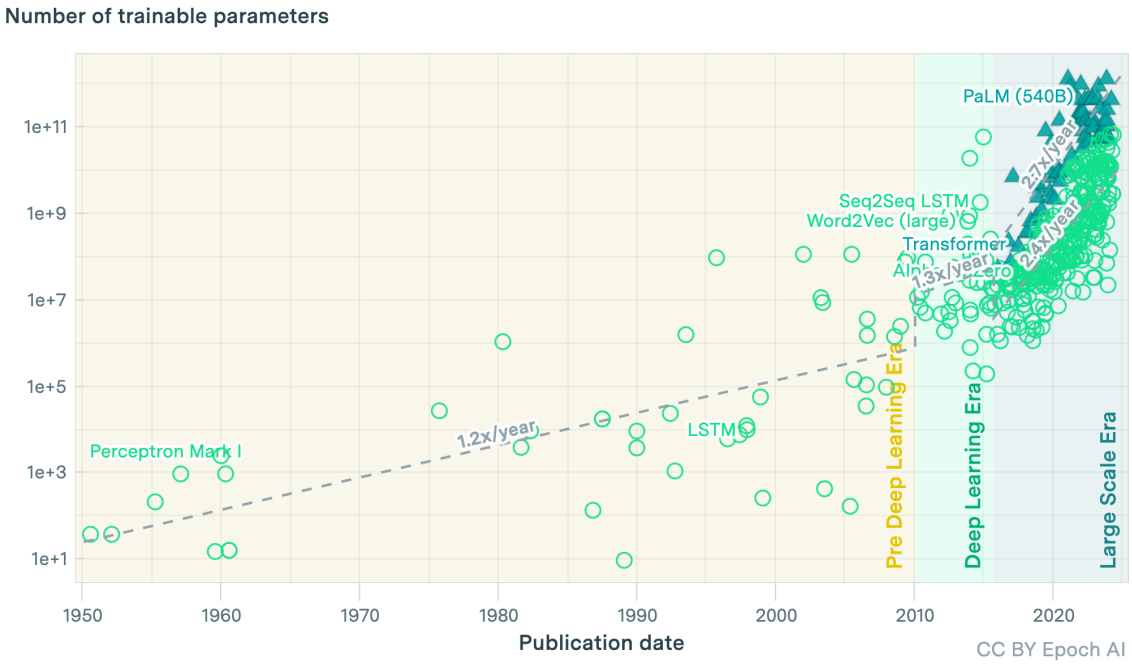

Figure 1·3: Number of trainable parameters in notable AI models over time. The figure is taken from Epoch AI's 2023 study (Epoch AI, 2023).

MWh of power and emit 502 tonnes of carbon (Maslej et al., 2024). While the emissions from individual inference queries may be relatively low, the total impact can surpass that of training when models are queried thousands or even millions of times daily. Although the prominent developers of AI models such as OpenAI, Google, Anthropic, and Mistral do not disclose their carbon footprint, their recent models with over a trillion parameters are expected to go well beyond the aforementioned numbers (Maslej et al., 2024). Hence, beyond speed, improvements in energy efficiency are also crucial.

The current trend of escalating compute and energy requirements and model sizes is unsustainable with existing technology, both in terms of performance and environmental impact. The massive power consumption and carbon emissions associated with training and deploying large-scale models underscore the critical need for innovation and advancements in both hardware and software design. These innovations are crucial to ensure continued progress in AI, driving advancements in numerous fields such as healthcare, education, and

finance.

1.1.2 Technology Scaling and AI Hardware Trends

For decades, processor performance improvements have been guided by Moore's Law, which states that the number of transistors in integrated circuits (ICs) doubles approximately every two years, driven by the continuous reduction in transistor dimensions through advancements in complementary metal-oxide-semiconductor (CMOS) technology. Alongside Moore's Law, Dennard scaling (Bohr, 2007) suggested that as transistors become smaller, their power consumption decreases proportionally to their area. However, as transistor sizes reached the nanometer scale, this power scaling decoupled from the device area due to physical constraints, leading to increased power consumption per unit area (Gargini, 2023). This phenomenon created the so-called *power wall*, limiting the increase in clock frequencies and posing a significant challenge to performance scaling. The rising transistor densities given a power budget led to dark silicon (Esmaeilzadeh et al., 2011), where parts of the IC are powered down to manage heat and power consumption. Techniques like dynamic voltage and frequency scaling (Le Sueur and Heiser, 2010) can offer partial mitigation, but the root problem persists at the device level.

Despite the power wall impeding performance scaling, the demand for high-performance execution of complex workloads continued to grow. In response to the breakdown of Dennard scaling, architects increased the number of logical cores in general-purpose processors and exploited parallelism at various levels. However, the benefits of large core counts are constrained by memory bottlenecks in a Von Neumann execution model, even when the workload offers a high degree of parallelism.

To boost performance further, architects have adopted heterogeneous computing frameworks, offloading critical tasks to hardware accelerators. These specialized co-processors, tailored for specific applications, outperform general-purpose processors in terms of throughput and energy efficiency. By bypassing the limitations of the Von Neumann architec-

ture, accelerators achieve high levels of parallelism and concurrency, effectively mitigating memory bottlenecks and enhancing compute density and energy efficiency. Today, modern AI hardware largely depends on these accelerators, such as graphic processing units (GPUs) and application-specific integrated circuits (ASICs).

A recent study (Hobbhahn et al., 2023) reports that GPUs have seen an energy efficiency doubling time of around 2.7 years over the past 15 years when using the 32-bit floating point (FP32) format. Moreover, adopting more efficient data formats and lower numeric precision has remarkably enhanced AI workload performance in recent years. For instance, NVIDIA's H100 GPU achieves speedups of approximately 7× with tensor-FP32, 15× with tensor-FP16, and 30× with 8-bit integer (INT8) compared to FP32 (Hobbhahn et al., 2023). Today, the highest-performing GPU, NVIDIA B200 SXM, has a 2.25e15 operations per second (OP/s) peak throughput in tensor-FP16 and 4.5e15 OP/s in INT8 (NVIDIA Corporation, 2024b).

ASICs like Google's Tensor Processing Unit (TPU), on the other hand, typically rely on systolic array (SA)-based architectures. SAs involve an array of processing elements (PEs), each connected to its neighbors. Each PE has a multiply-accumulate (MAC) unit along with local buffers to store inputs and outputs. Depending on the dataflow, data moves from one side of the SA to the other as MAC operations are performed. This structure, which has been used in many AI accelerators (Chen et al., 2016; Chen et al., 2019; Lee et al., 2018; Jouppi et al., 2017), enables data reuse and maximizes the energy efficiency in general matrix-matrix multiplication (GEMM) operations. Today, the most recent TPU (vp5) has a 4.59e14 OP/s peak throughput in Brainfloat-16 (BF16) (Google Cloud, 2024).

In recent years, there have been remarkable advancements in GPU and ASIC designs, maintaining an exponential trend in their performance improvements. However, these gains are largely due to specialization, the adoption of efficient data formats, and hardware-software co-design rather than technology scaling. As the benefits of specialization begin

to saturate and CMOS technology approaches its physical limits, performance improvements in CMOS-based hardware are expected to plateau. Given the rapid growth of AI workloads, engineers and researchers must seek alternative solutions to sustain the growth in AI capabilities.

1.1.3 Integrated Photonics as a Computing Platform for AI

The idea of computing with light is not new and has been explored starting from the 1960s (Reimann and Kosonocky, 1965). Replacing traditional CMOS-based transistors with optical transistors has been an appealing idea because of the high speed, bandwidth, and unique parallelism opportunities that light signals inherently offer. As a result, free-space optical computers have remained a major research interest over many decades (Bell, 1986; Caulfield, 1987; Hsu et al., 1988; Bernstein et al., 2021). However, these optical computers have not been widely adopted due to challenges associated with optical alignment, precision control over the phase and amplitude of the light field, and the limited component count that could not compete with digital electronics.

The advent of integrated photonics has largely solved the problem of controlling and stabilizing many photonic components, offering greater compactness for high integration compared to other photonic computing schemes. One particular flavor of integrated photonics, silicon photonics, has seen widespread integration in commercial CMOS foundries alongside CMOS transistors (Giewont et al., 2019). Validated process design kits (PDKs) developed by various foundries have pushed integrated photonics towards standardization, ensuring reliable performance and accessibility for designers and users (Ning et al., 2024). Consequently, many recent works leverage the highly parallel and efficient linear transformations enabled by silicon photonics to develop specialized DNN accelerators. These accelerators have demonstrated promising results, with orders of magnitude improvements in speed and energy efficiency compared to their electrical counterparts (Shen et al., 2017; Shiflett et al., 2021; Bangari et al., 2019; Shiflett et al., 2020; Peng et al., 2020; Wu et al.,

2021; Wetzstein et al., 2020; Shastri et al., 2021). Despite these promising results, several challenges remain in building photonic AI systems with current technology. We categorize the primary issues under two groups:

1. **The need for electronic components in photonic DNN acceleration:** The limitations of weak photon-photon nonlinearities and the lack of photonic information storage present significant challenges in designing practical general-purpose optical computing systems. The lack of efficient nonlinearity in photonics has led researchers to focus on developing specialized photonic units for GEMM operations, which dominate DNN workloads. However, electronic circuitry is still necessary for performing nonlinear operations, control tasks, and data storage. Moreover, each photonic operation results in some power loss in the optical signal. As the number of consecutive photonic operations increases, the required input power grows exponentially, fundamentally limiting the number of operations that can be performed consecutively, even during linear operations. This limitation necessitates the storage of intermediate data, which is not feasible in optical form alone. Consequently, conversions between digital and analog domains are required, along with the use of traditional memory units like static random access memory (SRAM) and dynamic random access memory (DRAM). Frequent digital-to-analog (D-to-A) and analog-to-digital (A-to-D) conversions, along with electrical-to-optical (E-to-O) and optical-to-electrical (O-to-E) conversions, combined with electronic arithmetic and memory operations, can create bottlenecks and significantly degrade the system's energy efficiency.

2. **Limited numeric precision in photonics:** In the analog domain, data are encoded in physical properties such as amplitude or phase, unlike the multiple bits used in the digital domain. This restricts analog operations to using only fixed-point (FXP) arithmetic, which offers a much lower dynamic range than floating-point (FP) arithmetic

for the same bit-width. Besides the absence of FP arithmetic, analog FXP operations present unique precision challenges in photonic tensor cores. The achievable precision in photonic hardware is limited by the signal-to-noise ratio (SNR) during analog operations and the bit precision of the digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). Photonics suffers from multiple noise sources and non-idealities, including process variations, device noise, and environmental factors. Achieving high SNR is challenging because it requires exponentially increasing the input power with increasing bit-width to suppress noise. Moreover, the energy consumption of DACs and ADCs increases exponentially with their bit precision, making high-precision data converters impractical and limiting the input and output precision in analog operations. These limitations in numeric precision can cause significant accuracy loss in photonic hardware, even in DNN inference, especially in large state-of-the-art DNNs tackling complex problems.

1.2 Thesis Contribution

The abovementioned challenges in photonic computing have limited its applicability to only very small DNNs and outdated tasks. While prior research has laid the groundwork for photonic AI hardware and shown promising results, there is a lack of detailed analysis to understand the technology's realistic potential and a clear roadmap for developing a hybrid electro-photonic accelerator capable of deploying state-of-the-art DNNs. Furthermore, the limited precision of photonic hardware has often been overlooked, as high precision was not required for the simple workloads evaluated in earlier studies. However, state-of-the-art DNNs demand higher precision to maintain accuracy, even for inference tasks. DNN training, in particular, has remained a distant goal for photonics, with efforts confined to a few studies focused on simple tasks and small models.

This thesis explores the opportunities and challenges of photonic computing for deep

learning acceleration with a pragmatic approach, to offer solutions to the abovementioned challenges. It is divided into three parts as follows:

1. **Architecting a complete electro-photonic system for DNN inference:** In this part, we design and evaluate a full electro-photonic accelerator, ADEPT, which leverages a photonic computing unit for performing GEMM operations, a vectorized digital electronic ASIC for performing non-GEMM operations, and SRAM arrays for storing DNN parameters and activations. In contrast to prior works in photonic DNN accelerators, we adopt a system-level perspective and show that the gains, while large, are tempered relative to prior expectations. Our goal is to encourage architects to explore photonic technology in a more pragmatic way considering the system as a whole to understand its general applicability in accelerating today's DNNs. We discuss the design steps and optimizations to minimize the overhead of electronic devices. Our evaluation shows that ADEPT can provide, on average, 5.73$\times$ higher throughput per watt compared to the traditional SAs in a full system, and at least 6.8$\times$ and 2.5$\times$ better throughput per watt, compared to state-of-the-art electronic and photonic accelerators, respectively.

2. **Unlocking high-precision in analog tensor cores:** In this study, we target the numeric precision challenge in analog computing and propose a solution based on the residue number system (RNS). RNS can help compose high-precision operations from multiple low-precision operations and eliminate the need for high-precision data converters and information loss. Our study shows that the RNS-based approach can achieve $\geq$ 99% of FP32 accuracy in state-of-the-art DNN inference using only 6-bit and training with 7-bit FXP arithmetic. These results imply that using RNS can reduce the energy consumption of analog accelerators with the same precision by several orders of magnitude while maintaining the same throughput. In addition, we present a fault-tolerant dataflow using redundant RNS (RRNS) to protect the compu-

tation against noise and errors inherent within analog hardware.

3. **High-precision photonic DNN training accelerator design:** In this part, we present a photonic DNN training accelerator, Mirage, based on the RNS-based framework introduced in the second part. Mirage employs a novel micro-architecture to support the RNS-based dataflow and modular arithmetic in the analog domain. By combining RNS and photonics, Mirage provides high energy efficiency without compromising precision and can successfully train state-of-the-art DNNs achieving accuracy comparable to FP32 training. Our study shows that on average across several DNNs when compared to SAs, Mirage achieves more than 23.8× faster training and 32.1× lower energy-delay product (EDP) in an iso-energy scenario and consumes 42.8× lower power with comparable or better EDP in an iso-area scenario.

1.3 Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review and provide a background on DNN basics, photonic devices in photonic tensor cores, RNS, and related work on photonic DNN acceleration. Chapter 3 presents the design and evaluation of our electro-photonic accelerator, ADEPT. In Chapter 4, we introduce our RNS-based framework for analog tensor cores to overcome precision challenges in analog computing. Chapter 5 presents our photonic DNN training accelerator, Mirage, that supports the RNS-based framework introduced in Chapter 4. In Chapter 6, we summarize the completed work and discuss several future directions.

Chapter 2

Background and Related Work

In this chapter, we provide background on DNNs, silicon photonic devices that are used to perform DNN operations, and related work on photonic DNN accelerators.

2.1 Deep Neural Networks

This section provides information about DNN basics, different types of DNNs, different layers in DNNs, DNN inference and training, and a summary of different data formats used for executing DNNs.

2.1.1 DNN Types

Multi-Layer Perceptrons (MLPs)

MLPs are the simplest form of DNNs. They include an input layer, one or more hidden layers, and an output layer, arranged in a feedforward fashion. Each neuron in a layer is connected to every neuron in the subsequent layer without any cycles or loops in the network, making these networks fully connected (FC). Figure 2·1(a) illustrates an example of MLP architecture with a single hidden layer. MLPs consist of FC layers and an activation function to introduce nonlinearity such as rectified linear unit (ReLU), Gaussian error linear unit (GELU), sigmoid, etc. MLPs are typically used for relatively naive classification problems or as part of a larger DNN architecture.

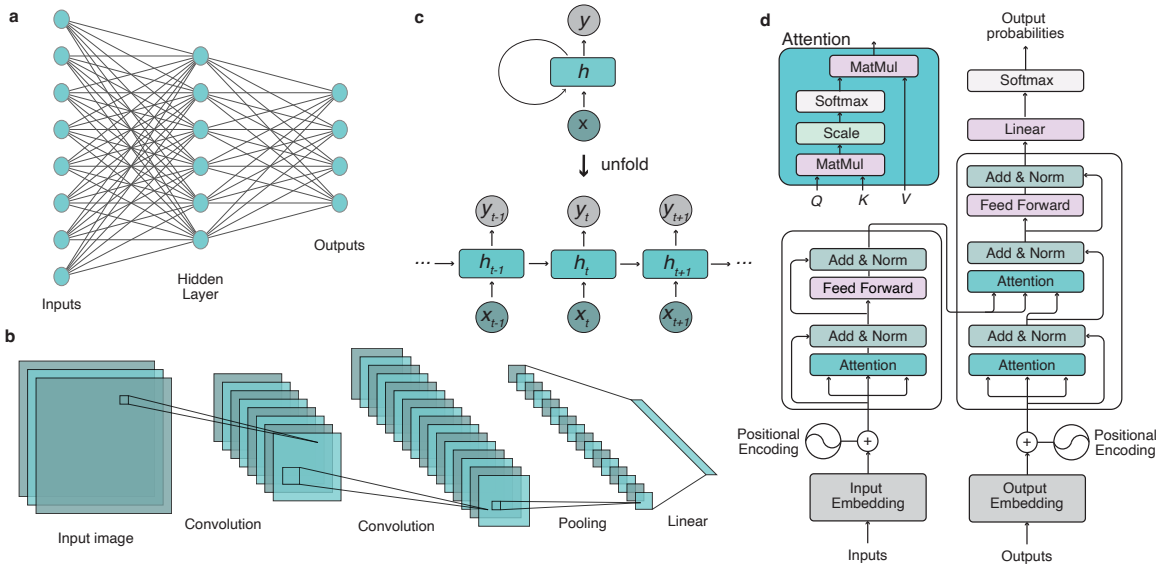

Figure 2·1: (a) MLP architecture. **(b)** CNN architecture. **(c)** RNN architecture. **(d)** Transformer architecture.

Convolutional Neural Networks (CNNs)

CNNs are a class of DNNs specifically designed for processing structured grid data, such as images. They mainly involve convolutional, pooling, and FC layers. CNNs excel in tasks such as image and video recognition, object detection, and segmentation due to their ability to capture the local dependencies in grid-like data structures. Figure 2·1(b) illustrates an example of CNN architecture. Some well-known CNN examples include AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2014), and ResNet (He et al., 2016).

Recurrent Neural Networks (RNNs)

RNNs are a type of DNNs designed for sequential data processing. Unlike feed-forward neural networks, RNNs have connections that form directed cycles, allowing them to maintain a hidden state that captures information about previous inputs, as shown in Figure 2·1(c). This makes RNNs particularly effective for tasks where context and order are important, such as time series prediction, natural language processing (NLP), and speech recognition.

Building on traditional RNNs, Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), which include gating mechanisms to regulate the flow of information, were developed to cope with issues like vanishing and exploding gradients and be able to process longer sequential data.

Transformers

Transformers are a revolutionary type of DNN architecture that was introduced by Vaswani et al. in 2017 (Vaswani et al., 2017). The key component of transformers is the self-attention mechanism, which enables a better understanding of context and relationships within the data and effectively captures long-range dependencies. Transformers are often used in machine translation, text generation, and summarization tasks. Some influential examples include BERT (Devlin et al., 2018) and GPT (Brown et al., 2020), which have set new benchmarks across a wide range of NLP tasks.

2.1.2 Layer Types in DNNs

FC (Dense) Layer

In an FC layer, the input vector to the layer x is transformed through a weight matrix W and a bias vector b to produce the output vector O, i.e., $O = \mathbf{W}\mathbf{x} + \mathbf{b}$. This operation is typically followed by a nonlinear activation function.

Convolution Layer

Convolution involves a set of learnable filters (also known as kernels) that slide over the input data to produce feature maps. Given an input image I of dimensions $I_H \times I_W \times C_{\text{in}}$, where I_H is the height, I_W is the width, and C_{in} is the number of channels (e.g., $C_{\text{in}} = 3$ for RGB images), a kernel K of dimensions $K_H \times K_W \times C_{\text{in}}$, and a stride s, the convolution

operation is performed as:

$$O[i][j] = \sum_{m=0}^{K_H-1} \sum_{n=0}^{K_W-1} \sum_{c=0}^{C-1} (I[si+m][sj+n][c] \cdot K[m][n][c]), \tag{2.1}$$

where $O[i][j]$ is the output feature map at position (i, j). The input images can also be padded by an amount p. For C_{out} kernels, Eq. (2.1) is repeated for each kernel to produce an output with dimensions $O_H \times O_W \times C_{\text{out}}$, where $O_H = \lfloor (I_H + 2p - K_H)/s \rfloor + 1$ and $O_W = \lfloor (I_W + 2p - K_W)/s \rfloor + 1$.

Convolution operation consists of dot products that can easily be rearranged in a GEMM form. This requires pre and post-processing of input and weight matrices. A commonly used pre and post-processing method is image-to-column (im2col) (Anderson et al., 2017). This method transforms the input image into a matrix where each column is a flattened patch of the image that the convolution filter will slide over ($O_H \cdot O_W$ patches per image, one for each output element). Similarly, all kernels are flattened to be multiplied with the input patches. With the im2col technique, convolution operation becomes a GEMM operation between two 2-dimensional matrices with dimensions $K_H K_W C_{\text{in}} \times C_{\text{out}}$ and $O_H O_W \times K_H K_W C_{\text{in}}$.

Recurrent Layer

Here, we will cover the LSTM layer as it is widely adopted in various architectures (He et al., 2019; Wu et al., 2016). Other recurrent layer types, i.e., RNNs and GRUs have a similar structure with only show slight differences. An LSTM layer consists of three gates: the *input gate* controlling how much of the new input should influence the hidden state, the *forget gate* deciding how much of the previous hidden state should be retained, and the *output gate* determining how much of the hidden state should be outputted. The equations for an LSTM cell are stated below:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}), \tag{2.2}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}), \tag{2.3}$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}), \tag{2.4}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}), \tag{2.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{2.6}$$

$$h_t = o_t \odot \tanh(c_t), \tag{2.7}$$

where i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, h_t is the hidden state, c_t is the cell state, and x_t is the input at time t, σ is the sigmoid function, and $\odot$ is the Hadamard product. The learnable weights (to be multiplied with input and hidden states) are denoted as $W_i = (W_{ii}|W_{if}|W_{ig}|W_{io})$ and $W_h = (W_{hi}|W_{hf}|W_{hg}|W_{ho})$, and the biases are denoted as $b_i = (b_{ii}|b_{if}|b_{ig}|b_{io})$ and $b_h = (b_{hi}|b_{hf}|b_{hg}|b_{ho})$. The matrix multiplication operations shown in Eqs. (2.2)-(2.5) can be performed at once as $W_i \times x + b_i$ (for the whole input sequence) and $W_h \times h_t + b_h$ for parallelization. The output for each gate (input, forget, cell, and output) can then be used separately.

Attention Layer

The attention layer is a crucial component in transformers. In an attention layer (see Figure 2·1(d)), three main vectors are computed from the input: the query (Q), the key (K), and the value (V). The attention mechanism computes a weighted sum of the values (V), where the weights are determined by the similarity between the query (Q) and the key (K). Mathematically, this mechanism can be described as follows:

$$O = \text{softmax}(Q \times K^T) \times V, \tag{2.8}$$

where O is the output. This process can be extended to multi-head attention, where multiple sets of queries, keys, and values are used to allow the model to jointly attend to information from multiple representation subspaces.

It is important to note that Q, K, and V, are all derived from the input sequence through learned linear transformations and are computed at runtime, i.e., they are not trained and fixed variables. This makes the GEMM operations in this layer different than the other layers described before (e.g., FC, convolution, and recurrent) where the equation is in the form of $O = W \cdot X$ and W is learnable and fixed during inference.

Pooling Layer

Pooling layers are mostly used in CNNs and perform downsampling operations to reduce the spatial dimensions of the input feature maps while retaining important information. This dimensionality reduction helps reduce the computational load and mitigate overfitting. There are different types of pooling operations such as max pooling where a defined window is reduced to its maximum element and average pooling where a defined window is reduced to the average of its elements.

Normalization Layer

Normalization is used to recenter the distribution of activation values by mapping all values of a feature to be in a particular range. This helps overcome the challenges imposed by exploding activations and fluctuating distributions at the layer's input. There are two main types of normalization: batch normalization where the activations are normalized across the batch, separately for each feature, and layer normalization where the activations are normalized across the feature, separately for each sample. Batch normalization is more common in MLPs and CNNs and is typically fused with the previous FC or convolution layer to reduce the number of operations, whereas layer normalization is frequently used in RNNs and transformers.

2.1.3 DNN Inference and Training

A DNN consists of a sequence of L layers. During inference, where the DNN is previously trained and its parameters are fixed, only a forward pass is performed. Generically, the input X to $(\ell+1)$-th layer of a DNN during the forward pass is the output generated by the previous ℓ-th layer:

$$X^{(\ell+1)} = f^{(\ell)}\big(W^{(\ell)}X^{(\ell)}\big), \tag{2.9}$$

where $O^{(\ell)} = W^{(\ell)}X^{(\ell)}$ is a GEMM operation, $W^{(\ell)}$ is the weight matrix and $f^{(\ell)}(\cdot)$ is the nonlinear function of the (ℓ)-th layer.

DNN training requires both forward and backward passes as well as weight updates. The forward pass in the training is performed the same way as in Eq. (2.9). After the forward pass, a loss value $\mathcal{L}$ is calculated using the output collected in the forward pass and the ground truth. The gradients of the activations and DNN parameters with respect to $\mathcal{L}$ for each layer are calculated by performing a backward pass after each forward pass:

$$\frac{\partial \mathcal{L}}{\partial X^{(\ell)}} = {W^{(\ell)}}^T \frac{\partial \mathcal{L}}{\partial O^{(\ell)}}, \tag{2.10}$$

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial O^{(\ell)}} {X^{(\ell)}}^T. \tag{2.11}$$

Using these gradients $\frac{\partial \mathcal{L}}{\partial W^{(\ell)}}$, i.e., $\Delta W^{(\ell)}$, the DNN parameters are updated in each iteration i as:

$$W_{i+1}^{(\ell)} = W_i^{(\ell)} - \eta \Delta W_i^{(\ell)}, \tag{2.12}$$

using a step size η for the stochastic gradient descent (SGD) optimization algorithm. Essentially, for each layer, one GEMM operation is performed during the forward pass and two GEMM operations are performed during the backward pass.

2.1.4 Data Formats for DNN Execution

The efforts to optimize DNN training or inference mainly revolve around improving the efficiency of MAC operations and matrix multiplications. To accelerate MAC operations while maintaining high accuracy, prior works have proposed various data formats based on integer/FXP and FP arithmetic. In this section, we provide a summary of these different formats.

Integer/FXP Format

Both integer and FXP numbers are represented with a fixed number of bits allocated for integer (and fractional) parts of a number. Integer arithmetic specifically deals with whole numbers, while FXP arithmetic allows for fractional values. In these representations, the gap between adjacent numbers is always equal to the same value.

For example, 8-bit integer arithmetic can only represent numbers between $[0, 2^8 - 1]$, which can also be mapped around zero as $[-(2^7 - 1), 2^7 - 1]$ to represent signed integers. Both formats are more compute-efficient compared to FP arithmetic with the same bit-width. Therefore, scalar quantization and integer/FXP arithmetic have been extensively explored for DNN inference and training (Courbariaux et al., 2014; Banner et al., 2018; Hubara et al., 2017; Zhou et al., 2016; Gupta et al., 2015; Jacob et al., 2018). While being more hardware-friendly, the dynamic range that FXP arithmetic can provide is much more limited compared to FP arithmetic for the same bit-width. This limited range can lead to accuracy degradation, especially in large, state-of-the-art DNNs handling complex tasks. In particular, training requires a wider dynamic range, which typically necessitates the use of FP arithmetic.

FP Format

FP arithmetic represents numbers as a sign bit, mantissa bits, and exponent bits. For example, In the 32-bit IEEE format, the first bit is allocated as the sign bit (S), the next 8 bits are allocated as the exponent field (E), biased by 127 to represent negative exponents, and the last 23 bits are the mantissa bits (M) of the normalized number. This representation is reconstructed as $1.M \times 2^{E-127}$. Due to the exponential field, the gaps between adjacent numbers are not equal, with large gaps between large numbers and small gaps between small numbers. This representation offers a wider range of values and higher precision compared to FXP and integer arithmetic with the same bit-width, however, FP arithmetic is slower and more costly. In addition to FP32 and FP16, which are very commonly used data types, especially for training, special FP formats have been proposed to improve hardware performance. Examples include BFLOAT16 (Wang and Kanwar, 2019), HFP8 (Sun et al., 2019), and TensorFloat (Stosic and Micikevicius, 2021).

Block FP (BFP) Format

BFP format provides a middle ground between FXP and FP formats. BFP format splits tensors into groups and assigns an exponent to each group that is shared by the elements within the group. For each group, the largest exponent among the group elements is chosen to be the shared exponent ($e_{\vec{v}}$). Given an $e_{\vec{v}}$ for a group $\vec{v}$ with a group size g, i.e., $e_{\vec{v}} = \max(e_{\vec{v}[i]})\ \forall i \in \{1, ..., g\}$, the mantissae of the group elements are shifted right by the difference between the shared exponent and their original exponent, i.e., $m_{\vec{v}[i]} = m_{\vec{v}[i]} >> (e_{\vec{v}} - e_{\vec{v}[i]})$, where $\vec{v}[i]$ is the i^{th} element of $\vec{v}$. The least-significant bits (LSBs) of the mantissae are then truncated depending on the number of mantissa bits.

This representation allows integer operations between groups using only sign and mantissa bits while preserving the dynamic range through a shared exponent. For example, when element-wise multiplying the elements of two groups, $\vec{v}_1$ and $\vec{v}_2$, the multiplications

are performed using $m_{\vec{v}_1}$ and $m_{\vec{v}_2}$ as signed integers. Unlike the FP arithmetic, the exponent of g output elements is also shared and is simply obtained by a single add operation, i.e., $e_{\vec{v}_{\text{out}}} = e_{\vec{v}_1} + e_{\vec{v}_2}$.

While this format is more efficient than FP arithmetic, thanks to the exponent shared by a group of elements, a higher dynamic range than FXP arithmetic can be preserved during operations. Prior works leveraged the BFP format for DNN inference (Song et al., 2018; Basumallik et al., 2022; Darvish Rouhani et al., 2020) and training (Drumond et al., 2018; Zhang et al., 2022b) as it is less costly than the FP formats and achieves better accuracy than the FXP formats with the same bit-width.

2.2 Silicon Photonics Devices

In this section, we cover the basics of key devices in silicon photonics and ways to build photonic tensor cores.

2.2.1 Mach Zehnder Interferometers (MZIs)

An MZI is a configurable photonic device that controls the interference of two light beams by adjusting the relative phase shift between the beams. A simple MZI consists of two directional couplers (DCs) and a differential phase-shift in between (see Figure 2·2(a)).

$$U(2) = \begin{bmatrix} \sin\phi & \cos\phi \\ \cos\phi & -\sin\phi \end{bmatrix}, \tag{2.13}$$

where ϕ is the phase difference between the two internal arms of the MZI. A larger rank-N unitary matrix can be achieved by composing these 2×2 structures in a mesh form using either a rectangular (Clements et al., 2016) or triangular (Reck et al., 1994) pattern. The rectangular pattern proposed by (Clements et al., 2016) is more widely adopted as it outperforms the triangular pattern due to its symmetry and reduced optical depth.

Previous works based on MZIs (Shen et al., 2017; Ramey, 2020) are mainly based on

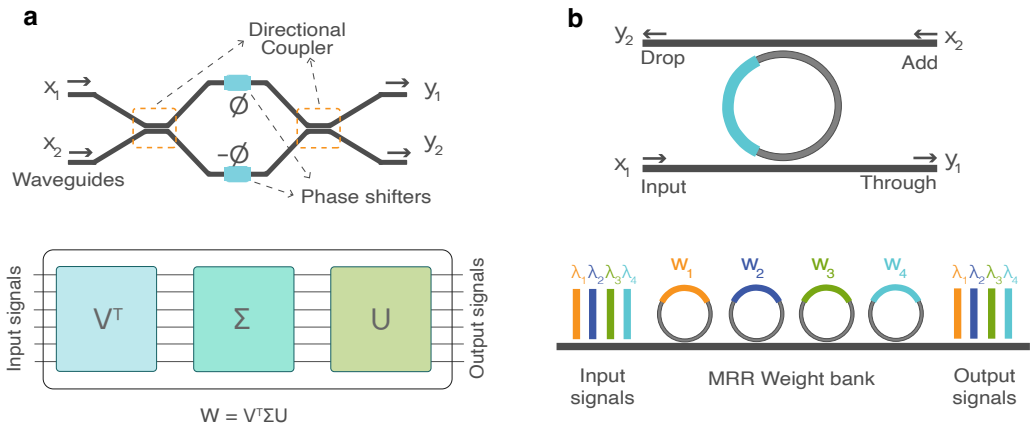

Figure 2·2: **(a)** MZI and MZI-based matrix-vector operation. **(b)** MRR and MRR-based weight bank.

the idea that any real-valued matrix can be decomposed into unitary and diagonal matrices by using singular value decomposition (SVD). A universal linear network can then be composed of two universal unitary circuits and an additional column of attenuators (an MZI with a single input and output) (Miller, 2013). By tuning the phase values of MZIs in this mesh structure, any real-valued matrix can be programmed into the array. A full matrix-vector multiplication (MVM) operation can then be performed by passing modulated input signals through the programmed MZI array.

An MVM between a matrix M and a vector v_{in} is achieved by (1) programming the matrix M in the array of MZIs; (2) encoding the vector v_{in} in the amplitude and phase (0 or π for sign) of the optical signals entering the array; and (3) obtaining the resulting vector $v_{\text{out}} = M \cdot v_{\text{in}}$ at the output of the array. When the vector v_{in} is inserted at GHz rate, a 100×100 array enables us to perform linear operations at 10 Tera Operations per Second (TOPS). Similarly, a full GEMM operation between two matrices can be achieved by encoding one matrix in the MZI array and by sending the other matrix through the array as modulated optical signals—one vector at a time.

2.2.2 Microring Resonators (MRRs)

MRRs are widely employed devices in photonic tensor cores because of their compactness and wavelength-dependent transmission characteristics allowing wavelength division multiplexing (WDM). An add-drop MRR, illustrated in Figure 2·2(b), has four optical ports where the microring is placed between two waveguides. The transfer characteristics (Ning et al., 2024) for the drop and through ports can be mathematically represented by the following equations:

$$T_{\text{through}} = \left|\frac{E_t}{E_{\text{in}}}\right|^2 = \frac{t_1^2 + t_2^2\alpha_{rt}^2 - 2t_1t_2\alpha_{rt}\cos\phi_{rt}}{1 + t_1^2t_2^2\alpha_{rt}^2 - 2t_1t_2\alpha_{rt}\cos\phi_{rt}} \tag{2.14}$$

$$T_{\text{drop}} = \left|\frac{E_d}{E_{\text{in}}}\right|^2 = \frac{\kappa_1^2\kappa_2^2\alpha_{rt}}{1 + t_1^2t_2^2\alpha_{rt}^2 - 2t_1t_2\alpha_{rt}\cos\phi_{rt}} \tag{2.15}$$

Here, κ and t represent the field coupling factor and transmission factor, respectively. The parameters $\alpha_{rt} = \exp(-\alpha \cdot 2\pi R)$ and $\phi_{rt} = \beta \cdot 2\pi R$ represent the round-trip field attenuation factor and phase, where α and β are the real and imaginary parts of the complex transmission coefficients, respectively. An all-pass MRR is a special case of an add-drop MRR, without a drop bus waveguide and with $t_2 = 1$ and $\kappa_2 = 0$. The phase at the through/drop port can be derived from the Eqs. (2.14) and (2.15).

Unlike MZIs, MRRs have a narrow bandwidth and are designed to have a resonant wavelength. In simple terms, if the wavelength of the signal on a waveguide that is next to an MRR matches the resonant wavelength of the MRR, the signal gets coupled into the MRR, otherwise, it keeps propagating down the waveguide. Depending on how close the two wavelengths (optical signal's and MRR's) are, the transmission coefficients, and therefore, the amount of signal present in the four ports shown in Figure 2·2(b) changes. The transmission coefficients can be adjusted by applying an electrical voltage to an MRR, shifting its resonant wavelength by a desired amount.

The narrow bandwidth of MRRs makes them effective tunable filters. This property is leveraged to build WDM-based structures, such as weight banks (Tait et al., 2016; Tait et al., 2017). A weight bank consists of a series of cascaded MRRs, each tuned to a different wavelength, as shown in Figure 2·2(b)(bottom). When a multi-color input, containing light signals matching the resonant wavelengths of the MRRs, passes through the weight bank, each component of the input signal is weighted by the corresponding MRR. Using this design, a full dot product can be performed.

2.2.3 Efficiency and Scalability of Photonic Devices

The scalability and efficiency of a photonic GEMM core are influenced by factors such as the type of optical devices used, the required bit precision, the operating frequency, and the presence of noise sources, leading to distinct tradeoffs and limitations for each design. In this section, we offer a brief comparison of two commonly used photonic devices, MZIs and MRRs, to help clarify some of the design choices discussed in later sections.

As the input signal passes through each photonic device, it experiences power loss due to the device's insertion and propagation losses. Consequently, increasing the number of optical devices along the optical path necessitates a higher input power to maintain signal integrity at the output. Typically, optical losses are greater for MZIs compared to MRRs. MRRs are also smaller in size (with a radius dimension of $\sim$10 μm) than MZIs (with a dimension of $\sim$100 μm). When these metrics are considered, MRR-based designs can provide superior power and area efficiency (Al-Qadasi et al., 2022).

However, the extinction ratio (ER)—a measure of how precisely light signals can be modulated by the photonic device—is constrained by how closely critical coupling can be achieved, which depends on the thermal stability of MRRs (Bogaerts et al., 2012). Recent demonstrations of MRRs can achieve an ER of $<$ 25 dB (Shen, Yun et al., 2017). This limited ER of MRRs can pose challenges in achieving the adequate precision required by DNN models. In contrast, MZI implementations demonstrated extremely high ER of

$>$ 60 dB (Wilkes et al., 2016).

The ER in MRRs can be improved by employing stabilization circuits, which have been successfully demonstrated in electro-photonic transceivers for communication in previous work (Padmaraju and Bergman, 2014; Thonnart et al., 2018; Sun et al., 2015). Unfortunately, these demonstrations have been limited to non-return-to-zero (NRZ) (Ramon et al., 2018) or pulse amplitude modulation (PAM)-4 (Sun et al., 2018) keyings, respectively. When higher precision is required, such as 8-bit, the power and area overhead of the stabilization circuitry will increase compared to what has been previously demonstrated. Recent studies also indicate that the ER of MRRs can be enhanced by cascading multiple MRRs (Cheng et al., 2020). Challenges such as thermal crosstalk can be mitigated with improved tuning efficiency by methods including placing air trenches (Dong et al., 2010), simultaneously controlling the actuators (Milanizadeh et al., 2019), and using photoconductive heaters (Jayatilleka et al., 2019). These developments represent significant progress toward the realization of practical MRR-based photonic tensor cores.

2.2.4 Device Modulation Mechanisms in Silicon Photonics

n silicon photonics, efficient reprogrammability is crucial for devices like MZIs and MRRs, which serve as the fundamental building blocks of computing units. Modulation in these devices can be achieved through various mechanisms, each presenting different trade-offs among key device metrics, such as modulation bandwidth, optical loss, and device size—factors that significantly influence the scalability of the design. These modulation mechanisms can be broadly categorized into three groups: thermo-optic, plasma-dispersion-based, and nano/micro-opto-electro-mechanical systems (N/MOEMS)-based devices, as outlined in Table 2.1. Thermo-optic devices are widely utilized in silicon photonics because of their simple fabrication process and high modulation efficiency; however, their modulation speed is generally limited to a few kHz (Harris et al., 2014; Watts et al., 2013). In addition, the heaters used for modulation in thermo-optic devices dissipate signif-

Table 2.1: Device modulation mechanisms and corresponding device metrics in silicon photonics modulators.

Mod. Mechanism	Optical loss	$V_\pi L$	Mod. Bandwidth
Thermo-optic	Low	Low	Low
Plasma dispersion	High	High	High
N/MOEMS	Low	Low	Moderate

icant power and can easily lead to thermal crosstalk. For high-speed modulation in silicon photonics, the most commonly used actuation mechanisms rely on the plasma dispersion effect, which controls the refractive index by manipulating the free carrier density through carrier depletion (Sun et al., 2018), accumulation (Fujikata et al., 2016), or injection (Patel et al., 2014). While these modulators can easily achieve bandwidths in the tens of GHz, they are typically lossy and require longer device lengths. Recently, N/MOEMS-based devices have emerged as a viable alternative to the other two mechanisms. These devices (Baghdadi et al., 2021; Ramey, 2020; Feng et al., 2020a) offer moderate modulation frequencies (up to a few hundred MHz) and low optical loss, with negligible static power consumption. When designing a photonic tensor core, it is essential to carefully evaluate these methods and consider the tradeoffs between speed, energy, and area to achieve a practical and efficient design.

2.2.5 Noise and Errors in Photonics

Photonic tensor cores are typically limited by thermal and shot noise (Garg et al., 2023). Shot noise arises from the statistical fluctuations in the number of photons or electrons. It can be approximated by a Gaussian distribution as

$$I_S = \mathcal{N}(0,\ 2q_e I_\mathrm{D} \Delta f)\,, \tag{2.16}$$

where q_e is the elementary charge, I_D is the photodetector current, and Δf is the bandwidth. Thermal noise originates from the resistor in the trans-impedance amplifier (TIA) circuitry

and can be modeled as

$$I_T = \mathcal{N}\left(0,\ \frac{4k_BT}{R}\Delta f\right), \tag{2.17}$$

where k_B is the Boltzman constant, T is the temperature, and R is the feedback resistor of the TIA.

Both shot and thermal noise are additive to the output, i.e., $\Sigma_j x_j w_j + \mathcal{N}(0,1)\sigma_{\text{noise}}$ for a dot product (Garg et al., 2023). In a photonic core, the output is captured as an analog current, I_{out}. In the presence of noise, to achieve a desired bit precision b, one should be able to reach 2^b separable levels when reading I_{out}, i.e., SNR $\geq 2^b$. This leads to an exponential increase in power consumption as bit precision rises. Additionally, higher noise levels necessitate greater input power to maintain the same SNR, i.e., the same bit precision, during analog operations.

In addition to noise, photonic devices are also affected by process variations and fabrication errors. These issues can lead to biases in phase shifters and drifts in the resonant wavelength of MRRs, resulting in errors during operations. Previous works have proposed various solutions, including careful selection of parameters during fabrication (Sunny et al., 2021; Mirza et al., 2022), novel device modifications (Song et al., 2022), and error correction methods (Bandyopadhyay et al., 2021; Hamerly et al., 2021b) for MRR- and MZI-based designs to mitigate or calibrate away these errors.

The achievable numerical precision of the output vector v_{out} is limited by how well the input vector v_{in} and the weight matrix W can be encoded. This output precision can be quantified by adding the errors in quadrature, i.e., $\Delta v_{\text{out}}^2 = \Delta v_{\text{in}}^2 + \Delta W^2$. The encoding error of the input vector $\Delta v_{\text{in}} \leq 2^{-b_{\text{in}}}$ can be quantified by the bit precision of the DACs, b_{in}. For the output vector to be captured by ADCs with a bit precision of b_{out}, Δv_{out} must be $\leq 2^{-b_{\text{out}}}$ for the error to be dominated by the ADC precision. ΔW can be calculated by considering the encoding error of the weight matrix ($\leq 2^{-b_{\text{in}}}$), and all the perturbations caused by the imperfections in photonic devices during operations. By increasing the bit

precision of DACs, we can reduce the encoding error, and in turn, Δv_{out} to achieve a higher output precision.

2.2.6 Nonlinear Operations

Nonlinear operations such as nonlinear activation functions or conditional if-else statements in the optical domain require the use of nonlinear optical media (Jackson, 1975). A nonlinear optical activation function can be achieved by using laser-cooled atoms, where higher intensity light is absorbed more (Zuo et al., 2019). Saturable absorbers, in which the amount of light absorbed decreases with increasing light intensity, have also been proposed as optical nonlinear activations (Shen et al., 2017; Bao et al., 2011). However, the practical implementation of these nonlinear optical activations remains challenging, particularly because (1) they have not yet been miniaturized, and (2) repeated use of the nonlinear activation function rapidly degrades the signal.

Amplification-based nonlinear functions using semiconductor optical amplifiers (SOAs) in III-V materials can help counteract the signal loss described above (Roelkens et al., 2007). In principle, a photonic tensor core can be built in the III-V platform itself (Shi et al., 2020), but it is still preferable to use silicon photonics as it can be monolithically integrated with the CMOS transistors (Giewont et al., 2019) needed for controlling the photonic devices. Packaging the III-V module with a silicon photonics module presents a significant challenge to its feasibility. Even if a practical packaging solution were available, the power required to sustain the optical signal throughout the entire inference process would increase exponentially with the number of neural network layers. Consequently, we conclude that optical nonlinearities are impractical at this time, and have chosen to design a system where these nonlinearities are handled electronically.

2.3 The Residue Number System

RNS, which will be used in the later chapters, is an alternative numeral system to the traditional binary number system (BNS). This section provides a background on the RNS and RRNS.

2.3.1 RNS Basics

The RNS represents an integer as a set of smaller integers, referred to as residues, which are derived by applying a modulo operation to the integer using a selected set of n co-prime moduli. As an example, consider an integer X. In RNS, X is represented with n residues $x=\{x_1,...,x_n\}$ for a set of n moduli $\mathcal{M}=\{m_1,...,m_n\}$ where $x_i=|X|_{m_i}\equiv X \bmod m_i$ for $i\in\{1,...,n\}$. The integer X can be uniquely reconstructed from its residues and the corresponding moduli using the Chinese Remainder Theorem (CRT):

$$X=\sum_{i=1}^{n}(x_i M_i T_i)_M, \tag{2.18}$$

if all the moduli are co-prime and $X\in[0,M)$ where $M=\prod_i m_i$. Here, $M_i=M/m_i$ and T_i is the multiplicative inverse of M_i (i.e., $|M_i T_i|_{m_i}\equiv 1$).

The RNS is closed under both addition and multiplication. This allows GEMM operations in DNNs to be performed within the RNS space, provided that the output of the dot products remains within the RNS range (i.e., $[0,M)$). This $[0,M)$ range can be shifted to be symmetric around zero, i.e., $[-\psi,\psi]$, where $\psi=\lfloor(M-1)/2\rfloor$, to represent negative values.

2.3.2 Forward and Reverse Conversion

To perform operations in the RNS space, the operands must first be converted to the RNS domain. Likewise, after completing operations in the RNS space, the output needs to be converted back to the BNS. The forward conversion from BNS to RNS involves a modulo operation with the selected moduli. The reverse conversion from RNS to BNS can

be accomplished using several methods, including the CRT in Eq. (2.18) (Mohan, 2002). However, several studies have demonstrated that traditional conversion methods, such as CRT, present performance limitations for high dynamic ranges when arbitrary moduli sets are used (Wang et al., 2002; Mohan, 2007). When designing a high-speed, low-energy system, these conversions can quickly become a bottleneck, particularly as the dynamic range of operation increases. To address this, alternative designs utilize look-up tables (LUTs) (Lamb and DeBrunner, 1995), base-extension (Shenoy and Kumaresan, 1989; Gregory and Matula, 1975), mixed-radix conversion (Yassine and Moore, 1991), and special moduli sets (Gholami et al., 2009; Ahmadifar and Jaberipur, 2015; Wang et al., 2000; Mohan, 2007; Wang et al., 2002; Hiasat, 2019). In particular, using a moduli set in the form of powers of two, e.g., $\{2^k-1,\ 2^k,\ 2^k+1\}$ where k is a positive integer, significantly reduces the complexity of the modulo operations as they can be performed as simple shift operations. Numerous variations of this method have been explored and have been shown to significantly reduce the hardware overhead associated with these forward and reverse conversions (Gholami et al., 2009; Ahmadifar and Jaberipur, 2015; Wang et al., 2000; Mohan, 2007; Wang et al., 2002; Hiasat, 2019).

2.3.3 Redundant RNS

In the case of RNS, even minor errors in the residues can lead to significant errors in the corresponding integer they represent, as these errors are amplified during the reverse conversion. The RRNS (James and Pe, 2015; Yang and Hanzo, 2001a; Yang and Hanzo, 2001b) can detect and correct errors—making the RNS-based analog core fault tolerant. RRNS uses a total of $n+k$ moduli: n non-redundant (as in RNS) and k redundant. In RRNS, the same RNS operations are applied to both redundant and non-redundant residues. Once the output residues are obtained, error detection is conducted using majority logic decoding, wherein we divide the total $n+k$ output residues into $\binom{n+k}{n}$ groups with n residues per group and compare the results obtained from each group. An RRNS$(n+k,n)$ code can detect up

to k errors and can correct up to $\lfloor \frac{k}{2} \rfloor$ errors.

2.4 Hardware Solutions for DNN Acceleration

This section provides an overview of recent advancements in hardware platforms for DNNs, highlighting their advantages, disadvantages, and notable examples.

2.4.1 CMOS-based Platforms

A variety of CMOS-based solutions have been developed to support the growing compute requirements of AI. These solutions include general-purpose central processing units (CPUs) (Intel Corporation, 2021; Intel Corporation, 2023), GPUs (NVIDIA Corporation, 2024b; Choquette et al., 2021; Choquette, 2023), field-programmable gate arrays (FPGAs) (Farabet et al., 2009; Gschwend, 2020; Qiu et al., 2016; Farabet et al., 2011a; Huimin Li et al., 2016; Sankaradas et al., 2009), and specialized ASICs (Jouppi et al., 2017; Chen et al., 2014b; Farabet et al., 2011b; Chen et al., 2016; Chen et al., 2019; Chen et al., 2014a; Du et al., 2015), each with their unique capabilities and challenges, making them suitable for different aspects of DNN workloads. While these solutions provide significant architectural and performance benefits for DNN execution, they are based on CMOS transistors—devices that no longer scale in area or energy consumption according to Moore's Law and Dennard Scaling (Theis and Wong, 2017).

CPUs are general-purpose platforms traditionally used for a wide range of computing tasks. Recent advancements in CPU architectures have focused on improving parallelism (e.g., vector operations), increasing core counts, and incorporating specialized instructions to enhance DNN performance (Intel Corporation, 2021; Intel Corporation, 2023). While being flexible and compatible with a broad array of software ecosystems and programming frameworks, they generally exhibit slower performance in executing DNN tasks compared to more specialized hardware and are less power-efficient due to their general-purpose de-

sign.

GPUs (NVIDIA Corporation, 2024b; Scherer et al., 2010; Choquette, 2023; Choquette et al., 2021) have emerged as a cornerstone for DNN training and inference due to their massively parallel processing capabilities. GPUs can execute thousands of operations simultaneously, significantly speeding up DNN tasks, and their high memory bandwidth, which enhances data transfer rates and computational throughput. This performance has been further supported with software packages and libraries such as NVIDIA's CUDA toolkit (NVIDIA Corporation, 2024a) that are optimized for tensor operations. However, high-performance GPUs can consume substantial power, posing challenges to energy efficiency, and their high cost can make them less accessible for smaller-scale applications.

FPGAs offer a customizable hardware platform that can be tailored to specific DNN architectures, allowing them to be programmed to meet the specific needs of different DNN models. They are often suitable for low-power systems due to their high power efficiency. However, their performance typically does not match GPUs and ASICs. Moreover, programming FPGAs requires specialized knowledge and tools, which can increase development time and complexity.

ASICs are custom-designed chips optimized for specific applications, including DNNs. Because ASICs are tailored specifically for DNN tasks, they can incorporate highly optimized circuits that perform these tasks more quickly and with lower power consumption compared to other types of hardware. The primary drawback is their lack of flexibility. Unlike FPGAs, ASICs are hardwired for specific functions. Any changes in DNN algorithms or new requirements necessitate redesigning and manufacturing new ASICs, which can be both time-consuming and costly.

SAs are highly efficient structures, designed to accelerate parallel computation tasks, particularly in GEMM operations. SAs are commonly used in many FPGAs and ASICs developed in academia (Chen et al., 2016; Chen et al., 2019) and industry (Google Cloud,

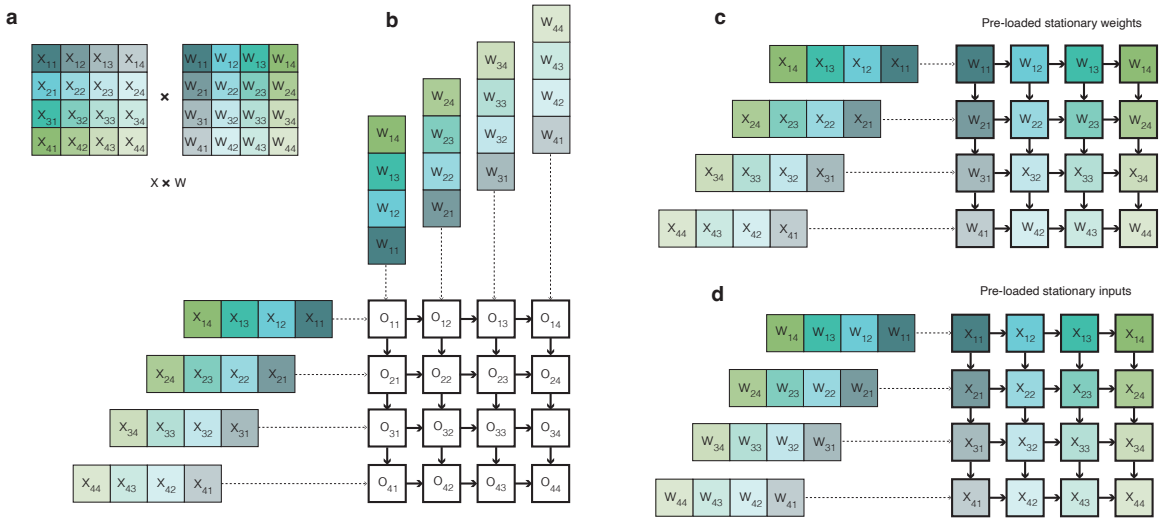

Figure 2·3: **(a)** Matrix multiplication between 4×4 input matrix X and 4×4 weight matrix W. **(b)** OS dataflow. **(c)** WS dataflow. **(d)** IS dataflow.

2024). SAs consist of a grid of PEs that perform synchronized computations and pass data to neighboring PEs in a rhythmic, pulse-like manner, hence the term "systolic". This design minimizes the need for extensive data movement, reducing latency and increasing throughput. Different dataflows can be implemented in systolic arrays to optimize for various performance metrics. The *weight-stationary* (WS) dataflow keeps the weights static within the PEs, reducing the need for frequent weight reloading and saving on memory bandwidth. Similarly, *input-stationary* (IS) dataflow keeps the inputs stationary within the PEs and moves the weights and partial outputs across the array. The *output-stationary* (OS) dataflow, on the other hand, keeps the partial sums stationary in the PEs until the final results are computed, which is beneficial for reducing the complexity of output data handling. There also exist other dataflows, optimized for specific hardware, such as row-stationary, adopted by the well-known DNN accelerator Eyeriss by Chen et al. (Chen et al., 2016). In this thesis, we consider the main three dataflows: WS, IS, and OS.

Compute-in-Memory (CiM) accelerators have emerged as a promising solution to the memory bottleneck in DNNs. These accelerators integrate memory and processing units together, allowing computations to occur directly within the memory array. CMOS-based

CiM accelerators leverage traditional memory technologies, such as SRAM (Chih et al., 2021; Sie et al., 2021) and DRAM (Kim et al., 2016; Gao et al., 2017), to perform operations like MVMs in parallel. This architecture significantly reduces data movement between memory and processing units. However, SRAM and DRAM are inherently volatile. In such systems, standby leakage power or refresh power can become significant, especially in low-power applications (Yu et al., 2021a).

2.4.2 Non-CMOS Platforms

In addition to advancements in CMOS-based technologies, several non-CMOS approaches have been explored to accelerate DNNs, aiming to overcome the limitations of traditional semiconductor methods. These emerging technologies leverage novel materials and computational paradigms to build CiM designs using non-volatile memory (NVM) structures.

NVMs offer the advantage of on/off capability without losing stored data. They typically have a smaller layout area than SRAM cells, resulting in higher integration density at the same technology node. These technologies include resistive random access memory (RRAM) (Song et al., 2017; Shafiee et al., 2016; Chi et al., 2016; Yao et al., 2020), phase change memory (PCM) (Le Gallo et al., 2023; Joshi et al., 2020; Kim et al., 2019), spin-transfer-torque magnetic random access memory (STT-MRAM) (Jain et al., 2017; Shi et al., 2020), spin-orbit-torque magnetic random access memory (SOT-MRAM) (Garello et al., 2019), ferroelectric field-effect transistor (FeFET) (Mikolajick et al., 2020), and electrochemical random access memory (ECRAM) (Tang et al., 2018).

While offering better density and energy efficiency than digital CiM solutions, for such technologies, multi-bit operation remains under exploration. Although there has been progress in research towards multi-bit per cell capabilities (Yao et al., 2020; Kim et al., 2019), currently, most foundries only offer single-bit NVM solutions (Yu et al., 2021a). Furthermore, these technologies face challenges related to variability and reliability. Updating values in an NVM array is costly and time-consuming, taking tens to hundreds of

nanoseconds (Xu et al., 2015; Yu et al., 2021b). Lastly, the low endurance of NVM cells limits the number of writes in the NVM array (Yu et al., 2021a).

2.4.3 Optical Platforms

The history of optical (or photonic) computing dates back to the 1960s when Bell Labs aimed to develop computers using optical transistors. Although this concept was appealing with the promise of extremely low-energy and high-bandwidth properties of light, it did not come to fruition due to the difficulties in optical alignment and calibration, bulky optical devices, as well as the absence of optical memory, a challenge that persists today. Since then, numerous advancements have made photonic platforms increasingly viable for computing, particularly in the context of deep learning. In this section, we summarize these advancements and explore how photonics is employed to accelerate deep learning.

Integrated photonics

After the optical transistor idea did not pan out after years of efforts, in 1994, (Reck et al., 1994) described a system that used an array of MZIs to perform matrix multiplications, which was initially for building an optical quantum processor. However, at the time, the bulky optical components made it very difficult to form a practical system. Thanks to the interest in optical telecommunications, advancements in photonics enabled the fabrication of photonic ICs (PICs) with a large number of photonic components on a single die including waveguides, modulators, and detectors to manipulate light and perform computations. The integration of photonic devices alongside CMOS transistors (Giewont et al., 2019) and PDKs becoming available in commercial CMOS foundries have further propelled research in academia and industry. Specifically, neural network computation has been a popular application for PICs, as the matrix-vector product is a fundamental operation in these applications where photonics offers an $O(N)$ energy scaling for N^2 fixed-point operations, unlike the digital systems that scale with $O(N^2)$ (Nahmias et al., 2019). The fundamental

parallelism of optics combined with high-speed low-energy photonic devices makes PICs a viable solution for DNNs (Ning et al., 2024).

As a result, many recent works leveraged the fundamental parallelism of optics combined with high-speed low-energy photonic devices to build specialized DNN accelerators. Prior works have explored MZI-based coherent architectures (Shen et al., 2017; Ramey, 2020), MRR weight banks (Tait et al., 2016; Tait et al., 2017; Bangari et al., 2019) and crossbar arrays (Feldmann et al., 2021; Ohno et al., 2022), novel architectures combining MZIs and MRRs (Shiflett et al., 2021; Shiflett et al., 2020), directional couplers (Zhu et al., 2024), and photonic switches (Peng et al., 2020). Some of these works integrated optical nonlinearities into their PICs with the goal of fully optical DNN accelerators (Feldmann et al., 2019; Mourgias-Alexandris et al., 2019; Jha et al., 2020), however, this approach poses significant scalability and practical implementation concerns due to the lack of maturity in optical nonlinearities, accumulated errors, and deterioration in the optical signal. While the experiments have almost exclusively focused on DNN inference, mainly due to the limited precision of analog operations, there have been a few works on leveraging PICs for DNN training (Bandyopadhyay et al., 2023; Filipovich et al., 2022; Pai et al., 2023; Hughes et al., 2018; Zhang et al., 2021). However, most demonstrations, even for inference, have been constrained to very small DNNs with a few layers and simple classification tasks.

Free-space optics

Free-space optics is another approach to optical computing that relies on the propagation of light through free space to perform computations, in contrast to integrated photonics. In these methods, diffractive optical elements (DOEs) can be employed to carry out MVMs. By encoding the weights of a DNN onto the DOEs, the input can be multiplied by the weight matrix in a single step, utilizing the interference patterns generated by the DOE. Some important examples include all-optical diffractive deep neural networks

(D^2NNs) (Lin et al., 2018) and diffractive processing units (DPUs) (Zhou et al., 2021). Another approach involves using diffractive lenses to implement convolution operations to implement CNN-like neural network architectures. This technique typically employs a combination of lenses (such as Fourier lenses) and spatial light modulators (SLMs) to perform the convolution (Sui et al., 2020; Yan et al., 2019; Chang et al., 2018; Miscuglio et al., 2020). Optalysys is an example from the industry that integrates silicon photonics and Fourier optics to accelerate CNNs, transformers, and fully homomorphic encryption (FHE) (Wilson, 2023; Cottle et al., 2020). While these systems can achieve exceptionally high data throughput and parallel processing capabilities, they generally require precise alignment and are sensitive to environmental disturbances, which limits their practical deployment. Given the current advancements, we believe that integrated photonics offers a more viable solution. Therefore, this thesis focuses on integrated photonics.

Evaluation of State-of-the-Art Photonic Architectures

Previous evaluations of several photonic tensor cores have typically been conducted in isolation from the systems surrounding the photonic cores (Feldmann et al., 2021; Xu et al., 2021; Shen et al., 2017; Wetzstein et al., 2020). While the performance numbers are impressive, it is essential to consider them within the context of a practical system. Some studies have attempted to integrate photonics with electronics (Shiflett et al., 2021; Mehrabian et al., 2018; Liu et al., 2019; Peng et al., 2020; Shiflett et al., 2020; Cheng et al., 2020), however, they have generally provided either a conceptual design or only a partial system evaluation. This section aims to highlight the importance of adopting a comprehensive, full-system perspective to better understand the limitations and opportunities of utilizing photonic cores.

Compute vs. Memory:

Small on-chip caches, typically on the order of hundreds of KBs as used in previous works (Shiflett et al., 2021; Shiflett et al., 2020; Peng et al., 2020)), are insufficient to

simultaneously store large DNN models, input/output data, and intermediate data. This limitation necessitates frequent off-chip memory accesses, which can stall the photonic core. Moreover, non-GEMM operations must be executed quickly enough to avoid throttling the high-throughput photonic core. Therefore, all electronic components within the accelerator need to be carefully designed and thoroughly analyzed to draw fair conclusions about the viability and performance of photonic technology.

Benchmarks: Prior photonic accelerators are either specifically designed for CNNs or report results only for CNNs (Shiflett et al., 2021; Bangari et al., 2019; Wu et al., 2021; Xu et al., 2021; Feldmann et al., 2021). While these accelerators excel at convolution operations, many are under-utilized and perform poorly when handling linear layers. In addition, several studies have relied on outdated and smaller neural networks that do not place significant stress on the memory system compared to state-of-the-art models. With the rise of non-CNN architectures, such as transformers, in recent years, focusing solely on CNNs offers a narrow view of the potential for photonic cores in DNN acceleration.

Overall, while previous works have provided insights into the raw capabilities of photonic compute cores, it is crucial to evaluate the entire system to fully understand the general applicability and real advantages of photonic technology in AI.

Chapter 3

System-Level Evaluation of Photonic DNN Inference

In this chapter, we design a full electro-photonic accelerator, ADEPT, for DNN inference. This study helps us understand the realistic potential of using photonics in today's AI systems. Unlike previous works where the photonic core is typically isolated from the rest of the system, we design and evaluate a complete system with a photonic tensor core and electronic data converters, non-linear units, and on-chip and off-chip memory units. Our evaluation shows that ADEPT can provide, on average, 5.73× higher throughput per watt compared to the traditional SAs in a full system, and at least 6.8× and 2.5× better throughput per watt, compared to state-of-the-art electronic and photonic accelerators, respectively[1].

3.1 Photonic Tensor Core Design

In ADEPT, we utilize an MZI-based tensor core due to the superior qualities of MZIs compared to MRRs (see Section 2.2.3). This section discusses the design steps of the photonic tensor core, the necessity of data conversion and tiling, and numerical precision during operations and provides a quantitative comparison against SAs, which can be considered the electrical counterparts of photonic tensor cores.

[1] Portions of this chapter were published previously in Demirkiran, Cansu, et al. "An electro-photonic system for accelerating deep neural networks." ACM Journal on Emerging Technologies in Computing Systems 19.4 (2023): 1-31. DOI:10.1145/3606949.

3.1.1 MZI Array

An MZI-based photonic tensor core uses that any real-valued matrix M can be decomposed into two unitary and one diagonal matrix via SVD, i.e., $M = U\Sigma V^T$, which can be represented by forming a mesh of MZIs (See Section 2.2.1. Figure 3·1(a) illustrates an example of programming a 3×3 matrix M into a 3×3 MZI array in Figure 3·1(b). To program a matrix M, it is first decomposed into the three matrices through the SVD. Next, the phase values required to program each of these three matrices, i.e., $(\phi_U, \phi_\Sigma, \phi_{V^T})$, are computed by using a phase decomposition algorithm, which breaks a large orthogonal matrix U into a series of 2×2 orthogonal matrices acting on different input rows (Clements et al., 2016). Finally, for a $h \times h$ sized M, a total of h^2 phase values are programmed into the array with h^2 MZIs to represent M. Importantly, the total number of phase values is equal to the number of elements in M. Therefore, the memory footprint required for storing the decomposed parameters is the same as that for storing the original matrix. While SVD works for any two-dimensional matrix, rectangular matrices result in unused optical channels in the MZI array, reducing efficiency. Therefore, in this study, we consider M to be square, with the dimensions of $h \times h$.

Figure 3·1(b) illustrates an example of how the input and output vectors are programmed and read out. Each element of the input vector is programmed using a Mach-Zehnder Modulator (MZM) along with a phase shifter, for encoding the amplitude and sign of the input optical signals, respectively. The output vector from the MVM operation, including both amplitude and sign, is detected using coherent detectors with the assistance of a local oscillator.

In this setup, once M is programmed into the MZI array, it can be reused for multiple input vectors without the need for reprogramming. This operation's rate is primarily constrained by the speed at which input signals can be modulated through the MZMs. To achieve high throughput, this design necessitates MZMs with a high modulation frequency.

Consequently, we opted for modulators based on the plasma dispersion effect, which can be modulated at $\geq$ 10 GHz (See Section 2.2.4 for details). While this modulator has a relatively high optical loss (1.2 dB), a single MZM per optical does not significantly impact the total loss on the optical path. On the other hand, there exist $2h+1$ MZIs per optical channel. Although the high modulation rate is desirable, such high optical loss per MZI drastically limits the scalability of the photo-core, which encourages us to use a low-loss thermo-optic or N/MOEMS-based MZI. For the MZI array, we prefer a NOEMS-based MZI design, as it provides a moderate modulation bandwidth (a few hundred MHz) as against a thermo-optic design (in the range of KHz). The NOEMS-based MZI takes a few ns to program, meaning that each time the matrix in the MZI array is reprogrammed, there will be an idle period where the photo-core is inoperable. However, with a dataflow minimizing the number of reprogramming in the MZI array, this overhead (e.g., $\leq$100 cycles for 10 GHz clock) is tolerable and does not significantly slow down the operations.

3.1.2 Photo-core Dataflow

The analog photonic computing unit in ADEPT, referred to as a photo-core, is designed to perform MVM operations that can eventually be composed into a GEMM operation. GEMM operations in DNNs (e.g., FC and convolution layers) typically involve a multiplication between a weight matrix and an input matrix. The shapes of these matrices vary in each layer in a DNN and can get quite large for some layers ($\geq$1000). The photo-core, however, has a fixed size of $h \times h$. Therefore, matrices bigger than the photo-core size are divided into $h \times h$ sized submatrices (also called tiles) and loaded into the photo-core one by one.

Figure 3·1(a) shows a simple example of this step. First, the input and weight matrices are flattened if necessary (for 2D convolutions) using im2col pre-processing (Anderson et al., 2017), and the weight matrix is broken into $h \times h$ tiles. Each tile is then decomposed into three matrices using SVD and the respective phase values ($\phi_U, \phi_\Sigma, \phi_{V^T}$) are calculated

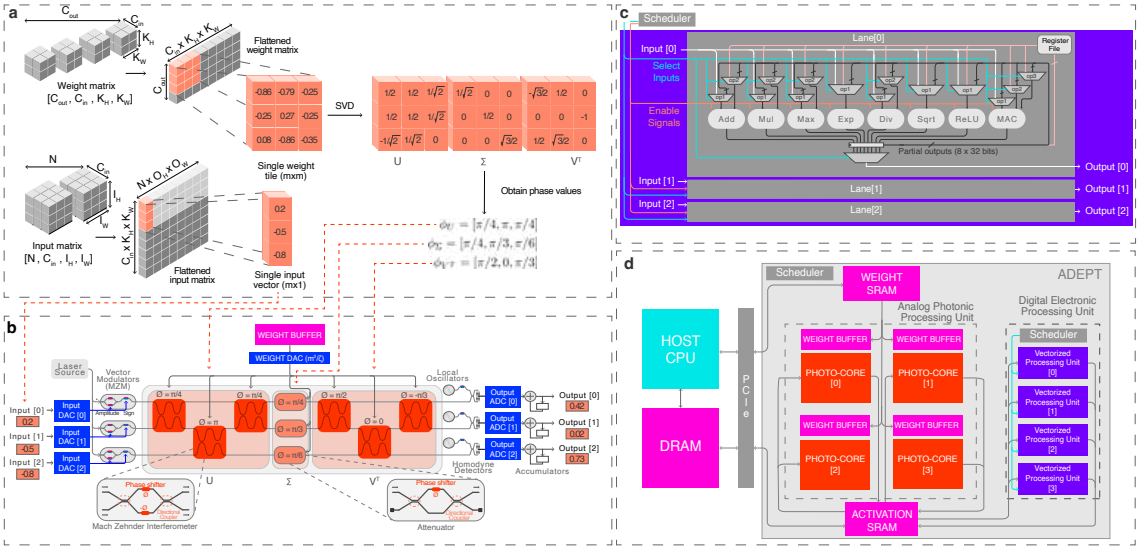

Figure 3·1: Diagram showing different components of ADEPT and how operations are performed. (a) Example GEMM operation in the photo-core. (b) Programming input and weight matrices into the photo-core. The $h \times h$ (here $h = 3$ as an example) photo-core consists of $2 \times h(h-1)/2 = 6$ MZIs (for U and V^T) and 3 attenuators (for Σ). (c) Microarchitecture for a single digital electronic vectorized processing unit. The unit comprises $h = 3$ digital lanes, each consisting of arithmetic units to perform non-GEMM operations. (d) Full system architecture including the host CPU, the DRAM, and ADEPT—interconnected using a PCI-e interface. As an example, we show four photo-cores and four vectorized processing units.

using the phase decomposition algorithm (Clements et al., 2016). Here, SVD and phase composition are costly operations, typically handled by a host CPU. Therefore, performing SVD on runtime before each reprogramming of the photo-core would stall the DNN execution. Fortunately, in DNN inference, the weights are fixed and reused for each new input. This encourages us to program the weights into the MZI array (instead of the input/activation matrices that are computed at runtime) and perform tiling, SVD, and phase decomposition operations once offline. This way, all operations shown in Figure 3·1(a) can be performed *only once upfront* for each weight tile in the host CPU. The phase values obtained from above can then be directly programmed into the MZI array during DNN inference, as shown in Figure 3·1(b).

In addition, we adopt a WS dataflow where a weight tile is kept stationary in the photo-core. Once the tile is loaded into the photo-core, the values are maintained in the MZI array while all input vectors that need to be multiplied with this specific tile are fed into the photo-core sequentially, vector by vector. This WS dataflow is crucial for minimizing the frequency of updates to the MZIs and for amortizing the power and latency costs associated with programming the MZI array.

Due to our WS approach, the partial output vectors generated by a weight tile do not contribute to the same final output result. Instead, the partial output vectors generated by the subsequent weight tiles are added to those stored in the SRAM, with the accumulation results then written back to the SRAM array. This approach requires additional SRAM reads and writes for accumulation compared to an OS dataflow. However, the OS approach necessitates reprogramming the MZI array every cycle, which either introduces long latencies between each MVM operation—reducing the operation rate to just a few hundred MHz—or results in high optical loss along the optical path, significantly limiting scalability. Similarly, an IS approach is inefficient as it requires the costly SVD and phase decomposition to be performed at runtime rather than offline.

3.1.3 Data Conversion

The input matrix and the computed phase matrix are initially stored as digital values. To utilize these digital values in analog operations, DACs are required for both inputs and weights. The outputs of the input DACs, which are analog voltages, are either passed through the input modulators (MZMs) to convert the electrical signals into optical signals with an amplitude proportional to the input value, or they are used directly to program the corresponding MZI via the outputs of the weight DACs.

Once an input vector passes through the MZI array, the output signals are detected by photodetectors, where the optical signals are converted into a photocurrent. This photocurrent is then converted into digital bits using ADCs. The precision of these DACs and ADCs

must be selected based on the required data precision to ensure the desired accuracy in the DNN, which is typically around 8 bits for DNN inference.

In an $h \times h$ photo-core, h^2 DACs will be needed. Depending on the chosen h, this many DACs can take up a significant chip area. Instead, we can utilize the idle time between two consecutive tiles to reprogram the MZIs, allowing us to perform multiple D-to-A conversions using the same DAC. This requires $\lceil h^2/\zeta \rceil$ DACs for ζ is the number of values that can be programmed within this idle period by a single DAC.

3.1.4 Numerical Precision and Accuracy

Maintaining numerical precision during DNN computation is a significant challenge when using an analog core. A similar issue arises in photonic computing: the numerical precision of the output vector v_{out} is limited by how well the input vector v_{in} and the matrix M can be encoded. The error of M can be calculated by considering all classical photonic operations, i.e., the transformations afforded by U, Σ, and V^T. We can then determine the magnitude of the total perturbation in M, which can be defined as:

$$\Delta M = \sqrt{\langle \|\Delta U\|^2 \rangle + \langle \|\Delta \Sigma\|^2 \rangle + \langle \|\Delta V\|^2 \rangle}. \tag{3.1}$$

The error of each matrix can be quantified by its normalized Frobenius norm $\|\Delta X\|^2 = \sum_{ij} |\Delta X_{ij}|^2 / m$, such that $\Delta X = \sqrt{\|\Delta X\|^2}$ where X is a placeholder for U, Σ, and V^T.

The phase encoding error and component error that can cause crosstalk are the two main device limitations that contribute to the perturbation of each transformation (Bandyopadhyay et al., 2021; Shafiee et al., 2022; Zhu et al., 2020; Shokraneh et al., 2020). The phase encoding can be quantified as $\varepsilon_\phi \leq 2^{-b_w}$, where b_w is the bit precision of the weight DAC. The error induced by a single-phase setting error is: $\|\Delta X\|^2 \approx \varepsilon_\phi^2 / m$ (Bandyopadhyay et al., 2021). Given that the matrix is composed of MZIs, which only use 50:50 DCs, we consider the DC splitting error as the component error. The matrix perturbation induced by a single

DC error ε_{DC} (typically due to fabrication imperfections) is $\|\Delta X\|^2 \approx 2\varepsilon_{DC}^2/m$ (Bandyopadhyay et al., 2021).

In practice, fabrication imperfections are typically correlated, meaning that neighboring devices often experience similar perturbations. However, accounting for these correlations can significantly complicate analyses and obscure mathematical clarity. To address errors in such scenarios, Monte Carlo simulation tools are advisable. For simplicity, we can assume an uncorrelated error model for both encoding and component errors.

In each MZI, there are two DCs and a single-phase shifter. In the Σ matrix, there are only m parallel MZIs, thus $\langle\|\Delta\Sigma\|^2\rangle = \varepsilon_\phi^2 + 4\varepsilon_{DC}^2$. In the matrices U and V^T, there are $m(m-1)/2$ MZIs. The depth of the photonic circuits in M grows with $O(m)$, and component errors cascade as light propagates down the mesh. A naïve programming of the phases will result ΔM that grows as $O(m^{1/2}\varepsilon)$ (Clements et al., 2016; Reck et al., 1994). However, more sophisticated error-correction methodologies (Bandyopadhyay et al., 2021; Hamerly et al., 2021a; Hamerly et al., 2021b) can provide a better scaling for the component errors, such that the error grows as $O(m^{1/2}\varepsilon_\phi + m\varepsilon_{DC}^2)$:

$$\langle\|\Delta U\|^2\rangle = \langle\left\|\Delta V^T\right\|^2\rangle = \frac{m(m-1)}{2}\left[\frac{\varepsilon_\phi^2}{m} + 2\frac{2\varepsilon_{DC}^4}{3m}(m+1)\right], \tag{3.2}$$

where the first term in the square bracket is the contribution due to phase encoding error and the second is due to component error. The corrected programming strategy effectively allows for a squaring of the component errors that is advantageous when $\varepsilon_{DC} \leq m^{-1/2}$. Finally, the error for the overall matrix M can therefore be defined as in Eq. (3.1).

The precision of the input and output vectors can now be quantified by adding the errors in quadrature: $\Delta v_{out}^2 = \Delta v_{in}^2 + \Delta M^2$, where $\Delta v_{in} \leq 2^{-b_{in}}$ and Δv_{out} must be $\leq 2^{-b_{out}}$. Assuming a reasonable DC splitting error of < 0.1 % (this quantity should be measured in the fabricated silicon photonic wafer), up to $\sim$8 bits output precision can be maintained in matrices with the size up to 256×256 when 10-bit and 12-bit DACs are used for inputs

and weights, respectively.

This analysis effectively correlates the SNR of the DACs with the SNR of the output ADC. It is important to note that the input and weight data encoded in the photonic GEMM device can still maintain 8-bit precision; however, they must be encoded using DACs with SNRs that match the required bit precision.

3.1.5 Photo-core Performance

In this section, we analyze the hardware performance of the MZI-based photo-core described earlier and explore the advantages of using a photonic core compared to a purely electronic core. Much of our evaluation of ADEPT focuses on comparisons with SAs, which are widely used for DNN acceleration due to their fast operations and high energy efficiency. We consider SAs a strong comparison point because they serve as the electrical counterpart to a photonic tensor core, given the similarity in dataflow between the two structures. By implementing the SA at the register transfer level (RTL), we were also able to explore various design choices (e.g., different array sizes and numbers of arrays) to ensure a fair head-to-head comparison with ADEPT.

The photo-core leverages light, which oscillates at hundreds of terahertz, giving it a significant bandwidth advantage over electronic SAs. In the photo-core, bandwidth is primarily limited by the sampling rate of data converters (up to 10 GHz in this work), whereas SAs are constrained by parasitic resistance, capacitance, and inductance. In fact, when using Cadence Genus with GF22FDX, the SAs could not meet the timing requirements for frequencies of 2 GHz and above. To address this, we employed parallelism to effectively operate the SA at higher frequencies. For instance, to achieve a 10 GHz operation, we used ten 1 GHz SAs with clock cycles offset by 100 ps. While the latency of this parallelized SA remains 1 ns, its throughput is equivalent to a single SA running at 10 GHz.

For this part of our analysis, we assume both photo-core and SA are isolated from the system, with weights and inputs preloaded and available in the SRAM, and that read/write

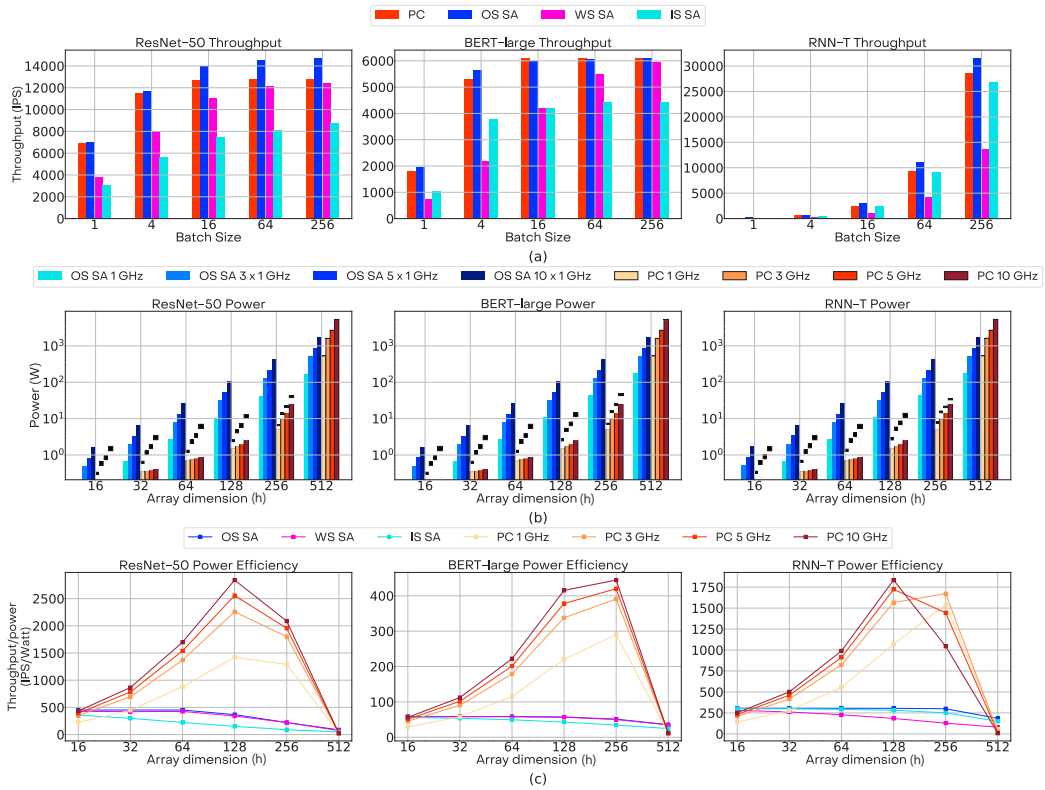

Figure 3·2: (a) Throughput vs. batch size of 128×128 photo-core (PC) and SA with three dataflows at 1 GHz clock. (b) Power consumption of the OS SA and the WS photo-core for different array sizes and clock frequencies. Laser power is shown with solid color, ADCs/DACs with the white diagonal pattern, and E-to-O/O-to-E conversion with the black diagonal pattern. (c) Power efficiency of SAs and photo-core for different array sizes and clock frequencies.

speeds are sufficient to meet the demands of both arrays. To provide SAs a strong baseline, we considered OS, WS, and IS for SAs as dataflow can have a significant impact on performance. Figure 3·2(a) shows the throughput we can achieve when using the three dataflows for SAs for three different benchmarks. We found that OS outperforms WS and IS for SAs, primarily due to the high latency associated with loading data into the SA between tiles in the WS and IS dataflows. Therefore, from here onwards, we use OS for SA in the rest of the comparison.

Throughput

To compare throughput, we use a single 128×128 array (this choice will be justified later in this section) for both the photo-core and the SA. Figure 3·2(a) shows the performance of the photo-core and that of the SA, both operating at 1 GHz clock rate for three different networks—ResNet-50, BERT-large, and RNN-T (one plot per network)—for different batch sizes in terms of inferences per second (IPS). In general, we observe that the photo-core's WS dataflow is more advantageous compared to the OS SA when the weight matrices are relatively large and the input matrices are small (e.g., RNN-T with small batch sizes) because each weight tile needs to be loaded only once.

Throughput vs. Batch Size: Figure 3·2(a) shows that as the batch size increases, throughput (and correspondingly utilization) of the arrays increases and eventually saturates. Among the three DNNs, we observe that the throughput saturates for ResNet-50 and BERT-large more quickly than RNN-T. This is mainly because the small input matrices in RNN-T require a fewer number of vectors to be multiplied with the same tile. Thus, the utilization and throughput continue to significantly increase until larger batch sizes for RNN-T. Moreover, as the batch size increases, latency overhead for loading new tiles becomes less significant as the majority of time is spent on performing MVM operations per tile.

Throughput vs. Clock Frequency: One approach to increase the throughput of any computing device is to increase the clock frequency. We therefore attempt to increase the clock frequency of the photo-core and the SAs from 1 GHz to 3 GHz, 5 GHz, and 10 GHz. The rate of MVM operations increases linearly with the clock frequency in both the photo-core and the SAs. However, programming values into the MZI array requires a fixed amount of time that is unaffected by changes in clock frequency. As a result, the increase in the photo-core's throughput is sub-linear.

Power Consumption

Figure 3·2(b) compares the average power consumed by the photo-core and the SA of different sizes. For the photo-core, the power consumption for lasers, DACs and ADCs, and E-to-O and O-to-E conversions are shown separately. For this analysis, we use a batch size of 256 to ensure that the throughput is nearly maximized across all networks. Overall, the photo-core's power consumption is lower than the SA up to an array size of 256×256.

For the SAs, the power consumption increases linearly with the number of PEs, i.e., quadratically with the array size h. For the photo-core, the laser power increases exponentially with the depth of the array. This is due to the fact that a linear increase in the number of optical devices on an optical channel leads to an exponential impact on optical power loss (in watts) in the propagating signal. As a result, Figure 3·2(b) illustrates that laser power significantly increases as the tile size grows, eventually dominating the power consumption for larger array sizes. For an $h \times h$ photo-core, we need h DACs, h E-to-O conversion circuits, h ADCs, and h O-to-E conversion circuits for modulating the input signals and reading out the output signals. The input/output DACs/ADCs perform a conversion each cycle. In addition, we need DACs for programming the $h \times h$ weight matrix. These DACs used for programming the weights into the MZIs are not active during each cycle. The weights are programmed into the MZIs once per tile, and the DACs remain idle until all MVMs for that tile are completed. However, as the array size increases and latency decreases, the average power consumption of the DACs/ADCs rises. This is because the same number of conversions are carried out in a shorter time span.

Power Efficiency

Figure 3·2(c) shows the power efficiency in IPS per watt (IPS/W) of SAs and photo-cores for different array sizes and clock frequencies. We observe that, for the photo-core, 128×128 is the most power-efficient array size for all experiments. This can be explained

by the fact that beyond a certain size, the laser power starts dominating the power consumption of the photo-core. Additionally, beyond a certain array size, the utilization decreases and so the throughput saturates. Therefore, due to the exponentially increasing laser power and saturating throughput, we observe a drop in the power efficiency beyond an array size of 128×128. For SAs, the power increases quadratically with the array dimension h. However, because the throughput increases less than quadratically with h, the power efficiency decreases as the array size increases.

Across different frequencies and array dimensions, we observe that photo-core can provide up to **9.87**×, **9.32**×, and **7.69**× better power efficiency than OS SA for ResNet-50, BERT-large, and RNN-T, respectively, when only GEMM operations are considered.

Overall, we observe that for the same clock frequency, while the throughput is comparable, photo-core provides a better power efficiency than the best-performing SA. As 128×128 array size and 10 GHz clock frequency is the most power-efficient combination, we use this configuration for further evaluations in later sections.

3.2 Building the Full System

After analyzing the photonic core isolated from the rest of the system, we start building the full accelerator architecture, ADEPT, which includes a custom digital electronic vectorized processing unit for non-GEMM operations and memory units for storing weight and activation data. ADEPT is placed within a system with a host CPU and off-chip DRAM. Additionally, we introduce optimizations and parallelism to the system and evaluate the full performance.

3.2.1 Vectorized Processing Unit for Non-GEMM Operations

Although more than 90% of the DNN operations are GEMM operations, a non-trivial amount of non-GEMM operations must also be performed as part of DNN inference. These

operations include element-wise non-linear operations (e.g., ReLU, GELU, and sigmoid); reduction operations (e.g., softmax and max-pool); batch and layer normalizations; and element-wise multiplication and addition (e.g., bias). It is more practical and efficient to perform these non-GEMM operations in the digital domain (See Section 2.2.6).

In ADEPT, we use a digital electronic ASIC to perform non-GEMM operations. We designed the ASIC in a vectorized manner where each unit has the same number of lanes as the number of optical channels in a photo-core such that the output of each optical channel in the photo-core is fed to one lane in the vectorized processing unit. The microarchitecture of a single lane in a vectorized processing unit is shown in Figure 3·1(c). Each lane has separate units for multiplication, addition, division, max, square root, and exponential operations (each 32-bit) that enable the system to complete a wide variety of non-GEMM operations. These arithmetic units are implemented as custom digital CMOS circuits. All lanes in the vectorized processing unit can operate in parallel and operations can be pipelined for non-GEMM operations that require multiple arithmetic operations. Each arithmetic unit uses a multiplexer to choose the input from 1) SRAM, 2) the output of any of the arithmetic units, or 3) the register files of the vectorized unit, as operands. Here the register files (64 KB each) are used to store the constants (which are loaded up front) for the non-GEMM operations or the outputs of the arithmetic units. Multiplexers are controlled by a scheduler that decides when each arithmetic operation is used. The outputs of digital electronic ASIC are written back to the SRAM—to be used in the next layer of the DNN.

To extract the maximum performance from ADEPT, we must match the throughput of the photo-core and the digital electronic ASIC. It is, however, challenging to design a digital ASIC at the photo-core's clock rate. To maintain the balance between the analog and digital parts of ADEPT, we use multiple vectorized processing units per photo-core. For a photo-core operating at 10 GHz (See Section 3.1.5 for the justification of this design

choice), we use 10 logical units in parallel for each operation within the individual vector lane, each operating at 1 GHz clock frequency and offset by 0.1 ns to one another.

3.2.2 Memory Units

ADEPT utilizes two separate SRAM units: one for input/output activations (referred to as the activation SRAM) and one for weights (referred to as the weight SRAM). The SRAM units can transfer data between each other through direct memory access (DMA) and communicate with the host and DRAM through the peripheral component interconnect express (PCI-e) fabric. The two SRAM units are separated because, generally, a dichotomy exists between the activations and the weights, and data transfer between them is not frequent. The activation SRAM is used to store both input and output activations because effectively, the output of one layer is the input of the next layer. At runtime, both the photo-core and the digital electronic ASIC read and write a vector of size $h = 128$ (the size of the photo-core, see Section 3.1.5 for the justification of this design choice) from and to the activation SRAM. We use separate dedicated read/write ports in the activation SRAM for the photo-core and the digital electronic ASIC.

Transferring a complete weight tile (128×128) from weight SRAM to photo-core in one step requires a large SRAM bandwidth. In contrast, transferring one vector at a time requires a large latency between tiles. Hence, we use a weight buffer for each photo-core as an intermediate stage. We load the tile for the next set of MVM operations into the weight buffer, while the photo-core is performing operations with the current weight values. The data is then programmed into the photo-core directly from the weight buffer, minimizing the latency between consecutive tiles in the photo-core and increasing the photo-core's overall utilization and the system throughput.

We also consider an off-chip DRAM as part of the full system. While the weight matrices fit the on-chip SRAM, inputs, and outputs need to be read from and written to an off-chip DRAM before and after every inference.

3.2.3 Performance Optimizations

In this section, we introduce two optimizations designed to efficiently orchestrate operations in ADEPT and maximize system performance by minimizing the latency overhead associated with non-GEMM operations and data transfers.

Pipelining

We implement pipelining for GEMM and non-GEMM operations in ADEPT. Specifically, as soon as an output vector—the final result after accumulating partial outputs from a GEMM operation—is generated, it is immediately sent to the digital electronic ASIC for non-linear operations. This allows non-GEMM operations to commence without waiting for the entire GEMM operation to complete.

Additionally, multiple non-GEMM layers may need to be executed consecutively, or a single layer might require more than one logical unit. To further optimize ADEPT, we pipeline these sequential non-GEMM operations within the digital electronic ASIC. For instance, in a softmax layer, the exponential unit, max unit, and multiplication unit are used in sequence. While one element is processed by the exponential unit, the previous output from the exponential unit can be processed by the max unit. Consequently, as long as data dependencies are maintained, different non-GEMM operations or various steps within a non-GEMM operation using different arithmetic units in the digital electronic ASIC can be parallelized and pipelined.

Figure 3·3 shows the impact of this optimization on the latency of ADEPT when running ResNet-50, BERT-large, and RNN-T. In ResNet-50, the max-pool, average-pool, ReLU activations, and softmax layers; in BERT-large, the layer norm, GELU, and softmax operations; and in RNN-T, the element-wise addition and multiplication, sigmoid, and tanh operations within the LSTM layers are all computed in the digital electronic ASIC. Although non-GEMM operations constitute a small percentage of the overall operations in

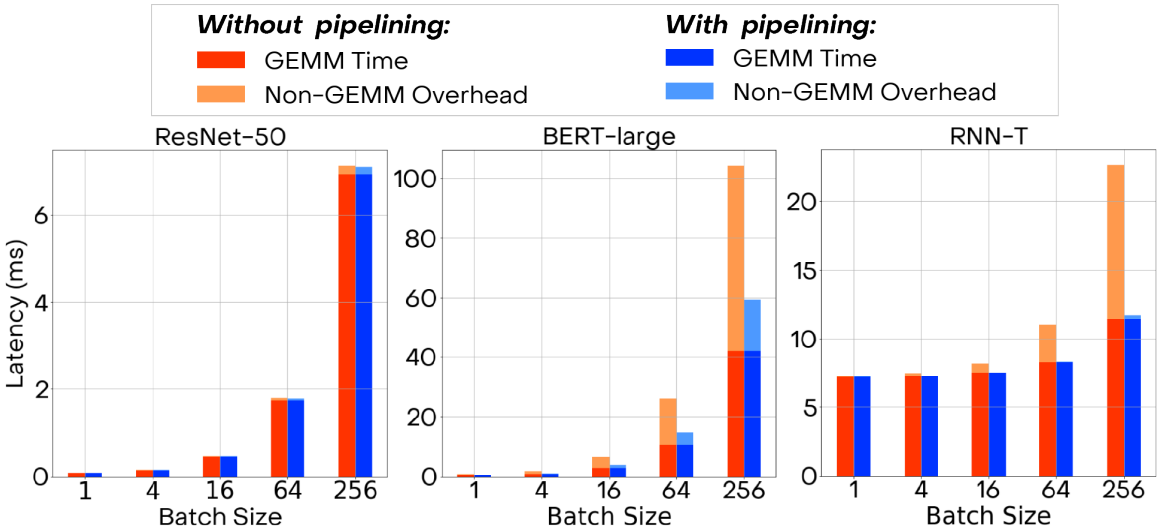

Figure 3·3: Latency of ADEPT with a 128×128 photo-core operating at 10 GHz clock with and without pipelining the GEMM and non-GEMM operations. Here the latency is for one batch of inputs for three networks. The results are calculated for varying batch sizes.

these networks, they can introduce significant overhead if not carefully pipelined.

In Figure 3·3, ResNet-50 experiences the least overhead from non-GEMM operations. With batch normalization folded, ReLU becomes the most frequent non-GEMM operation, which can be effectively overlapped with GEMM operations. In contrast, BERT-large and RNN-T see an increase in the number of cycles spent in the digital electronic ASIC due to division and exponential operations in GELU, softmax, sigmoid, and tanh functions.

As batch size increases, GEMM operations become more efficient because more input vectors are multiplied with the same tile, allowing for more frequent reuse of weights. However, the cycles required for non-GEMM operations increase linearly with batch size. Consequently, the time spent on non-GEMM operations grows more rapidly than the time spent on GEMM operations as batch size increases. This results in a smaller proportion of non-GEMM operations being overlapped with GEMM operations. We observe a reduction in latency of up to 5.73% in ResNet-50, 43.03% in BERT-large, and 48.22% in RNN-T when we introduce pipelining in the accelerator.

Optimized Buffering

is constrained by the rate at which data are fed into the photo-core. Although the latency and bandwidth of the activation and weight SRAM arrays can be designed to match the photo-core's throughput, the size of these arrays is limited. When activations and weights do not fit within these SRAM arrays, frequent DRAM accesses become necessary, which are slower and can easily bottleneck system performance.

To avoid being limited by DRAM latency during runtime, it may be necessary to limit the batch size for a given neural network. However, larger batch sizes generally lead to better throughput. To address this, we propose an optimized buffering method that maximizes the batch size stored in the activation SRAM without ever spilling back to DRAM during runtime. This method efficiently utilizes the available space in the SRAM during inference, allowing the inputs of the next batch to be loaded from DRAM without causing performance degradation.

We describe this optimized buffering method as a convex optimization problem. Let $\vec{x}_{\mathrm{c}} = [x_{\mathrm{c}}(t_0),\, x_{\mathrm{c}}(t_1),\, \ldots,\, x_{\mathrm{c}}(t_{\mathrm{max}})]$ be a vector representing the activation SRAM array usage while performing inference on a batch of activations over time. Here $t_{\mathrm{i+1}} = t_i + \Delta t$ where Δt is some time interval chosen to ensure the optimization problem is tractable for the host CPU. Similarly, $\vec{x}_{\mathrm{pcie}} = [x_{\mathrm{pcie}}(t_0),\, x_{\mathrm{pcie}}(t_1),\, \ldots,\, x_{\mathrm{pcie}}(t_{\mathrm{max}})]$ is a vector representing the activation SRAM usage of the data (next input batch) being transferred from DRAM into SRAM over time. For a given $\vec{x}_{\mathrm{c}}$, an optimal $\vec{x}_{\mathrm{pcie}}$ data transfer schedule can be obtained by solving the following optimization problem:

$$\begin{aligned} \text{Maximize:} \quad & \sum_{t=t_0}^{t_{\mathrm{max}}} x_{\mathrm{pcie}}(t) \\ \text{Subject to:} \quad & 0 \leq x_{\mathrm{c}}(t) + x_{\mathrm{pcie}}(t) \leq x_{\mathrm{max}}; \\ & x_{\mathrm{pcie}} \geq 0;\; x_{\mathrm{pcie}}(t_{-1}) = 0;\; x_{\mathrm{pcie}}(t_{\mathrm{max}}) = x_{\mathrm{input}}; \\ & 0 \leq \Delta x_{\mathrm{pcie}}(t) \leq \text{Max. PCI-e bandwidth} \end{aligned}$$

The constraints in the optimization problem can be described as: the total SRAM usage

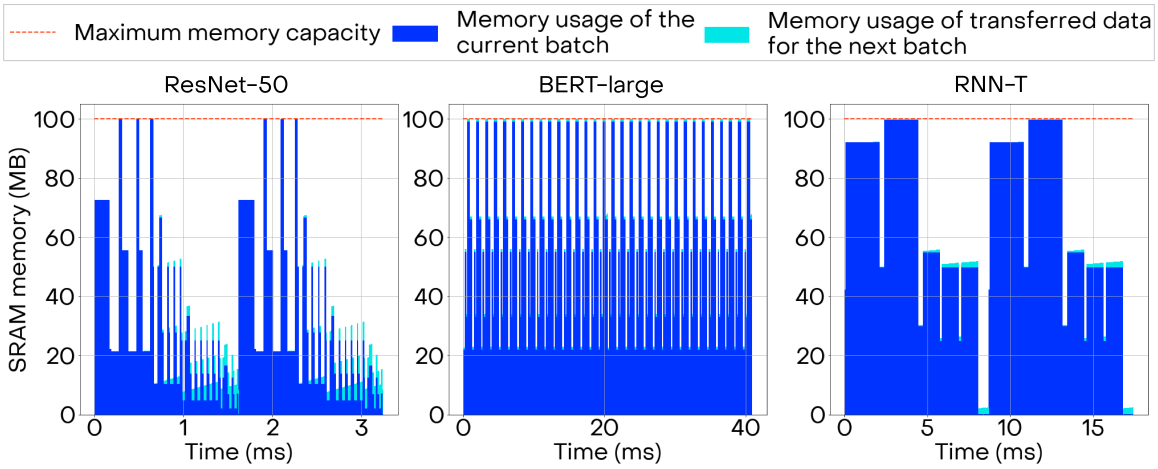

Figure 3·4: Activation SRAM usage for computing on the current batch of inputs along with data transfer for the next batch of inputs within ADEPT. Both input and output activations for the current batch must be stored in the activation SRAM (dark blue) while the input data are transferred for the next batch (light blue). A 128×128 photo-core at 10 GHz clock is used with batch sizes of 58, 88, and 50 for ResNet-50, BERT-large, and RNN-T, respectively to fully use the 100 MB activation SRAM capacity.

(1) should be less than the given SRAM size ($x_{\max}$), (2) should not be negative at any time, and (3) should start from zero; (4) the total amount of data transferred will be equal to the input size of the next batch, and (5) the data transfer rate should be slower than the maximum PCI-e bandwidth.

Here, maximizing this area under the curve of memory usage of the transferred data for the next batch guarantees transferring the data as soon as possible under the constraint of a maximum PCI-e bandwidth.

If the program fails to generate a schedule $x_{\text{pcie}}(t)$ that meets the specified constraints for a given batch size and maximum PCI-e bandwidth, a smaller batch size or a larger bandwidth (if available on the hardware) should be selected. We use the aforementioned optimization program to determine the largest batch size that ensures the memory usage for storing activations from both the current and next batch never exceeds the available SRAM size. This approach guarantees that all DRAM data transfers for the next batch of

inputs occur concurrently with the inference of the current batch. The optimized schedule is computed only once by the host CPU before runtime.

Up until now, we have used a large batch size of 256 to evaluate the saturated throughput of both ADEPT and SAs. However, given that the limited SRAM array size, a batch size of 256 may not be viable.

For this study, we choose a 100 MB activation SRAM and a 300 MB weight SRAM, which is enough to accommodate the weights of all three networks comfortably within a reasonable chip area. Figure 3·4 illustrates the usage of the activation SRAM array for both the current and next batch when utilizing the optimized DRAM access mechanism. We limit the batch size to the maximum value where inference on the entire batch can be completed without requiring any DRAM transfers—58, 88, and 50 for ResNet-50, BERT-large, and RNN-T, respectively. The activation SRAM stores the inputs and outputs of all GEMM and non-GEMM operations over time. When GEMM and non-GEMM operations are pipelined and run concurrently, the memory usage reflects the activation data for both operations. Figure 3·4 reveals that the networks do not use the whole SRAM array throughout the inference, which creates an opportunity to transfer the inputs of the next batch.

We compare the performance of our optimized buffering technique against double buffering (Sancho and Kerbyson, 2008), a commonly used method for minimizing the impact of data transfer latency. In double buffering, one-half of the memory is allocated for the current inference, while the other half is used for transferring the inputs for the next inference. Consequently, the maximum batch sizes for this scheme are half of those achievable with our optimized buffering technique for the three networks. For ResNet-50 and BERT-large, the optimized buffering technique improves the throughput only by 1.3% and 0.4% compared to double buffering. This is because these two networks have already high utilization in the photo-core and their throughputs are saturated for the considered

batch sizes. Remarkably, however, this optimization increases the throughput of RNN-T by 89.7% over double buffering.

Impact of Optimizations

Figure 3·5 summarizes the impact of the two optimizations—pipelining and optimized DRAM buffering, on ADEPT at a system level. The roofline represents the peak throughput of the photo-core, while the memory ceiling is determined by the bandwidth of the activation SRAM. The baseline (no optimization) scenario refers to the setup without any pipelining and with double buffering.

When comparing the three networks, ResNet-50 exhibits lower arithmetic intensity and is primarily memory-bound. As a result, we observe that the performance of ResNet-50 without optimizations is already near the roofline, meaning that additional optimizations only provide marginal performance improvements. BERT-large benefits from pipelining with a 1.76× better throughput because of the frequent non-GEMM operations. In contrast, the optimized DRAM buffering, which allows for larger batch sizes compared to double buffering, does not provide significant benefits for ResNet-50 because the photo-core's utilization is already saturated at smaller batch sizes. RNN-T, on the other hand, exhibits lower utilization compared to the other two networks. This lower utilization is primarily due to the recurrent nature of the network, which necessitates frequent changes of weight tiles and frequent non-GEMM operations within the LSTM layers. Therefore, supporting larger batch sizes via the optimized DRAM buffering increases the performance significantly by 1.92×, and pipelining improves the throughput for RNN-T by 1.83×.

The analysis presented in this section underscores the importance of considering non-GEMM operations and memory limitations, as well as the need to evaluate different types of DNNs. While non-GEMM operations and memory constraints can limit the throughput of the photo-core, these challenges can be mitigated and performance enhanced through appropriate optimizations, such as pipelining and efficient data buffering.

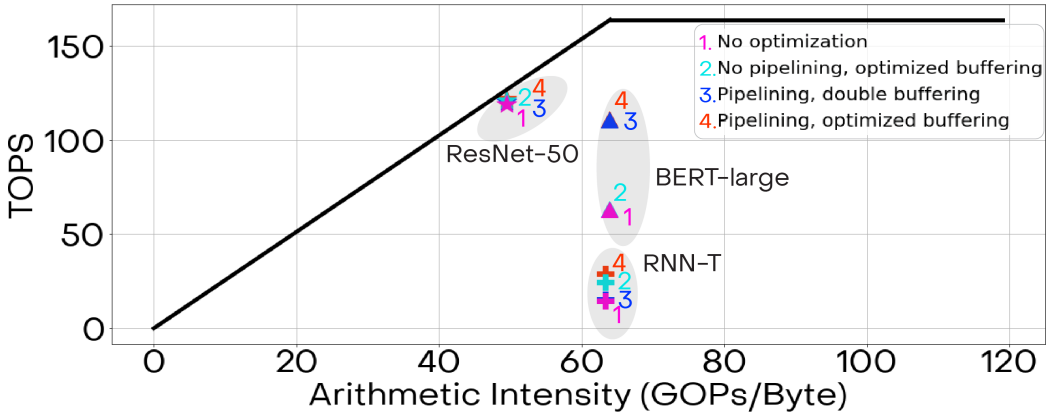

Figure 3·5: Roofline plot showing the effect of optimizations on ADEPT with a single 128×128 photo-core. The arithmetic intensity is calculated using MAC operations over activation SRAM reads/writes.

3.2.4 Parallelism

ADEPT can be scaled up to include multiple photo-cores. We offer two parallelization strategies for distributing the workload among multiple photo-cores: data parallelism and tile parallelism. Data parallelism aims to accelerate MVMs by copying the same weights to all photo-cores. Each photo-core performs the same operations on different inputs in a batch. Tile parallelism is a finer granularity model parallelism where different tiles of a weight matrix are distributed across multiple photo-cores. Unlike data parallelism, all inputs in one batch are sent to all photo-cores.

ADEPT can also use WDM-based parallelism. WDM uses multiple wavelengths for encoding different input vectors at once similar to data parallelism. The scheme requires multiplexing and demultiplexing circuits that can be constructed from microring resonators (Bogaerts et al., 2012) or cascaded unbalanced MZIs (Xu and Shi, 2018). WDM parallelism is equivalent to data parallelism in terms of throughput, but the same MZI array and weight DACs can be used by all inputs encoded in the wavelengths.

Figure 3·6 shows how the latency scales with the increasing number of photo-core counts for both data and tile parallelism. We use the batch sizes previously considered (see

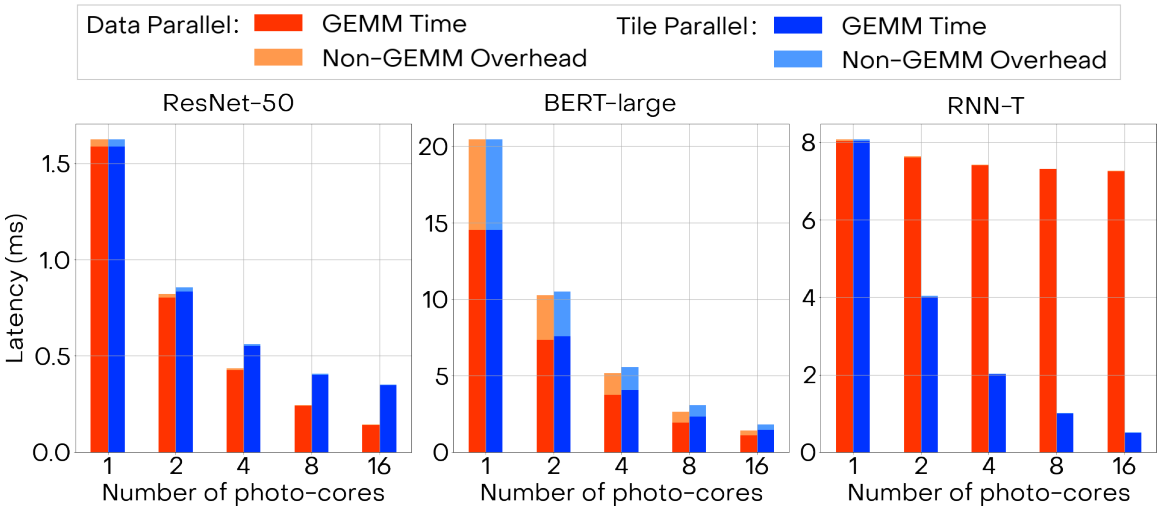

Figure 3·6: Latency of ADEPT (128 × 128 photo-core at 10 GHz clock) when executing the three neural networks with different photo-core counts using data and tile parallelism.

Section 3.2.3), i.e., 58, 88, and 50 for ResNet-50, BERT-large, and RNN-T, respectively. We keep these values constant as we increase the number of photo-cores.

Figure 3·6 shows that data parallelism provides an almost linear decrease in inference latency with increasing photo-core count when the number of input vectors within a batch is large enough to be shared among the photo-cores. The latency is dominated by MVM operations for large input sizes, and so as the number of photo-cores increases, the throughput proportionally increases. We observe this in ResNet-50 and BERT-large where the input matrices are large enough to be spread among the photo-cores and we can maintain high utilization. In contrast, when the number of input vectors per core decreases, the reduction in latency saturates due to the decrease in the utilization of the photo-cores. We observe this in RNN-T. Data parallelism provides 11.30×, 14.47×, and 1.11× lower latency for ResNet-50, BERT-large, and RNN-T when we increase the photo-core count from 1 to 16.

The advantage of tile parallelism is limited by the number of tiles in a GEMM layer. The networks with larger weight matrices (i.e., BERT-large and RNN-T) better exploit this parallelism. Tile parallelism provides 11.24×, 16.0×, and 4.62× lower latency for BERT-

large, RNN-T, and ResNet-50, respectively, when the photo-core count increases from 1 to 16.

Multiple photo-cores means a linear increase in the area and the power consumption for the analog photonic computing unit. WDM provides an opportunity to reduce this area increase. WDM allows the input vectors to be mapped across the different wavelengths that are routed to the same photo-core. Therefore, WDM offers the same throughput as data parallelism without using multiple copies of the MZI array and weight DACs. When we compare data parallelism with n photo-cores against a single photo-core leveraging n wavelengths in WDM, the photo-core with WDM uses $(n-1)m^2$ fewer MZIs and $(n-1)m^2/\zeta$ fewer weight DACs.

This WDM approach is, however, limited by the amount of optical power that can be injected into a single waveguide. High optical power causes strong nonlinear absorption in the waveguide and the peak optical power is limited to $\sim$30 mW for ensuring signal integrity (Chen and Joshi, 2013). For a 128×128 array operating at 10 GHz with a single wavelength, the peak laser power per waveguide is 20.7 mW, which is under the 30 mW limit. To leverage WDM and still stay under the nonlinearity limit, we need to scale down the photo-core size accordingly. For example, for multiplexing 2 wavelengths in an optical waveguide, we need to reduce the photo-core size to 64×64. This configuration achieves $1.4\times$ better IPS/W/mm^2 compared to two 64×64 photo-cores with a single wavelength and $1.23\times$ better IPS/W/mm^2 compared to a single 128×128 photo-core with a single wavelength, on average among the three evaluated networks, ResNet-50, BERT-large, and RNN-T, when all cores are operating at a 10 GHz clock frequency.

3.2.5 System Performance

The full electro-photonic system consists of a host CPU, DRAM, PCI-e bus and the electro-photonic accelerator ADEPT (see Figure 3·1(d)). ADEPT is connected to the host CPU and DRAM through a PCI-e bus. Host CPU handles the compilation and any other operations

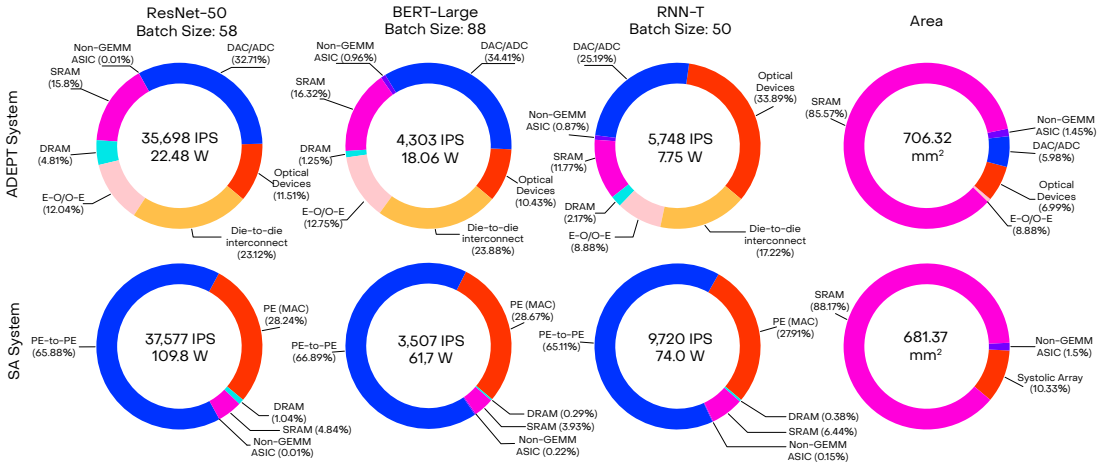

Figure 3·7: Average total (static and dynamic) power distribution and area distribution of ADEPT (128×128, 10 GHz photo-core) and the SA system (128×128, 10×1 GHz array, OS dataflow).

required by the DNN model that can be performed offline including pre/post-processing (e.g., resizing, decoding, etc.) and precomputation of the phase values for the MZIs. The inference is then performed fully in ADEPT without any interference from the host CPU.

In this section, to answer the main question of how much the real benefit in a complete system is, we include all the components of the system and the optimizations discussed in Section 3.2.3 and provide a full system-level comparison between 128×128 WS ADEPT and a 128×128 OS SA (see Figure 3·7).

From Figure 3·7, we can see that the optical devices in the photo-core (i.e., laser, MZIs, modulators) used for the GEMM computation take up only between 10-35% of the overall power consumption in the ADEPT system depending on the DNN model. The other components of the system (i.e., ADCs/DACs, O-to-E/E-to-O conversions, die-to-die communication, and SRAM) consume significant power—which proves the necessity of the system-level evaluation. For the SA, data transfer between the register files of the PEs dominates the power consumption of the SA system. We observe that SRAM dominates the area distribution for both electronic SAs and ADEPT for the chosen configuration.

In ADEPT, the photo-core and the digital electronic ASIC are in different chiplets to

take advantage of the technology nodes that provide the best performance for each individual electronic and photonic ICs. The two chiplets are 3D integrated through an interposer. In the latter, the SA and the rest of the electronic components share the same chiplet. The optimizations used for ADEPT are also applied to the SA system.

Our analysis shows that a system with ADEPT consumes 4.88× (109.8 W vs. 22.48 W), 3.42× (61.7 W vs. 18.06 W), and 9.55× (74.0 W vs. 7.75 W) less power for ResNet-50, BERT-large, and RNN-T, respectively. This translates to **4.89**×, **3.24**× and **9.06**× better power efficiency (IPS/W). Also, ADEPT provides **4.5**×, **2.97**× and **8.34**× better power-area efficiency (IPS/W·mm^2) compared to a SA. This shows us that although including the system components in evaluation decreases the performance of the standalone photo-core, we can still benefit from using photo-cores instead of SAs in a system.

3.2.6 Execution Model

In this section, we describe the execution model for using ADEPT as part of the full system. This process is summarized in Figure 3·8. In this process, we take a DNN model and compile it on the host CPU to generate a program in the form of a tensor-type graph. We use ONNX models (exported from common frameworks like PyTorch) and a loader to build a high-level program graph, where the nodes represent operations on higher-dimensional array data types. A directed acyclic graph (DAG) is then created using a cost-model-based partitioner, with nodes annotated to indicate whether the operations will be executed on the CPU or the ADEPT device.

For code generation and optimizations, we utilize a low-level virtual machine (LLVM)-based optimizer on the host CPU. The operations marked for execution on the ADEPT device are expanded into a stream of ADEPT instructions, and a scheduling pass is performed to overlap GEMM and non-GEMM operations. The annotated program graph is then used to optimize the schedule and pipeline computations between the host CPU and the ADEPT device, ensuring efficient communication between the two.

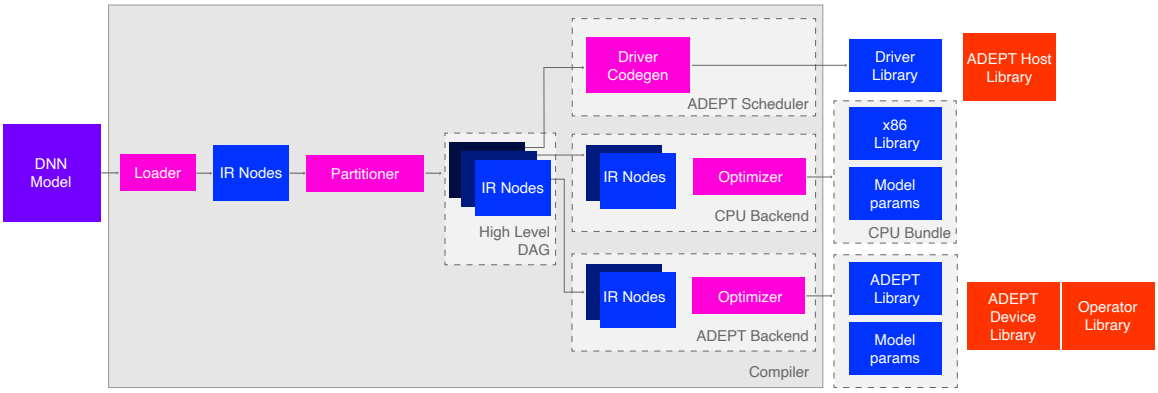

Figure 3·8: Compilation process of an ML model for ADEPT.

The generated code for these three partitions is linked with the corresponding libraries to produce two executable binaries: one for the host and one for ADEPT. Notably, the host CPU performs this compilation only once, after which all subsequent inferences are offloaded to ADEPT.

3.3 Related Work

For completeness, in this section, we compare the full ADEPT system against state-of-the-art electronic (Chen et al., 2019; Chen et al., 2016; Lee et al., 2018; Jouppi et al., 2020) and photonic (Shiflett et al., 2021; Peng et al., 2020; Liu et al., 2019) accelerators.

3.3.1 Electronic Accelerators

Besides the traditional SAs, over the years, more flexible electronic accelerator architectures have been proposed to efficiently accelerate DNN inference. Table 3.1 compares ADEPT against state-of-the-art electronic accelerators. Broadly, while is not the most area efficient, ADEPT provides at least 6.8× higher IPS/W compared to other electronic accelerators.

When compared to NVIDIA's H100 GPU (results obtained from MLPerf Inference v3.1 (Reddi et al., 2020) for an 8×H100 system), ADEPT provides 13.6×, 22.2×, and

Table 3.1: Comparison against state-of-the-art electronic and photonic accelerators.

	ADEPT (This work)		**Eyeriss** (Chen et al., 2016)	**Eyeriss v2** (Chen et al., 2019)	**UNPU** (Lee et al., 2018)	**TPU v3** (Jouppi et al., 2020)
Tech Node	90 nm photonics + 22 nm CMOS		65 nm	65 nm	65 nm	16 nm
Clock rate	10 GHz		200 MHz	200 MHz	200 MHz	940 MHz
Benchmark	AlexNet	ResNet-50	AlexNet	AlexNet	AlexNet	ResNet-50
Batch size	192	58	4	1	15	N/A
IPS	217, 201	35,698	35	102	346	32,716
IPS/W	7,476.78	1,587.99	124.80	174.80	1,097.50	18.18
IPS/W/mm^2	10.59	2.25	10.18	N/A	68.59	0.01

24.8× better IPS/W for ResNet-50, BERT-Large, and RNN-T, respectively.

3.3.2 Photonic Accelerators

For completeness, we provide Table 3.2, which provides a quantitative comparison against the state-of-the-art photonic accelerators. However, the limitations of these works mentioned in Section 2.4.3 should be taken into account for a fair comparison. The numbers reported in Table 3.2 depend on many design choices, i.e., optical components, ADCs/DACs, memory array sizes, non-linear arithmetic units, communication, etc., and how comprehensive the evaluation is.

Table 3.2 shows that while the full system of ADEPT has a relatively low area efficiency (due to large SRAM arrays), it can achieve 2.5 × better IPS/W than Albireo-C and 10.2 × better IPS/W than DNNARA for the same batch size of 1. Although the batch size is not reported in HolyLight, ADEPT's and HolyLight's power efficiencies are comparable when ADEPT uses a batch size of 1. However, ADEPT's activation SRAM array is adequate to support larger batch sizes which helps increase the utilization of the photo-core—providing a better system performance. When the maximum batch size available is used for ADEPT, it can achieve more than 8.3× better IPS/W compared to all other three photonic accelerators.

It should be noted that the small on-chip memory arrays used by previous works have a smaller area footprint than ADEPT, but the energy overhead of required DRAM transfers

Table 3.2: Comparison against state-of-the-art photonic accelerators.

	ADEPT (This work) (This work)				**Albireo-C** (Shiflett et al., 2021)	**DNNARA** (Peng et al., 2020)	**HolyLight-A** (Liu et al., 2019)
Clock rate	10 GHz				5 GHz	1.2 GHz	1.28 GHz
Benchmark	AlexNet		ResNet-50		AlexNet	ResNet-50	AlexNet
Batch size	1	192	1	58	1	1	N/A
IPS	6,478	217, 201	12,641	35,698	7,692	9,345	50,000
IPS/W	872.17	7,476.78	1,021.17	1,587.99	344.17	100	900
IPS/W/mm^2	1.23	10.59	1.59	2.25	2.75	0.45	40.07

should be considered to provide a fair comparison. The goal of this section is not to claim a more performant photonic core; in contrast, we aim to highlight the importance of a system-level analysis when evaluating photonic accelerators and encourage the community to adopt a pragmatic approach.

3.4 Evaluation Methodology

In this section, we describe our evaluation approach when we compare ADEPT against SAs and state-of-the-art accelerators. For our evaluation, we picked three DNNs: ResNet-50(He et al., 2016), BERT-large (Devlin et al., 2018), and RNN-T (He et al., 2019). These three state-of-the-art networks—all part of the MLPerf inference data-center benchmarks (Reddi et al., 2020)—represent the diversity in layer types, sizes, and shapes that we observe in DNNs. We combine architecture, circuit, and device-level analyses to evaluate the full system.

3.4.1 Architecture-level Analyses

For our architecture-level evaluation, we used a mix of SCALE-Sim (Samajdar et al., 2018) and RTL simulations. SCALE-Sim is a cycle-accurate simulator specifically designed for SA-based DNN accelerators. The simulator takes the SA configuration (such as array size and dataflow type) and the neural network specifications (like layer sizes and batch size) as inputs, calculates the number of cycles required to run the neural network, and gener-

ates traces for SRAM and DRAM reads and writes. To accurately model the photo-core in ADEPT, we adapted the WS dataflow within SCALE-Sim to reflect the WS methodology used by the photo-core. This involved incorporating the latency associated with programming the weight tile into the MZI array and accounting for the latency overhead from transferring weights from the SRAM to the weight buffer.

SCALE-Sim allows us to simulate our specific dataflow and make direct performance comparisons between the photo-core and SAs. However, since SCALE-Sim focuses solely on GEMM operations, we designed the digital electronic ASIC using SystemVerilog RTL to assess the performance of non-GEMM operations and incorporated the optimizations described in Section 3.2.3. For each DNN, we combined the timing results from SCALE-Sim and RTL simulations to determine the overall performance.

3.4.2 Circuit- and Device-level Analyses

We synthesized our RTL designs for the digital electronic ASIC in ADEPT and SAs using Cadence Genus (Cadence, 2024) with a standard cell library designed in the GF22FDX technology node (Global Foundries, 2024). We generated the SRAM arrays using an SRAM compiler for GF22FDX.

To minimize the impact of slow DRAM transfers on performance, we adopt a similar strategy to that used in previous works employing large on-chip memory arrays (Kim et al., 2021; Lavely, 2022). However, maintaining low access latency with a single large SRAM array is challenging. Therefore, we use multiple smaller SRAM sub-arrays to construct larger memory arrays, which helps achieve the desired capacity while maintaining manageable access latency. Each SRAM sub-array has 64 KB capacity with ~ 1 ns access latency. For high clock rates ($f_c > 1$ GHz), we read from multiple arrays, each offset by $1/f_c$ ns with its neighbor. In total, we use 300 MB weight SRAM and 100 MB activation SRAM.

The photo-core is powered by a laser. To determine the required laser power per channel

P, we conducted an analytical calculation that factors in (1) the laser wall-plug efficiency, (2) the losses of the various optical devices, and (3) the SNR needed for an 8-bit output, as follows:

$$P = \frac{(\kappa\,\mathrm{SNR}_{\mathrm{shot}})^2 \cdot (q\,\Delta f/4)}{\eta_{\mathrm{det}} \cdot \eta_{\mathrm{array}} \cdot \eta_{\mathrm{mod}} \cdot \eta_{\mathrm{cpl}} \cdot \eta_{\mathrm{laser}}}, \tag{3.3}$$

where $\mathrm{SNR}_{\mathrm{shot}}$ is the SNR assuming shot noise only and κ (assumed to be ≈ 3) accounts for noise contributions (e.g., thermal noise and transistor noise) other than the shot noise. The overall $\mathrm{SNR} = \kappa\,\mathrm{SNR}_{\mathrm{shot}} = 2^{b_{\mathrm{out}}}$ with b_{out} being the bit precision of the output ADC. Here, q is the elementary charge, and Δf is the bandwidth of the coherent detector (related to the clock frequency). The ηs account for the transmissivity from the laser to the detectors. η_{mod} is the transmissivity of the modulator ($\approx$1.2 dB loss (Akiyama et al., 2012)), η_{array} is the transmissivity of the MZI array ($\approx$0.04 dB loss per MZI (Baghdadi et al., 2021) and each signal passes through $2m+1$ MZIs), η_{cpl} is the fiber laser-to-chip coupling efficiency ($\approx$2 dB loss), η_{det} is the efficiency of the photodetectors ($\approx$80% (Lischke et al., 2015)), and η_{laser} is the wall-plug efficiency of the laser ($\approx$20% (Mourou et al., 2013)). All the photonic devices in the photo-core are simulated using Lumerical Maxwell-Equations solver FDTD and circuit-level simulator INTERCONNECT (Ansys, 2024).

The 10-bit inputs require a high ER in the input modulators. This can be achieved by using active optimization approaches (Wilkes et al., 2016; Miller, 2015). We use two additional MZIs per modulator as variable beam splitters for obtaining a perfect 50:50 splitting in both ends of the modulator. In addition, an equalized phase-dependent loss between the two middle arms is achieved as the MZM is driven in a differential push-pull manner.

The inputs and the weights require 10-bit and 12-bit DACs, respectively, to guarantee the 8-bit-precise outputs read by the ADCs (See Section 3.1.4). Due to the lack of publicly available DAC prototypes in GF22FDX with our desired precision, for our analysis, we used a 14-bit DAC (Huang et al., 2021) designed with 28 nm CMOS technology with a 10

GS/s sampling rate and 177 mW power consumption. Note that the power consumption of 10-bit and 12-bit DACs will be less than a 14-bit DAC. Therefore, we scaled the power numbers using a widely accepted figure of merit (FoM) for the performance of DACs, i.e., $\text{FoM} = 2^B \cdot f_s|_{6(b_{\text{DAC}}-1)} / P_{\text{DAC}}$. Here, b_{DAC} is the bit precision of the DAC, $f_s|_{6(b_{\text{DAC}}-1)}$ is the output signal frequency where the spurious free dynamic range has dropped with 6 dB (= 1 bit) in comparison with the expected results ($\approx 6b_{\text{DAC}}$), and P_{DAC} is the power consumption of the whole DAC (Li and Zhou, 2020). In essence, the power consumption of a DAC—with the same FoM—is proportional to 2^b_{DAC}. Therefore, it can be assumed that a 12-bit DAC (for the weights) with the same FoM will consume $2^2 = 4$ times less power than a 14-bit DAC. Similarly, a 10-bit DAC (for the inputs) will consume $2^4 = 16$ times less power than a 14-bit DAC. The 10-bit input and 12-bit weight DACs will then consume 11.06 mW and 44.25 mW, respectively. Similar to DACs, we use 10-bit ADCs in 28 nm technology at the output. As mentioned in Section 3.1.3, each 10 GS/s DAC is responsible for multiple conversions per tile ($\zeta = 100$). Each ADC has a 5 GS/s sampling rate and consumes 29 mW (Guo et al., 2020).

The E-O and O-E conversion power is based on the total energy required to operate the modulator circuitry, which is $\sim$ 20 fJ/bit, and the detector circuitry, which is $\sim$ 297 fJ/bit (Sun et al., 2015). Each DRAM access is assumed to be 20 pJ/bit (Horowitz, 2014). The die-to-die interconnect between the photonic and electronic chiplets consumes 0.3 pJ/bit (Dehlaghi and Chan Carusone, 2016).

3.5 Discussion

In this study, our goal was to develop a balanced architecture that harnesses the advantages of photonics for accelerating GEMM operations while (1) avoiding bottlenecks from digital electronic processes or storage limitations, and (2) more than offsetting the overheads associated with electrical-optical and analog-digital conversions. We demonstrate that, al-

though performance expectations should be moderated compared to prior works that evaluated photonic cores in isolation, photonics technology can still be effectively leveraged for DNN acceleration through careful hybrid system design. In this section, we address the limitations of photonics technology and our proposed design, along with potential solutions.

In a hybrid architecture like ADEPT, frequent data conversions between the analog and digital domains are necessary, and these conversions contribute significantly to the overall energy consumption. Although advancements in process technology may reduce the overhead of these conversions, they will continue to pose a fundamental limitation on the system's speed and efficiency. As a potential solution, performing more operations in the optical domain could be considered. However, this approach increases the cumulative optical loss due to the larger number of optical devices, which in turn reduces the SNR and the achieved bit precision at the output.

The modulation bandwidth of MZIs can be seen as another limitation, contributing to latency overhead when reprogramming the MZI array. Furthermore, this MZI-based approach necessitates SVD and phase decomposition of the original values before they can be programmed into the MZIs. For completeness, we measured this one-time cost for a 128×128 matrix. We performed SVD on a 128×128 array with randomly chosen values between [-1,1] using the NumPy Linear Algebra module. Then, we implemented and timed the phase decomposition algorithm as described in Clements et al. (Clements et al., 2016). We repeated the experiment 10 times for different matrices on a 2 sixteen-core 2.8 GHz Intel Gold 6242 (Intel, 2024) and calculated the average energy consumption to be 1728.9 Joules (DRAM energy included, calculated using PyRAPL (PowerAPI, 2024)) and the average runtime to be 11.5 seconds. As an example, in ResNet-50, there are a total number of 1756 128×128 tiles. This makes the total energy consumption approximately 2724.7 kJ and the total latency approximately 5 hours in total. This process can be further

accelerated through GPU implementations. In the context of DNN inference, where the weight values remain constant, the SVD and phase decomposition need to be performed only once, and the associated cost is spread across all inferences. However, in situations where the weight values are not fixed—such as during DNN training—the MZI-based approach becomes impractical. Performing SVD and phase decomposition in real-time would introduce significant delays, severely hindering the performance of the photo-core.

In addition, in a photonic core, the dynamic range of values is constrained by the output ADCs. During MVM operations, the product of 8-bit inputs and weights generates more than 16 bits of information, but the ADCs reduce the precision of these MVM outputs back to 8 bits. This loss of information in the partial outputs is a challenge unique to analog cores and can lead to accuracy degradation in DNNs compared to 8-bit digital hardware. For DNN inference, additional training efforts can help mitigate this accuracy loss and maintain it within the desired range. We verify that 8-bit precision is adequate to achieve a $\geq 99\%$ of the FP32 accuracy in the benchmarks we evaluated (e.g. ResNet-50, BERT-large, and RNN-T) after performing several epochs of quantization-aware training (QAT) (Wu et al., 2020; Krishnamoorthi, 2018). While this precision is adequate for inference, training in the photo-core requires higher precision, which is challenging to maintain in analog hardware. This precision limitation due to ADCs and solutions for achieving higher dynamic range for DNN inference and training will be explored in Chapter 4.

Lastly, in our study, we utilized large SRAM arrays to accommodate all DNN weights on-chip and support large batch sizes without being bottlenecked by DRAM transfers. Consequently, SRAM occupies the majority of the area in both ADEPT and electronic SAs. While having large local SRAMs is advantageous for performance, it also tends to reduce chip yield due to the larger chip size. Moreover, SRAM size cannot be increased indefinitely; therefore, for larger DNNs with billions to trillions of parameters, DRAM transfers will become unavoidable.

In such scenarios, scaling out to multiple chips is necessary to expand the total SRAM cache size and enhance performance. Addressing the challenge of designing a scaled-out system with multiple chips, each featuring multiple photo-cores, mapping a DNN model across these accelerators, and coordinating the communication between them, remains part of our future work

3.6 Chapter Summary

In this chapter, we proposed and evaluated a complete hybrid system for accelerating DNN inference and introduced an electro-photonic accelerator, ADEPT. We demonstrated that effectively accelerating DNN inference with photonics requires a tight interplay between photonic compute units for GEMM operations, electronic logic units for non-GEMM operations, and memory units. Through the implementation of optimization methods for pipelining operations and data transfers, we showed that it is possible to leverage the high throughput of the photonic tensor core without being constrained by slower electronic units. Our evaluation compared the performance of the entire system against a similar system where the photo-core is replaced by an SA, revealing that ADEPT can achieve a 5.7$\times$ improvement in throughput per watt on average among the evaluated state-of-the-art DNNs. Furthermore, ADEPT can achieve 6.8$\times$ and 2.5$\times$ better throughput per watt compared to state-of-the-art electronic and photonic DNN accelerators, respectively. Our reported numbers suggest that while the performance expectations should be tempered compared to prior works, photonics technology is still a promising candidate for next-generation AI hardware.

Chapter 4

Unlocking High-Precision in Analog Tensor Cores via the Residue Number System

In this chapter, we target the precision challenges in analog computing, agnostic to the underlying technology, and propose using RNS to overcome this challenge by breaking high-precision operations into multiple low-precision operations. We apply this RNS-based approach to DNN inference and training to build high-precision, energy-efficient analog tensor cores that provide high DNN accuracy. We also explore fault tolerance in RNS-based analog tensor cores and use RRNS error-correcting codes to protect the computation against errors and noise present in analog hardware[1].

4.1 Precision Challenges in Analog Computing

Independent of the technology, in an analog MVM core, inputs and weights in a DNN layer are passed through DACs and encoded in an analog property (e.g., phase, amplitude, etc.). After the analog dot products are performed, the output data are passed through ADCs. Figure 4·1 illustrates this dataflow in a conventional analog core performing a single MVM operation. The precision of the analog operation is determined by (1) the precision of DACs, (2) the precision of ADCs, and (3) the SNR during the analog operations.

A dot product between b_{in}-bit input and b_w-bit weight—both h-long vectors and encoded by DACs—results in $b_{\text{out}}{=}b_{\text{in}}{+}b_w{+}\log_2(h){-}1$ bits of information. For example,

[1]Portions of this chapter were published previously in Demirkiran, Cansu, et al. "A blueprint for precise and fault-tolerant analog neural networks." Nature Communications 15.1 (2024): 5098. DOI:10.1038/s41467-024-49324-8

for 8-bit DACs, the output will require more than 16 bits, necessitating an ADC with $b_{\text{ADC}} \geq 16$ to ensure no information loss. Figure 4·2(a) shows the approximate energy consumption per conversion for different bit precision in DACs and ADCs (Murmann, 2021). As shown in the figure, ADC energy consumption is significantly higher—by two orders of magnitude—than DAC energy consumption. Furthermore, ADC energy consumption increases exponentially with bit precision, approximately $4\times$ energy per conversion for each additional bit. For the aforementioned 8-bit example, a single A-to-D conversion would require ≥ 1 nJ energy. Considering the low energy consumption of MAC operations performed in the analog domain (tens-to-hundreds of fJ/MAC), high-precision ADCs can easily dominate the total energy consumption. Furthermore, as bit precision increases, the required SNR during analog operations also increases exponentially, leading to higher power demands and limiting the precision achievable in analog cores.

As a result, energy-efficient analog accelerator designs typically employ ADCs with lower precision than b_{out} and only capture the b_{adc} most significant bits (MSBs) from the b_{out} bits of each partial output. Figure 4·2(b) shows the impact of this approach on inference accuracy in ResNet-50 CNN (He et al., 2016) when classifying the ImageNet dataset (Krizhevsky et al., 2012). The figure shows that this approach causes accuracy degradation, especially as the vector size h increases. Essentially, to efficiently execute large DNNs using analog accelerators, finding a better way to achieve high accuracy is crucial than simply increasing the bit precision of the data converters.

4.2 RNS-Based Analog DNN Computation

The RNS uses n co-prime integer moduli to represent numbers as a set of residues. These residues are calculated by performing a modulo operation on the integer in the BNS using the selected moduli set (See Section 2.3.1) RNS is closed under addition and multiplication operations, allowing for an MVM operation to be performed in the RNS space. Using the

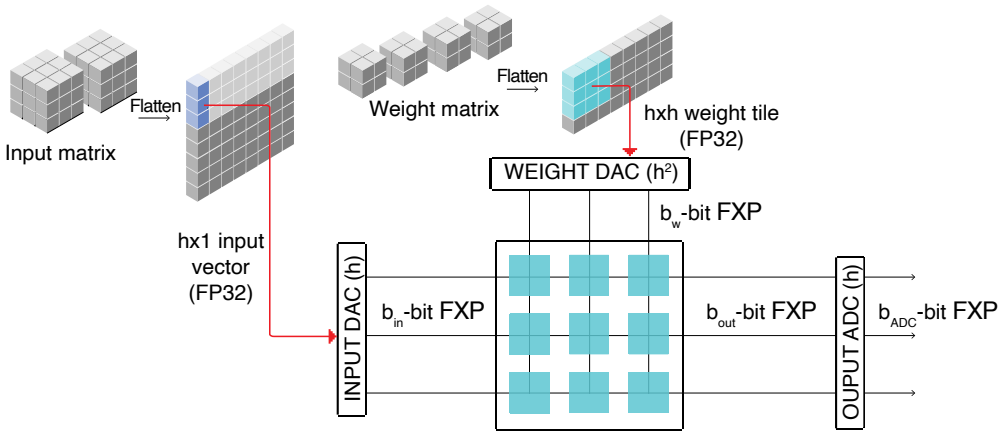

Figure 4·1: Dataflow for a conventional analog core.

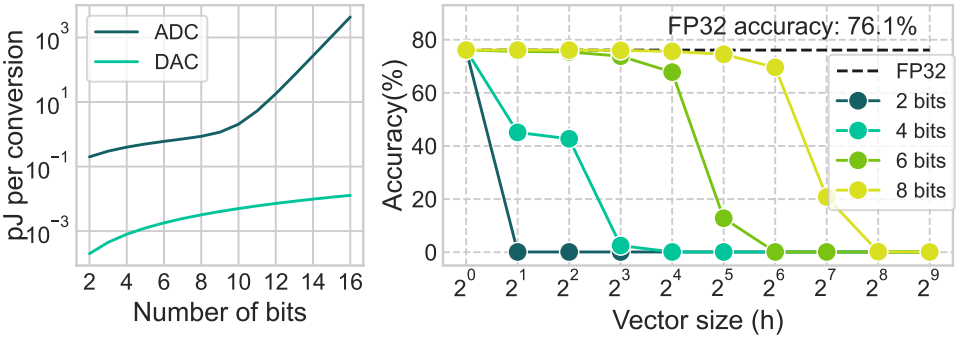

Figure 4·2: (a) Energy consumption per conversion for DACs and ADCs. (b) Accuracy degradation in ResNet50 on Imagenet in a conventional analog core.

RNS, Eq. (2.9) can be rewritten as:

$$X^{(\ell+1)} = f^{(\ell)}\left(\mathrm{CRT}\left(\left|\left|W^{(\ell)}\right|_{\mathcal{M}}\left|X^{(\ell)}\right|_{\mathcal{M}}\right|_{\mathcal{M}}\right)\right). \tag{4.1}$$

The same approach applies to Eqs. (2.10) and (2.11) in the backward pass.

A moduli set $\mathcal{M} = \{m_1, m_2, \ldots m_n\}$ must ensure that the outputs of the RNS operations stay in the RNS range, i.e.,

$$\log_2 M \geq b_{\text{out}} = b_{\text{in}} + b_w + \log(h) - 1 \tag{4.2}$$

Table 4.1: Data and data converter precision in RNS-based, LP FXP, and HP FXP analog cores.

	RNS-based Core (This work)					LP FXP Core				HP FXP Core		
b_{in}, b_w	b_{dac}	$\log_2\mathcal{M}$	b_{adc}	Moduli Set ($\mathcal{M}$)	RNS Range (M)	b_{dac}	b_{out}	b_{adc}	Lost LSBs	b_{dac}	b_{out}	b_{adc}
4	4	4	4	$\{15,14,13,11\}$	$\simeq 2^{15}$	4	14	4	10	4	14	14
5	5	5	5	$\{31,29,28,27\}$	$\simeq 2^{19}$	5	16	5	11	5	16	16
6	6	6	6	$\{63,62,61,59\}$	$\simeq 2^{24}$	6	18	6	12	6	18	18
7	7	7	7	$\{127,126,125\}$	$\simeq 2^{21}$	7	20	7	13	7	20	20
8	8	8	8	$\{255,254,253\}$	$\simeq 2^{24}$	8	22	8	14	8	22	22

should be guaranteed for a dot product between h-long b_{in}-bit input and b_w-bit weight vectors. This constraint prevents overflow during RNS operations.

The selection of $\mathcal{M}$, constrained by Eq. (4.2), has a direct impact on the precision and the energy efficiency of the RNS-based analog core. Table 4.1 compares RNS-based analog GEMM cores with example moduli sets and regular FXP analog GEMM cores with various bit precision. Here, we show two cases for the regular FXP representation: (1) the low-precision (LP) case where $b_{\text{out}} > b_{\text{adc}} = b_{\text{dac}}$, and (2) the high-precision (HP) case where $b_{\text{out}} = b_{\text{adc}} > b_{\text{dac}}$. It is important to note that all three types of analog cores represent data as FXP numbers. We refer to a conventional analog core that performs computations using the standard representation as a regular FXP core. Conversely, an RNS-based core refers to an analog core that executes computations on FXP residues.

The LP approach uses low-precision ADCs causing $b_{\text{out}} - b_{\text{adc}}$ bits of information loss in every dot product. In contrast, the HP approach uses high-precision ADCs to prevent this loss. For the RNS-based core, we picked $b_{\text{in}} = b_w = b_{\text{adc}} = b_{\text{dac}} = \lceil \log_2 m_i \rceil \equiv b$ for ease of comparison against the FXP cores. Table 4.1 shows example moduli sets that are chosen to guarantee Eq. (4.2) for $h = 128$ while keeping the moduli under the chosen bit-width b. For n moduli with bit-width of b, M covers $\approx n \cdot b$ bits of range at the output. In Table 4.1, h is chosen to be 128 as an example considering the common layer sizes in the evaluated MLPerf (Inference: Datacenter) benchmarks, providing high throughput with

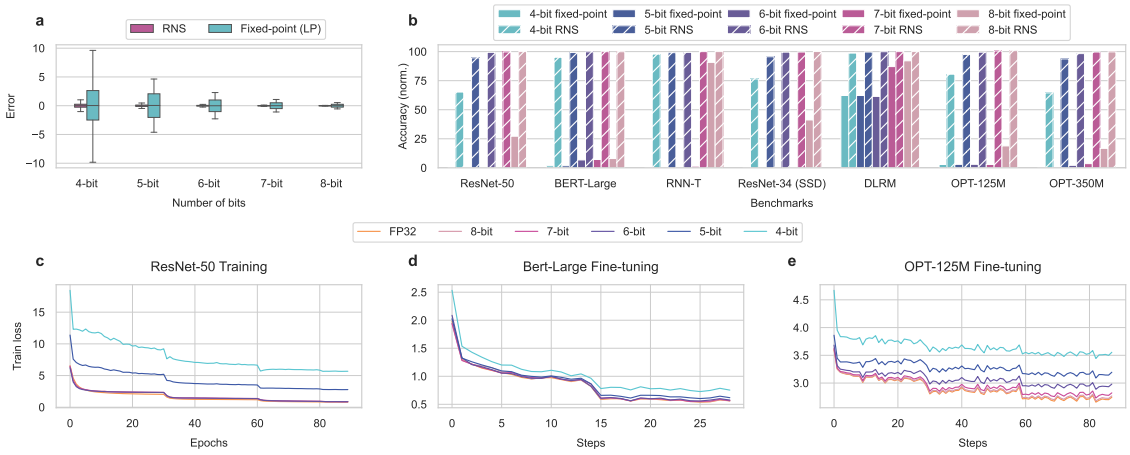

Figure 4·3: (a) The distribution of average error observed at the output of a dot product performed with the RNS-based analog approach (pink) and the LP regular FXP analog approach (cyan). Error is defined as the distance from the result calculated in FP32. The experiments are repeated for $10,000$ randomly generated vector pairs with a vector size of $h = 128$. The center lines of the boxes represent the median. The boxes extend between the first and the third quartile of the data, while whiskers extend $1.5\times$ of the inter-quartile range from the box. (b) Inference accuracy of regular FXP (LP) and RNS-based cores (See Table 4.1) on MLPerf (Inference: Datacenters) benchmarks. The accuracy numbers are normalized by the accuracy achieved in FP32. (c-e) Loss during training for FP32 and the RNS-based approach with varying moduli bit-width. ResNet-50 (c) is trained from scratch for 90 epochs. BERT-Large (d) and OPT-125M (e) are fine-tuned from pre-trained models for 2 and 3 epochs, respectively.

high utilization of the GEMM core.

Figure 4·3a compares the error (relative to the FP32 results) observed when performing dot products using both the RNS-based core and the LP FXP core with the same bit precision. Both cores employ the configurations listed in Table 4.1 for the example vector size $h = 128$. The larger absolute error observed in the LP FXP case highlights the impact of the abovementioned information loss due to $b_{\text{adc}} < b_{\text{out}}$. The HP FXP scenario is not shown, as the observed error is equivalent to that of the RNS case.

Figure 4·3b compares the inference accuracy in MLPerf (Inference: Datacenters) benchmarks (Reddi et al., 2020) and OPT (Zhang et al., 2022a) (a transformer-based large lan-

Table 4.2: MLPerf (Inference: Datacenters) benchmarks (Reddi et al., 2020).

DNN	Task	Dataset
ResNet-50	Image classification	ImageNet (Deng et al., 2009)
BERT-Large	Question answering	SQuADv1.1 (Rajpurkar et al., 2016)
RNN-T	Speech recognition	Librispeech (Panayotov et al., 2015)
ResNet-34 (SSD)	Object detection	MS COCO (Lin et al., 2014)
DLRM	Recommendation	1TB Click Logs (Zhao et al., 2021)
OPT-125M	Language Modeling	Wikitext (Merity et al., 2016)
OPT-350M	Language Modeling	Wikitext (Merity et al., 2016)

Table 4.3: Validation accuracy results after training/fine-tuning.

Precision	ResNet-50 Acc.(%)	BERT-Large F1 Score (%)	OPT-125M Acc.(%)/PPL
FP32	75.80	91.03	43.95/19.72
8-bit	75.77	90.98	43.86/20.00
7-bit	75.68	90.97	43.59/20.71
6-bit	75.13	90.85	42.79/22.62
5-bit	59.72	90.81	41.45/26.17
4-bit	42.15	89.66	38.64/35.65

guage model (LLM)) when run on an RNS-based analog core and a FXP (LP) analog core. The HP FXP analog core is not shown as its accuracy is the same as the RNS-based core. The evaluated DNNs, their corresponding tasks, and the datasets are listed in Table 4.2. Figure 4·3b shows that the RNS-based approach significantly ameliorates the accuracy drop caused by the low-precision ADCs used in the LP FXP approach for all evaluated DNNs. By using the RNS-based approach, it is possible to achieve $\geq$99% of FP32 accuracy (this cut-off is defined in the MLPerf benchmarks (Reddi et al., 2020)) for all evaluated benchmarks when using residues with 6 bit precision. This number can be lowered to 5 bits for BERT-Large and RNN-T and 4 bits for DLRM.

Besides its success in DNN inference, the RNS-based approach opens the way for analog computing to be used in tasks that require higher precision than DNN inference such as

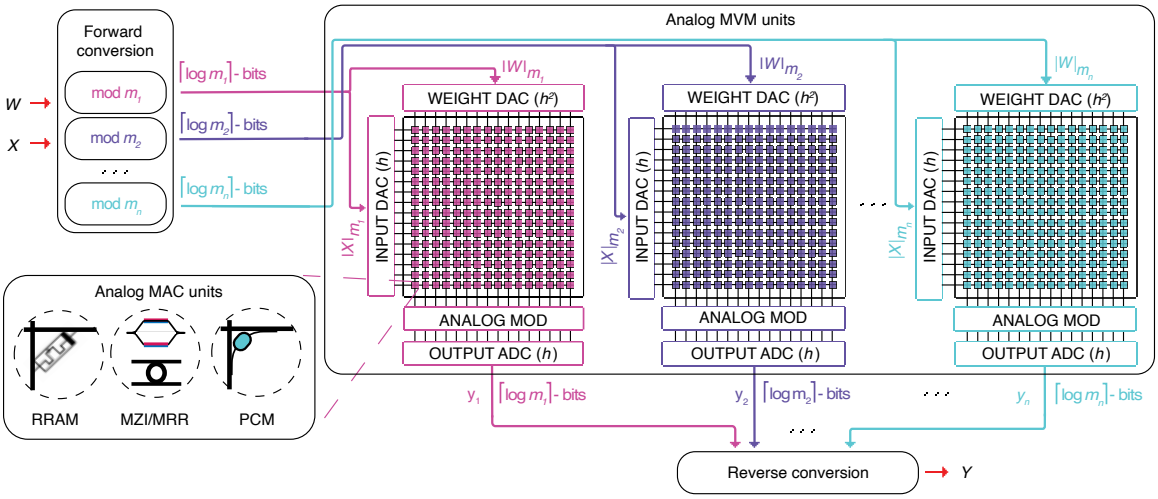

Figure 4·4: RNS-based analog GEMM dataflow. The operation is shown for a moduli set $\mathcal{M} = \{m_1, \ldots, m_n\}$. The n $h \times h$ analog MVM units are represented as generic blocks for n moduli. The dataflow is agnostic to the underlying analog technology.

DNN training. Figures 4·3(c-e) show the loss calculated during DNN training/fine-tuning. Table 4.3 reports the validation accuracy after FP32 and RNS-based low-precision training. Here, the GEMM operations during the forward and backward passes of training are performed using the same methodology as in inference, with weight updates carried out in FP32. These experiments reveal that our approach can achieve $\geq$99% FP32 validation accuracy after training ResNet-50 from scratch using only 6-bit moduli and fine-tuning BERT-Large and OPT-125M by using 5-bit and 7-bit moduli, respectively. These results are particularly promising, as previous attempts at analog DNN hardware using the LP FXP approach fail to successfully demonstrate the training of state-of-the-art DNNs due to the limited precision inherent in this method.

Figure 4·4 illustrates the dataflow of the RNS-based analog core when performing MVM as part of the DNN inference/training. An input vector X and a weight matrix W to be multiplied in the MVM unit are first mapped to signed integers. To mitigate the quantization effects, X and each row in W are scaled by an FP32 scaling factor that is unique to the vector (See Section 4.5). The signed integers are then converted into RNS residues through

modulo operation (i.e., forward conversion). By construction, each residue is within the range of $[0, m_i)$. To achieve the same throughput as a FXP analog core, the RNS-based analog core with n moduli requires using n analog MVM units—one for each modulus—and running them in parallel. Each analog MVM unit requires a set of DACs for converting the associated input and weight residues into the analog domain. The MVM operations are followed by an analog modulo operation on each output residue vector. Thanks to the modulo operation, the output residues—to be captured by ADCs—are reduced back to the $[0, m_i)$ range. Therefore, a bit precision of $\lceil \log_2 m_i \rceil$ is adequate for both DACs and ADCs to perform input and output conversions without any information loss. The output residues are then converted back to the standard representation in the digital domain using Eq. (2.18) to generate the signed-integer output vector, which is then mapped back to an FP32 final output Y. The non-linear function f (e.g., ReLU, sigmoid, etc.) is then applied digitally in FP32.

4.3 RRNS for Fault Tolerance

Analog compute cores are highly sensitive to noise. With RNS, even minor errors in the residues can lead to significant errors in the corresponding integer values they represent. The RRNS (James and Pe, 2015; Yang and Hanzo, 2001a; Yang and Hanzo, 2001b) can detect and correct errors—making the RNS-based analog core fault tolerant. RRNS uses a total of $n+k$ moduli: n non-redundant and k redundant. An RRNS$(n+k, n)$ code can detect up to k errors and can correct up to $\lfloor \frac{k}{2} \rfloor$ errors. In particular, the error in the codeword (i.e., the $n+k$ residues representing an integer in the RRNS space) can be one of the following cases:

- Case 1: Fewer than $\lfloor \frac{k}{2} \rfloor$ residues have errors—thereby they are correctable,
- Case 2: Between $\lfloor \frac{k}{2} \rfloor$ and k residues have errors or the codeword with more than k errors does not overlap with another codeword in the RRNS space—thereby the error

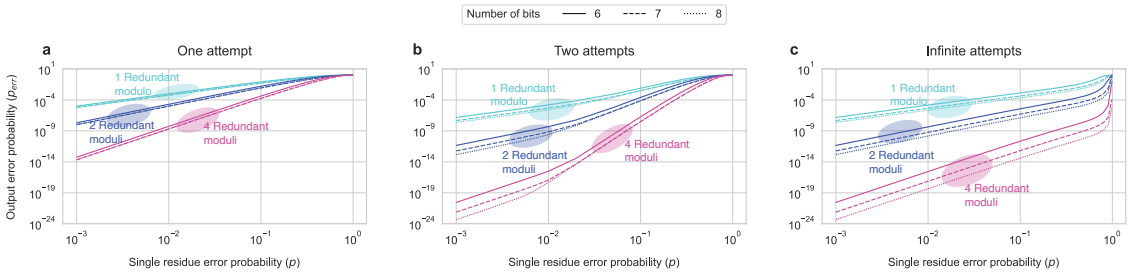

Figure 4·5: Calculated output error probability ($\mathbf{p}_{\text{err}}$) versus single residue error probability (**p**).**a-c** p_{err} for one (**a**), two (**b**), and infinite (**c**) error correction attempts and a varying number of redundant moduli (k).

is detectable,

- Case 3: More than k residues have errors and the erroneous codeword overlaps with another codeword in the RRNS space—thereby the error goes undetected.

Errors are detected by using majority logic decoding wherein we divide the total $n+k$ output residues into $\binom{n+k}{n}$ groups with n residues per group and compare the results obtained from each group. If more than 50% of the groups have the same result, then the generated codeword is assumed correct. This either corresponds to Case 1 where the result is actually correct or Case 3, where the erroneous codeword generated by the majority of the groups overlaps with another codeword. The latter situation leads to an incorrect majority among the groups causing the error to go undetected. In contrast, not having a majority indicates that the generated codeword is erroneous and cannot be corrected. This corresponds to Case 2. In this case, the detected errors can be eliminated by repeating the calculation. One simple way of performing majority logic decoding in this context is to convert the residues in each $\binom{n+k}{n}$ group back to the standard representation via CRT to generate an output value for each group and compare the results. To optimize the hardware performance of this error detection process, more efficient base-extension-based algorithms (Shenoy and Kumaresan, 1989) instead of CRT can be applied.

The final error probability in an RRNS code is determined by the non-correctable er-

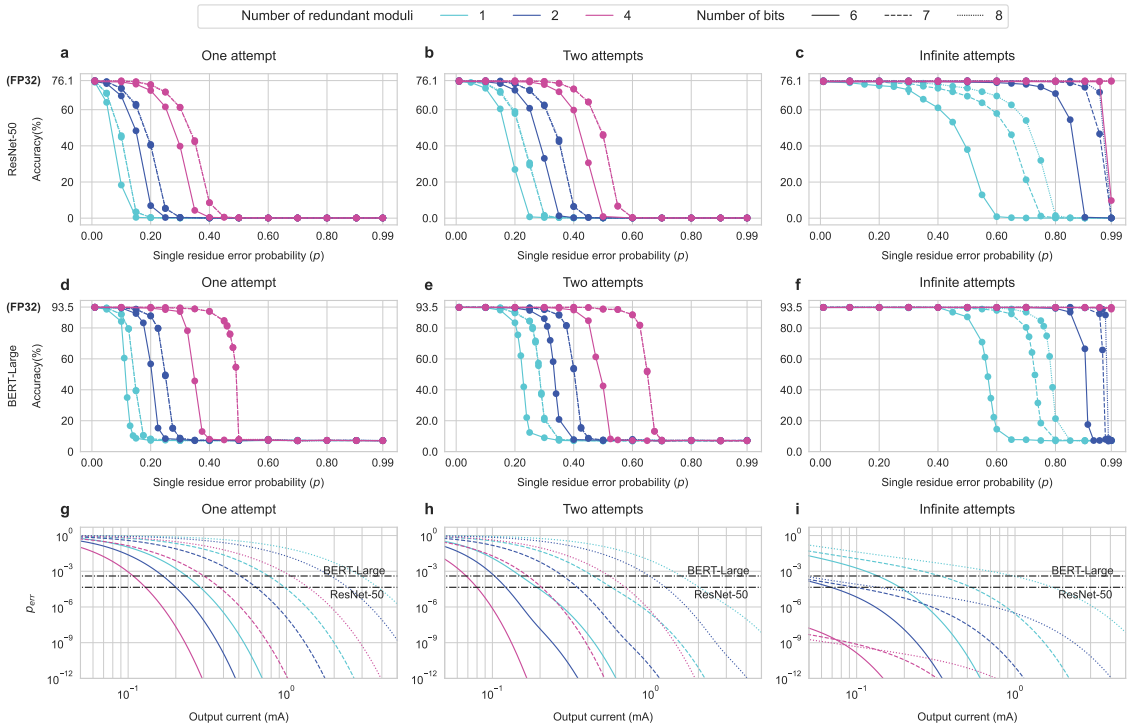

Figure 4·6: (a-f) The plots show ResNet-50 (a-c) and BERT-Large (d-f) inference accuracy under varying p for RRNS with one (a and d), two (b and e), and infinite (c and f) error correction attempts and a varying number of redundant moduli (k). (g-i) p_{err} caused by shot and thermal noise versus the output current at the photodetector in an analog photonic accelerator for RRNS with one (g), two (h), and infinite (i) error correction attempts and varying k. The horizontal black lines show the cut-off points where larger p_{err} starts degrading the accuracy for the evaluated DNNs (i.e., ResNet-50 and BERT-Large).

ror probability observed in the residues. The overall error rate is also influenced by the selected moduli set and the number of correction attempts made for the detected errors (See Appendix A.1). Let p_c, p_d, and p_u be the probabilities of Cases 1, 2, and 3 occurring, respectively, when computing a single output. Overall, $p_c + p_d + p_u = 1$. For a single attempt (i.e., $R = 1$), the probability of producing the incorrect output integer is $p_{\text{err}}(R = 1) = 1 - p_c = p_u + p_d$. It is possible to repeat the detected erroneous calculations R>1-times to minimize the amount of uncorrected error at the expense of increasing com-

pute latency and energy. In this case, the probability of having an incorrect output after R attempts of error correction is

$$p_{\text{err}}(R) = 1 - p_c \sum_{r=0}^{R-1} (p_d)^r. \tag{4.3}$$

As the number of attempts increases, the output error probability decreases and converges to $\lim_{R\to\infty} p_{\text{err}}(R) = p_u/(p_u + p_c)$.

Figure 4·5 shows how p_{err} changes with the error probability in a single residue (p) for different numbers of redundant moduli (k) and attempts (R) and moduli sets with different bit-widths. Broadly, as p increases, the p_{err} tends to 1. For a given number of R, a higher bit precision and higher k results in a lower p_{err}. For a fixed k and a fixed number of bits per moduli, p_{err} decreases as R increases.

Figure 4·6 investigates the impact of noise on the accuracy of two large and important MLPerf benchmarks—ResNet-50 and BERT-Large—when error correction is implemented using RRNS. The two models show similar behavior: increasing k and increasing R decrease p_{err} for the same p, enabling to sustain high accuracy for higher p. ResNet-50 requires ∼3.9 GigaMAC operations (GOp) per inference on a single input image. For a 128×128 MVM unit, inferring an ImageNet image through the entire network involves computing ∼29.4M partial output elements. Therefore, the expected transition point from an accurate network to an inaccurate network is at $p_{\text{err}} =\leq 1/29.4\text{M} = 3.4 \times 10^{-8}$. This p_{err} transition point is $\leq 1/358.6\text{M} = 2.8 \times 10^{-9}$ for BERT-Large. Figure 4·6 shows, however, that the evaluated DNNs are more resilient to noise than expected: they can tolerate higher p_{err} while maintaining good accuracy. The accuracy of ResNet-50 only starts degrading (below 99% FP32) when $p_{\text{err}} \approx 4.5 \times 10^{-5}$ (1000× higher than the estimated value) on average amongst the experiments shown in Figure 4·6. This transition probability is $p_{\text{err}} \approx 4 \times 10^{-4}$ for BERT-Large (on average $100,000\times$ higher than the estimated value).

In analog hardware, expected p and p_{err} depend on the underlying analog technology, device characteristics, and many other device-specific factors. As an example, we examine a photonics-based RNS analog accelerator design that is constrained by thermal and shot noise. In such a system, the noise can be modeled as a Gaussian distribution that is additive to the output value, i.e., $\Sigma_j x_j w_j + \mathcal{N}(0,1)\sigma_{\text{noise}}$ for a dot product (Garg et al., 2023). For an analog core where output is captured as an analog current, let us define the maximum achievable current as I_{out}, representing the largest output value. A higher I_{out} requires a higher input power, but results in a higher SNR and lower p_{err}, creating a tradeoff between power consumption and noise tolerance.

Without any redundant moduli ($k = 0$), $I_{\text{out}} \leq 1$ mA is adequate to prevent accuracy loss due to analog noise in both DNNs (6-bit case). This cut-off is at 2 mA and 8 mA for 7-bit and 8-bit cases, respectively. For instance, for a photonic system, $I_{\text{out}} \leq 1$ mA requires $\sim$1 mW (0 dBm) output power (for a photodetector with 1 A/W responsivity)—which is feasible assuming a 10 dBm laser source and 10 dB loss along the optical path.

The required I_{out} can be further lowered by using RRNS. Figs. 4·6(g-i) shows the relationship between I_{out} and the expected p_{err} for different RRNS. For a smaller number of bits and a higher k, a lower I_{out} is adequate to stay under the cut-off p_{err} for the evaluated DNNs. For example, a 6-bit RRNS with $k = 1$ requires $I_{\text{out}} = 0.1$ mA for a single error correction attempt as against the $k = 0$ case where $I_{\text{out}} = 1$ mA is needed to avoid accuracy loss due to analog noise. The required I_{out} similarly decreases with the increasing R. Please see Noise Modeling under Methods for details.

4.4 Energy and Area Efficiency

Figure 4·7a shows the energy consumption of DACs and ADCs per dot product for the RNS-based and FXP (LP and HP) analog hardware configurations. To achieve the same throughput as the (LP/HP) FXP cores, the RNS-based core with n moduli must use n sets

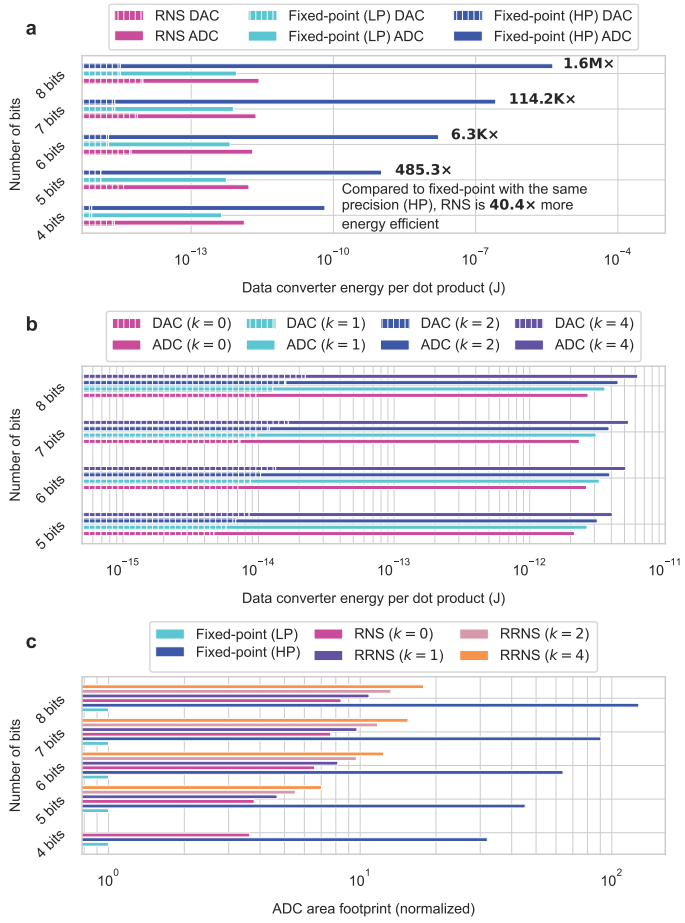

Figure 4·7: (a-b) Energy consumption of DACs and ADCs per dot product for the RNS-based and the regular FXP (a) and the RNS and RRNS-based analog approaches (b). (c) Normalized area of ADCs for the FXP, RNS, and RRNS-based approaches.

of DACs and ADCs. This makes the energy consumption of the RNS-based core $n\times$ larger compared to the LP FXP approach. However, the LP FXP approach with low-precision ADCs experiences information loss in the partial outputs and hence has lower accuracy.

The RNS-based and HP FXP approaches provide the same bit precision (i.e., the same DNN accuracy). Yet, using the RNS-based approach is orders of magnitude more energy-efficient than the HP FXP approach. This is mainly because of the high cost of high-precision ADCs required to capture the full output in the HP FXP approach. ADCs domi-

nate the energy consumption with approximately three orders of magnitude higher energy usage than DACs with the same bit precision. In addition, energy consumption in ADCs increases exponentially with increasing bit precision (Murmann, 2021). This favors using multiple DACs and ADCs with lower precision in the RNS-based approach over using a single high-precision ADC. The RNS-based approach briefly provides a sweet spot between the LP and HP FXP approaches without compromising accuracy and energy efficiency.

Figure 4·7b shows the energy consumption of DACs and ADCs when RRNS is used. The plot only shows the 5-to-8-bit cases as there are not enough co-prime moduli smaller than 15 to use RRNS for the 4-bit case. RRNS results in an approximately linear increase in energy consumption as the number of moduli $(n+k)$ increases. The compute time does not increase with the increasing k as operations for different moduli are independent and can be performed in parallel.

The area footprint of data converters has a weaker correlation with their bit precision than their energy consumption. In a study by Verhelst and Murmann in 2012 (Verhelst and Murmann, 2012), the authors observed that the area footprint of ADCs is proportional to $2^{\alpha}b$ where $\alpha \in [0.11, 1.07]$ depending on the type of the ADC, and is $\alpha = 0.5$ when all ADC types are considered. Assuming the same technology node is used, Figure 4·7c shows the normalized area footprint of ADCs for the LP and HP FXP, RNS ($k = 0, n$ ADCs per dot product) and RRNS ($k > 0, n+k$ ADCs per dot product) approaches. While the area footprint of the RNS and RRNS-based approaches are higher than the LP FXP approach, they have a smaller area footprint than the HP FXP approach for all bit precisions. In addition, the same study points out that the sampling frequency of ADCs is independent of the area footprint. Therefore, in the RNS and RRNS approaches, instead of having multiple ADCs per dot product, one can use a single and faster ADC and perform multiple conversions using the same ADC to achieve the same throughput with better area efficiency.

4.5 Evaluation Methodology

In this section, we describe our evaluation methodology for data converter energy calculation and accuracy modeling.

4.5.1 Data Converter Energy Estimation

The DAC and ADC energy numbers in Figs. 4·7a and 4·7b are estimated by using equations formulated by Murmann (Murmann, 2021; Murmann, 2024). The energy consumption of a DAC per b-bit conversion is

$$E_{\text{dac}} = b^2 C_u V_{\text{DD}}^2, \tag{4.4}$$

where $C_u = 0.5$ fF is a typical unit capacitance and V_{DD} = 1V is the supply voltage (Murmann, 2021). The energy consumption of an ADC per b-bit conversion can be estimated as

$$E_{\text{adc}} = k_1 b + k_2 4^b. \tag{4.5}$$

For calculating the coefficients k_1 and k_2, we used the data from the ADC survey collected by Murmann (Murmann, 2024). The dataset includes all ADC literature published in the two main venues of the field, the International Solid-State Circuits Conference (ISSCC) and the VLSI Circuit Symposium, between the years 1997 and 2023. We removed the data points with a sampling frequency lower than 1 GHz as our design requires high-speed data converters. k_1 is calculated as the average of the three samples with the smallest E_{ADC}/b and k_2 as the average of the three samples with the smallest $E_{\text{ADC}}/4^b$ among the available data points (Murmann, 2024).

4.5.2 Accuracy Modeling

Both RNS-based and conventional FXP analog cores are modeled using PyTorch to estimate inference and training accuracy. Convolution, linear, and batched matrix multiplication (BMM) layers are executed as GEMM operations, which are computed tile-by-tile

through a series of tiled-MVM operations, determined by the tile size of the analog core. Each input, weight, and output tile is quantized according to the specified bit precision.

Before quantization, the input vectors and weight tiles are first dynamically scaled at runtime to mitigate the quantization effects as follows: For an $h \times h$ weight tile $\mathcal{W}_t$, we denote each row vector as $\mathcal{W}_{rt}$ where the subscript r stands for the row and t for the tile. Similarly, an input vector of length h is denoted as $\mathcal{X}_t$ where t indicates the tile. Each weight row $\mathcal{W}_{rt}$ shares a single FP32 scale $s^w_{rt} = \max(|\mathcal{W}_{rt}|)$ and each input vector $\mathcal{X}_t$ shares a single FP32 scale $s^x_t = \max(|\mathcal{X}_t|)$. h scales per $h \times h$ weight tile and one scale per input vector, in total $h+1$ scales, are stored for each tiled-MVM operation. The tiled MVM is performed between the scaled weight and input vectors, $\widehat{\mathcal{W}_{rt}} = \mathcal{W}_{rt}/s^w_{rt}$ and $\widehat{\mathcal{X}_t} = \mathcal{X}_t/s^x_t$, respectively, to produce $\widehat{Y}_{rt} = \widehat{\mathcal{W}_{rt}}\widehat{\mathcal{X}_t}$. The output $\widehat{Y}_{rt}$ is then quantized (if required) to resemble the output ADCs and multiplied back with the appropriate scales so that the actual output elements $Y_{rt} = \widehat{Y}_{rt} \cdot s^w_{rt} \cdot s^x_t$ are obtained.

Here, the methodology is the same for RNS-based and regular FXP cores. For the RNS-based case, in addition to the description above, the quantized input and weight integers are converted into the RNS space before the tiled-MVM operations. MVMs are performed separately for each set of residues and are followed by a modulo operation before the quantization step. The output residues for each tiled MVM are converted back to the standard representation using the CRT.

To accurately model the quantization during forward and backward passes, all GEMM operations (i.e., convolution, linear, and BMM layers) are sandwiched between an input operation O_{in} and an output operation O_{out}. This makes the operation order O_{in}-GEMM-O_{out} during the forward pass, and O_{out}-GEMM-O_{in} in the backward pass. O_{in} quantizes the input and weight tensors in the forward pass and is a null operation in the backward pass. In contrast, O_{out} is a null operation in the forward pass and quantizes the activation gradients in the backward pass. In this way, the quantization is always performed before the GEMM

operation. The optimizer (i.e., SGD or Adam) is modified to keep a copy of the FP32 weights to use during the weight updates. Before each forward pass, the FP32 weights are copied and stored. After the forward pass, the quantized model weights are replaced by the previously stored FP32 weights before the step function so that the weight updates are performed in FP32. After the weight update, the model parameters are quantized again for the next forward pass. This high-precision weight update step is crucial for achieving high accuracy in training.

We trained ResNet-50 from scratch by using SGD optimizer for 90 epochs with a momentum of 0.9 and a learning rate starting from 0.1. The learning rate was scaled down by 10 at epochs 30, 60, and 80. We fine-tuned BERT-Large and OPT-125M from the implementations available in the Huggingface transformers repository (Huggingface,). We used the Adam optimizer for both models with the default settings. The script uses a linear learning rate scheduler. The learning rate starts at $3e{-}05$ and $5e{-}05$ and the models are trained for 2 and 3 epochs, respectively for BERT-Large and OPT-125M.

Noise Analysis

In analog hardware, both shot noise and thermal noise can be modeled as Gaussian distributions, i.e., $I_{\text{shot}} \sim \sqrt{2q_e\Delta f I_{\text{out}}}\mathcal{N}(0,1)$ where q_e is the elementary charge, Δf is the bandwidth, I_{out} is the output current of the analog dot product and $I_{\text{thermal}} \sim \sqrt{\frac{4k_B\Delta f T}{R_{\text{TIA}}}}\mathcal{N}(0,1)$ where k_B is the Boltzmann constant, T is the temperature, and R_{TIA} is the feedback resistor of the transimpedance circuitry.

For a modulus m, the consecutive output residues represented in the analog output current should be at least I_{out}/m apart from each other to differentiate m distinct levels. An error occurs in the output residue when $\sqrt{I_{\text{shot}}^2 + I_{\text{thermal}}^2} \geq I_{\text{out}}/2m$ as the residue will be rounded to the next integer otherwise. Therefore, the error probability in a single residue can be calculated as $p = P(\sqrt{2q_e\Delta f I_{\text{out}} + \frac{4k_B\Delta f T}{R_{\text{TIA}}}}\mathcal{N}(0,1) \geq I_{\text{out}}/2m)$. We used $\Delta f = 5$ GHz, $T = 300$ K and $R_{\text{TIA}} = 200\Omega$ as typical values in the experiments shown in Fig-

ure 4·6(g-i). For a calculated p, $p_{err} = 1-(1-p)^n$ for an n-moduli RNS ($k = 0$). For RRNS ($k > 0$), p_{err} can be obtained using Figure 4·5 or Eq. (4.3).

4.6 Related Work

RNS is a well-explored numeral system that has been used in a variety of applications including digital signal processing (Jenkins, 1980), cryptography (Yen et al., 2003), and DNNs (Samimi et al., 2020; Salamat et al., 2018). RNS-based DNN computation in digital hardware was proposed for improving energy efficiency by breaking numbers into residues with fewer bits. Res-DNN (Samimi et al., 2020) proposes an RNS-based version of the popular DNN accelerator Eyeriss (Chen et al., 2016) and RNS-Net (Salamat et al., 2018) uses a PiM-based design and simplifies RNS operations to PIM-friendly ones. A similar work, DNNARA (Peng et al., 2020), is a nanophotonic (not analog) RNS-based DNN inference accelerator where the authors use 2×2 optical switches to build a network and manipulate the route of the light through this network to perform multiplication and additions using a one-hot encoded mapping. While all three works are similar to our study in terms of using RNS for DNN inference, we are the first to propose using RNS in the context of analog DNN computation. In addition, these accelerators all propose fully RNS-based dataflows without switching back and forth between RNS and BNS. Although this approach of staying in the RNS domain removes the cost of the RNS-BNS conversions, it requires periodically performing overflow detection and ranging operations in the RNS domain to preserve the integrity of RNS operations. More importantly, these fully RNS-based computations force the end-to-end DNN to be computed in FXP arithmetic. Performing nonlinear operations in the RNS domain requires using approximations (e.g., Taylor series expansion) to reduce nonlinear operations into multiply and add operations. These approximations in nonlinear functions cause information loss and demand higher data precision. As a result, these previous works use 16-bit or higher precision to represent data to achieve high accu-

racy and their proposals are limited to DNN inference. In our approach, switching back and forth between RNS and BNS for each MVM operation allows us to control the precision of nonlinear operations (which are performed on digital hardware) independently and perform scaling (dynamic quantization) before MVM operations to alleviate the quantization errors at the data converters (See Section 4.5.2). This approach also enables us to perform backpropagation and successfully train DNNs with low-precision arithmetic (7-bit) besides DNN inference. In contrast to the few previous analog DNN training demonstrations(Bandyopadhyay et al., 2023; Filipovich et al., 2022; Pai et al., 2023; Hughes et al., 2018; Zhang et al., 2021) that were limited to very simple tasks (e.g., MNIST classification) and DNNs with a few small layers, our approach can achieve a much higher dynamic range through RNS and can successfully train state-of-the-art DNNs. At last, different from previous works, we analyze the impact of noise on accuracy in RNS-based DNN inference and integrate RRNS to combat the accuracy loss caused by the errors in analog hardware.

4.7 Discussion

The RNS (and the fault-tolerant RRNS) framework proposed in this chapter is agnostic to the analog technology employed. This framework requires GEMM operations to be performed in the RNS space, requiring modular arithmetic in the analog domain. Analog GEMM is well-explored in the literature. Previous works leveraged photonics (Shen et al., 2019; Xu et al., 2021; Tait et al., 2017; Peng et al., 2020; Shiflett et al., 2020; Shiflett et al., 2021), crossbar arrays consisting of RRAM (Yao et al., 2020; Chi et al., 2016; Shafiee et al., 2016; Hu et al., 2016; Tang et al., 2017), switched capacitors (Bankman and Murmann, 2015; Bankman and Murmann, 2016), PCM cells (Feldmann et al., 2021), STT-RAM (Jain et al., 2017; Shi et al., 2020), etc. One can use these methods followed by a modulo operation in the analog domain to perform modular GEMM operations.

The analog modulo operation can be performed electrically or optically. As an electri-

cal solution, one can use ring oscillators, a circuit that generates a continuous waveform by cycling through a series of inverters (Ordentlich et al., 2018), to perform modulo operations. By carefully designing the parameters of the ring oscillator, it is possible to create an output frequency that corresponds to the desired modulus value.

Alternatively, the phase of an optical signal can be leveraged for performing modulo due to the periodicity of phases in optical systems. The optical phase is inherently modular against 2π. By modulating the phase of an optical signal, one can achieve modular MAC operations in the analog domain. We leverage this idea in Chapter 5 to design an RNS-based photonic DNN accelerator.

The moduli sets and tile sizes provided in this chapter are used as examples. For a specific analog technology, a thorough design space exploration should be conducted, taking into account the trade-offs unique to that technology. The design can be optimized for various metrics, such as throughput, energy efficiency, and area efficiency, leading to different optimal configurations. In these varying configurations, the required bit precision for RNS operations may differ from the reported discussed in this chapter.

4.8 Chapter Summary

In this chapter, we addressed the precision challenges in analog computing by employing the RNS, which constructs high-precision operations from multiple low-precision operations. This approach mitigates the need for high-precision data converters and prevents information loss. Our experiments demonstrated that the RNS-based technique can achieve over 99% of FP32 accuracy for DNN inference using only 6-bit and for training with 7-bit FXP arithmetic. These findings suggest that RNS can significantly reduce the energy consumption of analog accelerators without compromising throughput or precision. In addition, we introduced a fault-tolerant dataflow utilizing RRNS to detect and correct errors in analog hardware.

Chapter 5

RNS-based Photonic DNN Training Accelerator Design

In this chapter, we leverage the RNS-based framework proposed in Chapter 4 and design an electro-photonic DNN training accelerator, Mirage. To perform RNS operations, we propose novel photonic modular arithmetic units based on cascaded phase shifters. By combining RNS and photonics, Mirage provides high energy efficiency without compromising precision and can successfully train state-of-the-art DNNs achieving comparable accuracy to FP32 training. Our study shows that on average across several DNNs when compared to systolic arrays, Mirage achieves more than 23.8× faster training and 32.1× lower EDP in an iso-energy scenario and consumes 42.8× lower power with comparable or better EDP in an iso-area scenario[1].

5.1 RNS-Based Dataflow in Mirage

As mentioned in Chapter 4, RNS is closed under addition and multiplication, allowing a GEMM operation to be performed in the RNS space. Figure 5·1 shows the dataflow for a tiled-GEMM operation as part of the forward pass in Mirage. In the RNS space, each matrix is represented by n residue matrices for n moduli. GEMM in the RNS space is then a set of modular GEMM operations—one GEMM per modulus, n GEMMs in total. After the GEMM operations are performed, the resulting n output residue matrices are converted

[1]Portions of this chapter were published previously in Demirkiran, Cansu, et al. "Mirage: An RNS-Based Photonic Accelerator for DNN Training." 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). IEEE, 2024. DOI: 10.1109/ISCA59077.2024.00016

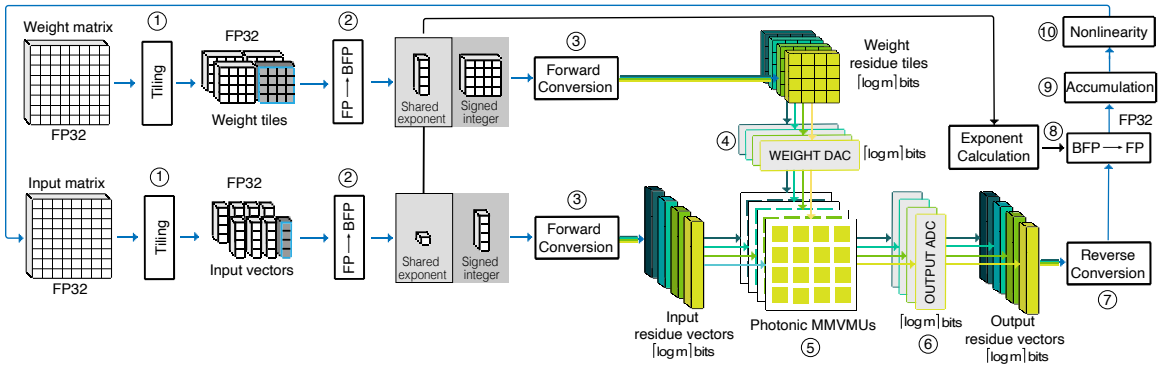

Figure 5·1: Mirage's RNS-based dataflow for a single tiled-MVM operation as part of a forward pass. We show a four-moduli case in this figure as an example.

back to a single output matrix in BNS by using Eq (2.18). The same approach applies to the GEMM operations in the backward pass as stated in Eqs. (2.10) and (2.11).

The dataflow follows these steps:

① The FP32 input and FP32 weight matrices are tiled with a tile size equal to the photonic MVM array size.

② The FP32 values are converted to BFP. In an MVM operation with BFP values, the input vector and each row of the weight tile represent a BFP group[2]. For each group, the largest exponent among the group elements is chosen to be the shared exponent ($e_{\vec{v}}$). Given an $e_{\vec{v}}$ for a group $\vec{v}$ with a group size g, i.e., $e_{\vec{v}} = \max(e_{\vec{v}[i]})\ \forall i \in \{1, ..., g\}$, the mantissae of the group elements are shifted right by the difference between the shared exponent and their original exponent, i.e., $m_{\vec{v}[i]} = m_{\vec{v}[i]} >> (e_{\vec{v}} - e_{\vec{v}[i]})$, where $\vec{v}[i]$ is the i^{th} element of $\vec{v}$. The LSBs of the mantissae are then truncated depending on the number of mantissa bits (b_m).

③ We perform 'forward conversion' to convert the 'b_m+1'-bit signed integers, i.e., sign and mantissa bits of the BFP values, of the input vector and weight tile into the RNS space.

[2]The group size in BFP, denoted as g, the vector size, denoted as h, and the horizontal dimension of the photonic MMVMU (the number of optical MAC units within a row), all signify the count of elements within the vectors subjected to a dot product. From this point onward, we will collectively refer to these three terms as g.

Forward conversion generates n input vectors and n weight tiles in total for n moduli. Here each residue is represented by $\lceil \log_2(m_i) \rceil$ bits where m_i is the i^{th} modulus value. Forward conversion is a modulo operation that can be simplified into a simple shift operation when special moduli sets are used (See Section 2.3.2.

④ Each weight residue tile is passed through $\lceil \log_2(m_i) \rceil$-bit DACs to ensure no information loss and programmed into the MMVMUs. The input vectors do not require DACs thanks to our photonic core design as each individual digit of the binary input is multiplied separately (more details in Section 5.2).

⑤ Analog modular MVM operations are performed in the modular MVM units (MMVMUs) using cascaded phase shifter blocks. The operations are inherently modular as the operands are encoded directly in the amount of phase shift applied to an optical signal (See Section 5.2).

⑥ The outputs are detected by photodetectors and converted into the digital domain by using $\lceil \log_2(m_i) \rceil$-bit ADCs. Here, it is important to note that weight DACs and output ADCs have the same precision. This is because the use of modular arithmetic in the analog domain ensures that the data bit-width does not grow during operations. Therefore, the output can be collected with no information loss at the ADCs with the same bit-width as DACs.

⑦ The collected output residues are converted back to BNS. This 'reverse conversion' can be implemented using Eq. (2.18). Similar to forward conversion, using special moduli sets alleviates the complexity and overhead of this step.

⑧ The exponents of the output vector are digitally calculated in parallel with the analog modular MVM operations. Using the output mantissae and exponents, FP32 values are constructed.

⑨ The partial outputs are accumulated to compose the final GEMM output. The dataflow in Mirage requires the partial outputs to be written to the on-chip memory. For

each partial output, a read-accumulate-write operation is performed.

(10) Steps 2-9 are repeated for each tile in a layer and nonlinearity is then applied to the final GEMM result (digitally in FP32).

Steps 1-10 are repeated for each layer until the forward pass is complete. Input and weight gradients are then calculated in a similar manner where input and weight matrices in the diagram are replaced by the operands in Eqs. (2.10) and (2.11). Once the weight gradients are obtained, the weight values are updated according to Eq. (2.12). Here, we perform all the GEMM operations in BFP, however, we store the weights in FP32 in the memory and perform the weight updates in FP32.

5.2 Modular Arithmetic with Photonics

In Mirage, we perform tiled-MVM operations in the RNS domain, as shown in Figure 5·1, step 5. This step requires a new photonic core design for performing *modular* arithmetic in the analog domain, unlike the conventional photonic cores relying on standard FXP arithmetic. This section describes our novel modular MAC unit and how we build modular dot products and MVMs using this unit.

5.2.1 Modular Multiplication Unit (MMU)

The phase difference between input and output in a typical dual-rail MZM (as shown Figure 5·2(a)) is

$$\Delta\Phi = \frac{VL}{V_{\pi\cdot \mathrm{L}}}, \tag{5.1}$$

where $V_{\pi\cdot \mathrm{L}}$ is the modulation efficiency of the phase shifter and a constant value. $\Delta\Phi$ is then proportional to both the length of the phase shifter, L, and the applied voltage, V. A regular multiplication, i.e., xw, is performed through *amplitude* modulation by adjusting the attenuation caused by the phase shift on the input signal. However, in RNS, a modular multiplication, i.e., $|xw|_m$ is needed. In Mirage, this behavior is achieved through *phase*

modulation. By encoding x and w in L and V, we can obtain a phase shift that is proportional to xw and inherently modular with 2π in the same MZM design, i.e., $\Delta\Phi \propto |xw|_{2\pi}$.

However, L is fixed after fabrication and cannot be changed at runtime. Therefore, we use separate phase shifters for each digit of the binary operand, with the length of each shifter proportional to the weight of the corresponding binary digit, as illustrated in Figure 5·2(b). To encode a b-bit value, we use b phase shifters with lengths 2^0L, 2^1L, ..., $2^{b-1}L$ for each bit from LSB to MSB and control each digit separately. For performing a multiplication, we map one operand (w in this example) to the applied voltage and apply the same voltage to all b digits. We then use the second operand (x) digit-by-digit to turn ON or OFF the voltage on each shifter separately. This mapping requires an AND operation between the first operand and each digit of the second operand, i.e., $V^{(d)}=(wV_0)\wedge x^{(d)}$, $\forall d \in \{0, ..., b-1\}$.

Figure 5·2(b) shows a multiplication between two 3-bit integers where w is encoded in the applied voltage as an analog value while the digits of x control if V is applied to each digit or not. For example, assume x=101 and w=011. In this case, we set $V=wV_0=3V_0$ for all three digits[3]. As the LSB of x, $x^{(0)}=1$ and $1\wedge V=3V_0$, $3V_0$ is applied to the L-long phase shifter. This creates a $3\Phi_0$ phase shift on the optical signal passing through. The second digit of x, $x^{(1)}=0$. As $0\wedge V=0$, no voltage is applied to the $2L$-long phase shifter resulting in no phase shift on the optical signal. Similar to the LSB, $x^{(2)}=1$ and $1\wedge V=3V_0$ voltage is applied to the $4L$-long phase shifter. This causes a $12\Phi_0$ phase shift as the phase shift is proportional to $V \cdot L$. As the same optical signal goes through all the cascaded phase shifters, the sequentially introduced phase shifts are accumulated. By applying opposite signed voltages to the symmetrical arms of the dual-rail setup, a total of $(3+0+12)\Phi_0 = 15\Phi_0 \propto xw$ is applied to the signal ($15/2\ \Phi_0$ from each arm). The observed phase shift, however, is $|15\Phi_0|_{2\pi}$ as the optical phase is modular with 2π.

[3] V_0 represents a unit voltage value that results in a unit phase shift (Φ_0) in a L-long phase shifter. Φ_0 is set to be $2\pi/m$ to perform a modulo m operation, which is explained later in the section. For a given Φ_0, the

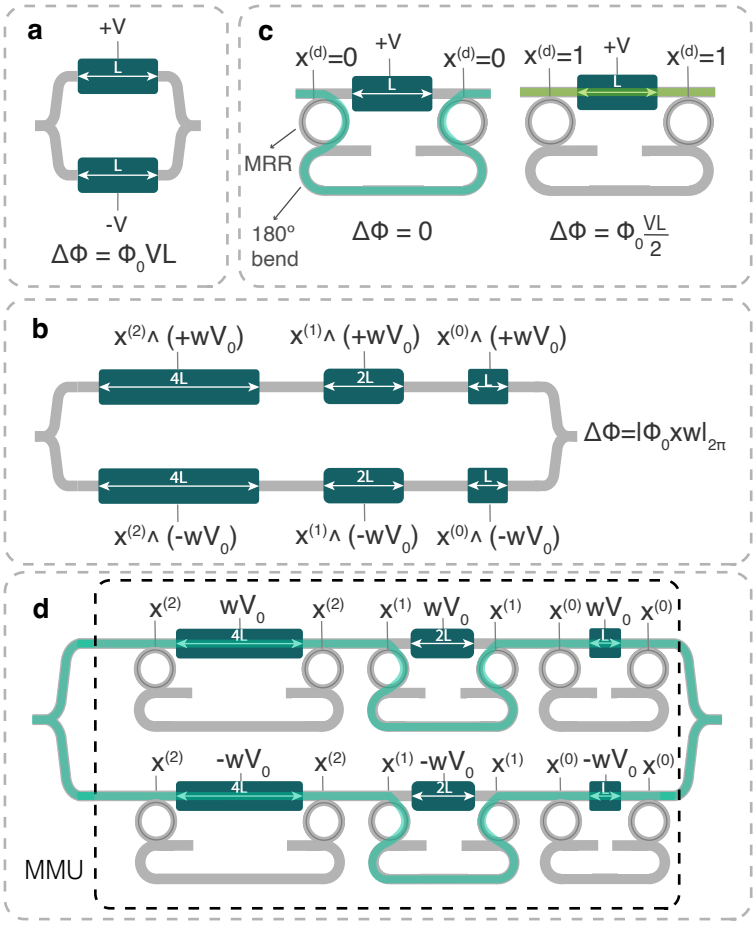

Figure 5·2: (a) Simple MZM with phase shifters with length L and applied voltage V. (b) 3-bit modular multiplication using cascaded phase shifters. (c) Routing light using MRR switches. (d) 3-bit modular multiplication using MRR switches.

By adjusting Φ_0 to be $2\pi/m$, modular arithmetic with arbitrary modulus m instead of 2π, i.e., $|xw|_m$, can be obtained as

$$\Delta\Phi_{\text{total}} = \Big|\Big(\sum_{d=0}^{b-1}(2^d x^{(d)} w\frac{2\pi}{m})\Big)\Big|_{2\pi} = \frac{2\pi}{m}\big|(xw)\big|_m. \tag{5.2}$$

This adjustment is done through the unit applied voltage, i.e., $V_0 = 2V_\pi/m$. The resulting output value ($\Delta\Phi_{\text{total}}$) read at the end of the optical path is then multiplied back by $m/2\pi$ to obtain the output.

absolute values for V_0 and L depend on the $V_{\pi\cdot L}$ of the phase shifters and the maximum available bias voltage.

For a modulus m, both x and w are both integer residues varying between $[0,\ m-1]$, which can be mapped around zero as $[-\lfloor\frac{m-1}{2}\rfloor, \lceil\frac{m-1}{2}\rceil]$. In this case, the maximum output on an MMU is $xw = \lceil\frac{(m-1)^2}{2}\rceil$, requiring a $\Delta\Phi_{\max} = \lceil\frac{(m-1)^2}{2}\rceil\frac{2\pi}{m}$ phase shift. The MMU should be able to reach $\Delta\Phi_{\max}$ when the bias voltage (V_{bias}) is fully applied. This requires a total phase shifter length of

$$L_{\text{total}} = \frac{V_{\pi L}}{V_{\text{bias}}}\frac{\Delta\Phi_{\max}}{\pi}. \tag{5.3}$$

Given a $\Delta\Phi_{\max}$, there is a trade-off between V_{bias} and L_{total} caused by the constant $V_{\pi L}$ of the phase shifters. $V_{\pi L}$ depends on the chosen actuation mechanism of the devices as mentioned in Section 2.2.4.

In Mirage, the input operands of the MMU (x and w) are integer elements of the residue matrices to be multiplied during a tiled-GEMM operation—which is a series of tiled-MVM operations. During the tiled-GEMM operation, one MVM is performed every cycle. For each MVM operation, at least one of the input operands needs to be updated. In the MMU design shown in Figure 5·2(b), an update in x, w, or both, all result in reprogramming the phase shifters *every cycle*. In this case, phase shifters must have high modulation bandwidths ($\geq$GHz) to perform high-speed MAC operations. As mentioned in Section 2.2.4, this high bandwidth in phase shifters can be obtained through free-charge dispersion, however, such devices are typically quite lossy and have relatively high $V_{\pi L}$ values (the lower the better) causing longer device lengths. Using these high-bandwidth phase shifters, one can achieve high-speed operations, but this can easily lead to $\geq$10 dB optical loss and tens of mm length per MMU, significantly limiting the scalability of the design. Alternatively, one can use thermo-optic or M/NOEMS-based phase shifters with lower optical loss and lower $V_{\pi L}$. Yet, these tuning mechanisms require long delays (μs-to-ms) for reprogramming, limiting the clock speed of the photonic core to KHz to a few hundred MHz. Effectively, both options, high optical loss/longer device length or low modulation bandwidth, result in poor performance.

To resolve this issue, we modified our design to leverage data stationarity during operations by encoding the two operands (x and w) onto separate devices, which is shown in Figure 5·2(c) for a single MMU digit. Here, instead of turning the applied voltage ON or OFF via a digital AND operation, we obtain the same behavior using MRR switches to change the route of the light to go through or bypass the phase shifter to avoid frequent reprogramming in phase shifters.

MRRs are optical devices that are designed to have a resonant wavelength. If the wavelength of the signal on a waveguide that is next to an MRR matches the resonant wavelength of the MRR, the signal is coupled into the MRR, otherwise, it keeps propagating down the waveguide. By applying a voltage to an MRR, its resonant wavelength, and therefore, the route of the light on the waveguide can be changed. In Figure 5·2(c), assume the default resonant wavelength (when no voltage is applied to the MRR) and the passed optical signal's wavelength are the same. If the corresponding binary digit of the input operand is zero, i.e., $x^{(d)}$=0, no voltage is applied to the MRR so the optical signal is coupled into the MRR and routed through the bypass waveguide with no phase shifter (Figure 5·2(c), left). In contrast, if $x^{(d)}$=1, a voltage high enough to shift the resonant wavelength is applied to the MRRs so that the input signal is not affected and it propagates on the upper arm containing the phase shifter (Figure 5·2(c), right). Figure 5·2(d) shows the same multiplication unit as Figure 5·2(b) with MRR switches for the abovementioned example where $x = 101$ and $w = 011$.

In the modified design (Figure 5·2(c-d)), as x and w are encoded onto separate devices, the change in x does not impact the voltage applied to the phase shifter. Therefore, with a dataflow where w is stationary, the voltage applied to the phase shifters (wV_0) can be kept fixed during MVM operations and requires reprogramming only once for each tiled-GEMM operation, instead of every cycle. By minimizing the number of times we have to reprogram the phase shifters, we can use more efficient and low-bandwidth phase shifters with-

out compromising performance. The new design requires adding MRR switches, which introduces extra optical loss and increases the area. However, previous works show that MRRs can easily achieve tens of Gb/s with a much smaller optical loss and area footprint compared to high-bandwidth phase shifters (Ohno et al., 2021). Therefore, using a combination of MRR switches and low-bandwidth phase shifters in the MMU enables us to achieve a high-speed and scalable design.

5.2.2 Modular Dot Product Unit (MDPU) and MMVMU

Similar to cascading phase shifters in a single MMU, we can cascade multiple MMUs to construct an MDPU and perform a modular dot product. The phase shifts introduced by each MMU accumulate and the modular dot product is obtained by measuring the total phase shift ($\Delta\Phi_{\text{total}}$) on the optical signal. As the MMU operands are already scaled by $2\pi/m$, the dot product result will also be modular with m. $\Delta\Phi_{\text{total}}$ in an MDPU with g MMUs in a row is

$$\Delta\Phi_{\text{total}} = \Big| \sum_{j=0}^{g-1} \Big(\sum_{i=0}^{b-1} (2^i x_j^{(i)} w_j \frac{2\pi}{m}) \Big) \Big|_{2\pi} = \frac{2\pi}{m} \Big| \sum_{j=0}^{g-1} (x_j w_j) \Big|_m. \tag{5.4}$$

The final result of the dot product is collected at the end of the MDPU by detecting the optical phase and multiplying it by $m/2\pi$.

Multiple MDPUs construct an MMVMU and can perform a modular MVM operation in a single cycle. In Mirage, we utilize n MMVMUs to perform n modular MVMs in parallel for an RNS with n moduli. The n MMVMUs together form an RNS-MMVMU. As illustrated in Figure 5·3(a), the input residue vector x_i and weight residue tile w_i for each modulus $i \in \{1, \ldots, n\}$ are sent to the i^{th} MMVMU. x_i is broadcasted to all MDPUs within an MMVMU. As w is mapped to an analog value representing an integer smaller than m, it passes through $\lceil \log_2 m \rceil$-bit DACs without any information loss. In contrast, x is encoded digit-by-digit so the digits can directly be used to control the MRRs without DACs. The results from each MMVMU are detected via phase detection units and are passed through

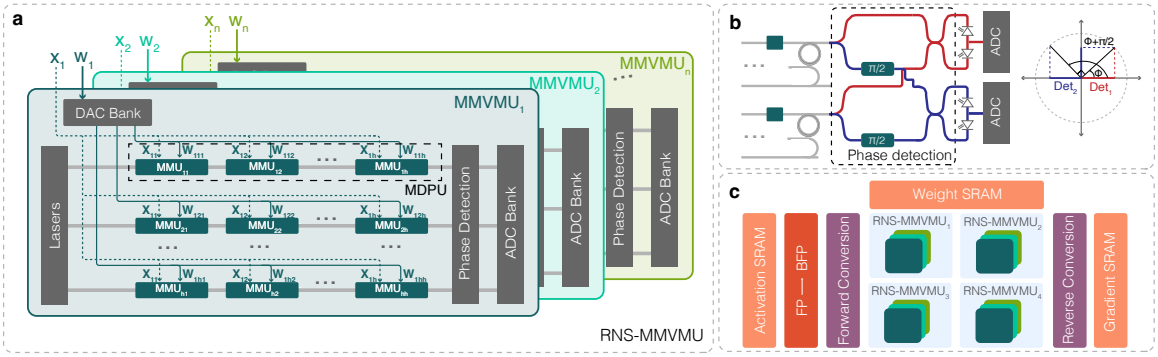

Figure 5·3: (a) RNS-MMVMU micro-architecture. (b) Phase detection unit. The top arms of the two rows detect the amplitude of the incoming signals directly while the bottom arms apply $\pi/2$ radians phase shift and detect the amplitude. Phase detection is done by using these two amplitude values. (c) Main components of Mirage architecture with four RNS-MMVMUs and three moduli as an example.

$\lceil \log_2 m \rceil$-bit ADCs. These values are then converted to BNS via the reverse conversion unit.

5.2.3 Phase Detection Unit

In Mirage, the output of a modular MVM operation is stored in the phase of the output signal from an MMVMU. However, a photodetector can only detect the amplitude of a signal. To detect the phase successfully, we need two amplitude measurements with 90 degree separation (Taylor, 2009). Figure 5·3(b) shows the detection setup in Mirage. Here, the idea is to read both x and y coordinates in the polar plane to determine the phase angle. We first directly measure the amplitude (x coordinate). Then, we apply a $\pi/2$ phase shift and measure the amplitude again (y coordinate). The combination of these two measurements is unique to the corresponding phase level. To measure the x and y components separately, we split both arms in the dual-rail setup into two. The upper splits of both arms (output of the MDPUs) are directly sent to the first set of balanced photodetectors. We apply a $\pi/2$ phase shift to the lower splits of the two arms and send them to the second set of balanced photodetectors. It should be noted that this setup requires two detections and two ADCs per MDPU and twice the laser power to be injected (compared to a single-detection setup).

5.3 Moduli Selection

Moduli selection plays a crucial role in designing Mirage. For a chosen $\mathcal{M}$, there is a limited range of values that can be uniquely represented. This range is called the dynamic range of the RNS and is $[0, M)$, where $M = \prod_i m_i$. To preserve the integrity of operations, the residues in the RNS space must stay within the RNS range, limiting the bit-width of the operands and the number of operations that can be performed in the RNS space. To this end, for a tiled-MVM operation in the RNS space, we need to ensure that

$$\log_2 M \geq b_{\text{out}} = 2(b_m+1) + \log_2(g) - 1, \tag{5.5}$$

for a BFP configuration with b_m bits of mantissa and a group size of g. Here, b_m+1 is used for b_{in} and b_w as both inputs and weights use the same BFP configuration.

In Mirage, the DNN accuracy is determined by the chosen b_m and g and is independent of the exact values of the moduli as there is no information loss during RNS operations as long as M is chosen to be large enough to guarantee we satisfy Eq. (5.5). However, the selected moduli set has a significant impact on the hardware performance. Higher b_m or g naturally dictates a larger M. A larger M requires either a higher number of moduli or larger moduli values. While the number of moduli determines the number of MMVMUs in Mirage, the bit-width of the moduli determines the bit precision of the data converters and the SNR that needs to be maintained in the photonic core.

Importantly, the selection of moduli impacts the cost of the forward and reverse conversion circuits. Typically, as M and the moduli values get larger, these conversions get slower and more energy-consuming. Several works showed that traditional conversion methods such as CRT pose performance limitations for high dynamic range when arbitrary moduli sets are used (Wang et al., 2002). In a high-speed low-energy design such as Mirage, these conversions can easily become the bottleneck. Instead, using special moduli sets and conversion hardware can alleviate the hardware overhead of these operations signifi-

cantly. In Mirage, we use a three-moduli set in the form of $\{2^k-1,\ 2^k,\ 2^k+1\}$ where k is a positive integer (Hiasat, 2019). This set reduces modulo operations into simple shift operations. During the forward conversion, $|A|_{2^k}=A>>k$, $|A|_{2^k+1}=|A|_{2^k}+1$ (subtract 2^k+1 if $\geq 2^k+1$), and $|A|_{2^k-1}=|A|_{2^k}-1$ (add 2^k-1 if <0). The reverse conversion is typically more costly than the forward conversion due to the modulo M operation with large M values. Similar to the forward conversion, the special moduli set can simplify this operation as $M=2^{3k}-2^k$ and $|R|_M=|R|_{2^{3k}}-|R|_{2^k}=(R>>3k)-(R>>k)$ (add M if <0). Previous works show that this design can provide a high dynamic range (up to 24 bits) with ~2 GHz throughput with very low power consumption (~1 mW). Please refer to Hiasat (Hiasat, 2019) for implementation details.

5.4 Mirage Accelerator Design

Figure 5·3(c) shows the main components of Mirage. Mirage consists of a photonic and an electronic chiplet that are integrated via 3D integration. When executing a DNN layer, first, the input and weight matrices are tiled and the FP-to-BFP and BNS-to-RNS (forward) conversions are performed on these tiles (steps 1-3 in Figure 5·1). These operations are handled by the electronic chiplet. The integer residues obtained by the forward conversion are then sent to the photonic chiplet (after passing through DACs if the data are mapped to analog voltages). Each input vector-weight tile residue pair for a modulus is sent to the corresponding MMVMU on the photonic chiplet. The outputs of the analog modular MVM operations are collected from the photonic chiplet through photodetectors and the TIA circuitry that are placed on the electronic chiplet. The results are converted back to the digital domain via ADCs. The RNS-to-BNS (reverse) conversion is then performed on the output residues and the values are converted back to FP from BFP on the electronic chiplet. The outputs of the tiled-GEMM operations are accumulated and nonlinearity (ReLU, MaxPool, etc.) is applied digitally in FP32. Steps 7-10 are performed via dedicated electronic

circuitry in Mirage.

The data is read/written from/to the on-chip SRAM arrays. In our design, there are three separate SRAM arrays for storing activations, weights, and gradients, that are placed on the electronic chiplet along with the other digital circuitry. In Mirage, the photonic circuit is clocked at 10 GHz, whereas the digital circuits are clocked at 1 GHz. It is crucial to match the throughput of the digital electronic components with the photonic compute unit as digital operations are much slower and can easily become the bottleneck in the accelerator. For this purpose, we use 10 copies of each digital circuit that are interleaved by 0.1 ns. Each RNS-MMVMU has its own 10 dedicated SRAM sub-arrays for each SRAM type. Every 0.1 ns cycle, an RNS-MMVMU reads and writes from/to one of these 10 interleaved SRAM sub-arrays. The same approach is used for digital conversion circuits. For each RNS-MMVMU, there exist 10 RNS-BNS converters and 10 FP-BFP converters, each triggered with 0.1 ns offset. This interleaved structure enables memory accesses and digital compute to be fast enough such that it does not limit the performance of the photonic core even though the SRAM sub-arrays and digital circuits individually have a 1 ns clock period. All operations, i.e., SRAM reads, BFP conversions, forward conversions, DAC operations, modular MVMs, photodetections, ADC operations, reverse conversions, accumulations, and SRAM writes, are pipelined to achieve a 10 Giga MVM per second throughput in each RNS-MMVMU.

5.5 Sensitivity Analysis

In this section, we analyze the impact of the number of mantissa bits (b_m) and group size (g) in the BFP representation and the accuracy-energy consumption tradeoffs introduced by these choices. After selecting the optimal b_m and g, we perform sensitivity analysis for the number of MDPUs in an MMVMU and the number of RNS-MMVMUs in Mirage. Lastly, we explore several dataflows for Mirage and the systolic array to improve utilization.

5.5.1 BFP Parameters

The BFP configuration, i.e., b_m and g choice, significantly impacts the accuracy and hardware performance of Mirage. Figure 5·4(a) shows the validation accuracy after training ResNet18 with varying b_m and g in Mirage. The results indicate that we cannot reach high accuracy when b_m=3 and the minimum b_m we can use is 4 to achieve comparable accuracy to FP32 training. However, choosing b_m=4 allows us to go up to only g=16 without a drop in accuracy. When b_m=5, we can go up to higher g values (up to 64), which enables us to perform more MAC operations in parallel. This encourages us to take a deeper look into the b_m=4 and b_m=5 cases.

Using the $\{2^k-1, 2^k, 2^k+1\}$ moduli set, the minimum k we can choose that satisfies Eq. (5.5), k_{min}=4 when b_m=3. Similarly, k_{min}=5 for b_m=4, and k_{min}=6 for b_m=5. In Figure 5·4(b), we compare the energy consumption per MAC operation of an RNS-MMVMU consisting of 3 MMVMUs (one for each modulus) for varying g and b_m. This energy consumption includes lasers, MRR tuning, DACs and ADCs, TIAs, FP-BFP and RNS-BNS conversions. While a higher g increases the number of MACs performed in parallel and helps amortize the cost of the components over g MACs, it also requires more optical elements cascaded in an optical channel and increases the optical loss. A higher optical loss requires an exponentially higher laser power in the photonic array to maintain the same SNR. As it can be seen from Figure 5·4(b), b_m=4 when g=16 provides the best energy efficiency among all the options that yield high accuracy in Figure 5·4(a). Considering these results, in Mirage, we choose b_m=4 and g=16 and use these values for the rest of the experiments.

5.5.2 Choice of the Array Size and Number of Arrays

As mentioned earlier, g controls the horizontal array size of the RNS-MMVMU, i.e., the number of MMUs in each MDPU. We can, however, increase the vertical size by increasing

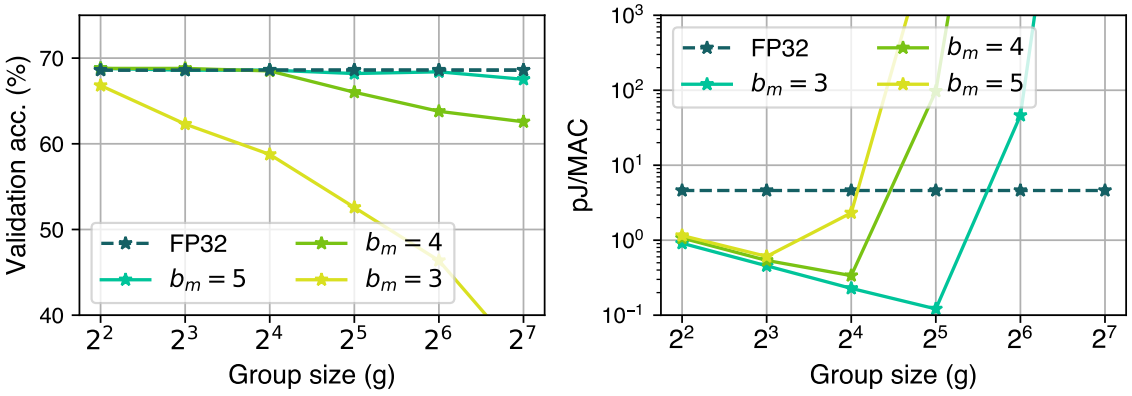

Figure 5·4: (a) ResNet18 validation accuracy on Imagenet after training from scratch for 60 epochs and (b) energy per MAC operation (pJ/MAC) for varying b_m and g. This analysis includes energy consumed by lasers and tuning circuitry, TIAs, DACs and ADCs, FP-BFP, and RNS-BNS conversions. Here, ResNet18 is shown as an example. We observed similar behavior for other evaluated DNNs.

the number of optical channels (MDPUs) for higher throughput. Additionally, we can utilize multiple RNS-MMVMUs on the same chip to further improve parallelism. Figure 5·5 (a) and (b) show the spatial utilization (how fully the MMVMUs are used) for a varying number of MDPUs in an MMVMU and a varying number of RNS-MMVMUs when g=16, respectively. The spatial utilization starts decreasing for almost all DNN models after 32 MDPUs per MMVMU. In Figure 5·5 (b), we fix the MMVMU array size to be 16×32 and increase the number of RNS-MMVMUs in Mirage. Here, we observe a decline in spatial utilization after 8 RNS-MMVMUs for most models. Considering these experiments, in Mirage, we choose MMVMU array size to be 16×32 and the number of RNS-MMVMUs to be 8.

Dataflow Choice

Dataflow choice has been shown to have a critical impact on the performance of DNN hardware accelerators (Chen et al., 2016). For DNN inference, the typical dataflows can be listed as weight stationary, input stationary, and output stationary. The performance

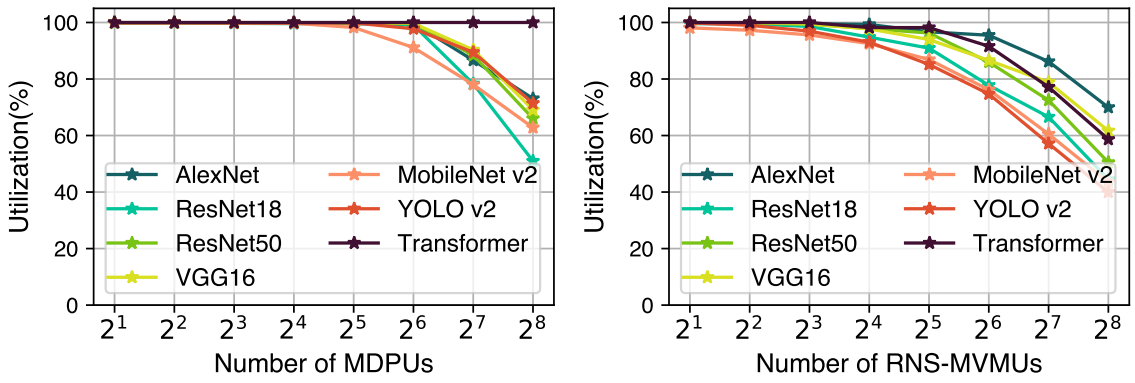

Figure 5·5: (a) Number of MDPUs versus spatial utilization (%). (b) Number of RNS-MMVM units versus spatial utilization (%).

of these dataflows depends on the DNN model (layer shapes and sizes), the chosen batch size, and the underlying hardware. In this section, we investigate the impact of different dataflow options on the performance of Mirage and traditional systolic arrays. The dataflow names are intuitive for DNN inference. However, during training, we perform three GEMM operations per layer and the operands change for each operation. In the forward pass, we perform $O{=}WX$. In the backward pass, we perform $\Delta X{=}W^T\Delta O$ and $\Delta W{=}\Delta O X^T$.

To avoid confusion, we renamed these three dataflows, weight, input, and output stationary, to DF1, DF2, and DF3, respectively. DF1 is equivalent to the weight stationary dataflow where the first operands (W for the forward pass, W^T for ΔX calculation, ΔO for ΔW calculation) in the abovementioned GEMM operations are kept stationary, DF2 is equivalent to the input stationary dataflow where the second operands (X for the forward pass, ΔO for ΔX calculation, X^T for ΔW calculation) are kept stationary, and DF3 is equivalent to the output stationary dataflow where the output of the GEMM operations (O, ΔX, and ΔW, respectively) are kept stationary.

Figure 5·6(a) shows the latency per training step (a single batch of 256) for different dataflows when running AlexNet on Mirage and a traditional systolic array with the same array size as Mirage and a clock frequency of 1 GHz. In Mirage, we only consider DF1 and

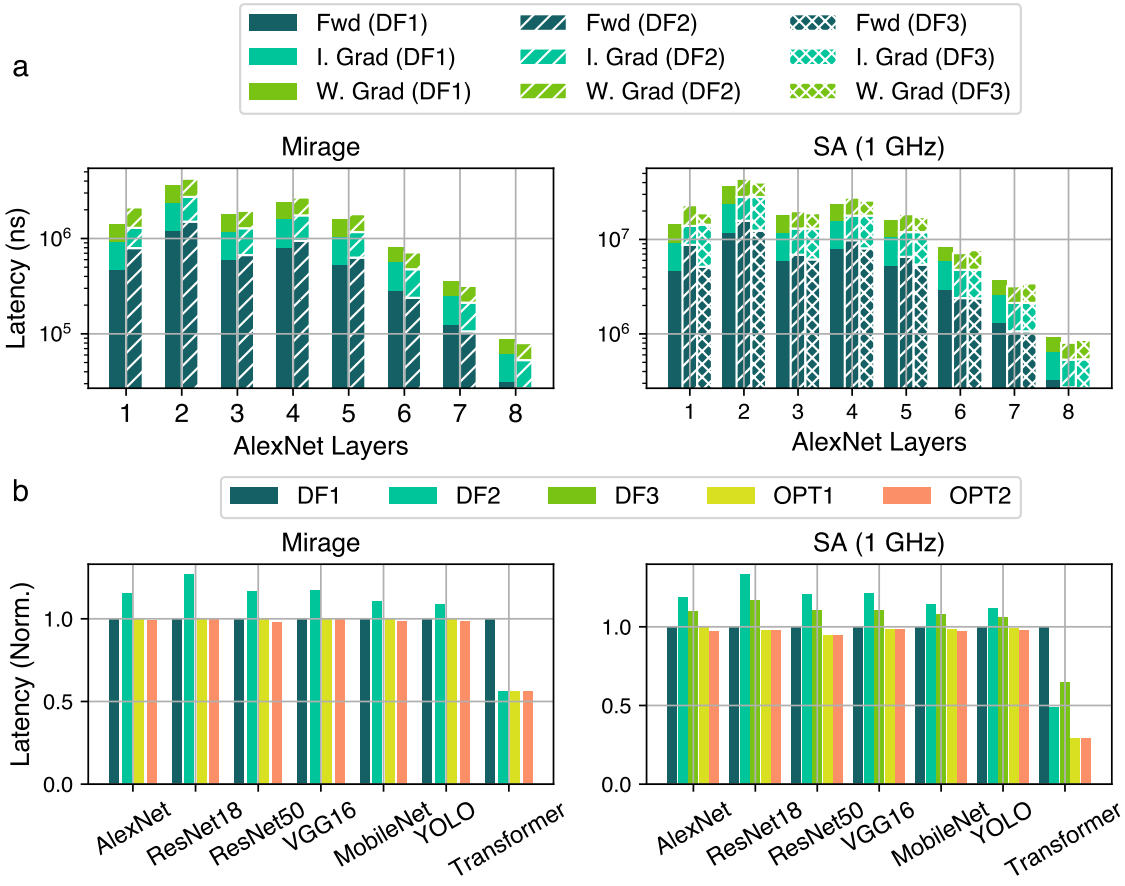

Figure 5·6: (a) Latency per step for each layer of AlexNet for Mirage (left) and a 1 GHz digital systolic array (right). (b) Latency per step for different DNNs and impact of dataflow for Mirage (left) and a 1 GHz digital systolic array (right). The numbers for all dataflows are normalized to the DF1 results for all models.

DF2 dataflows. This is because the DF3 dataflow requires both operands to be modified every cycle in the photonic arrays. However, as discussed in Section 2.2.4, the modulation bandwidth of phase shifters is a limiting factor on the operation speed in the photonic core. Therefore, it is preferable to minimize the number of updates in phase shifters to achieve high utilization of the photonic core. DF1 encodes the first operand in the phase shifters and DF2 encodes the second operand in the phase shifters. In both cases, the values encoded in the phase shifters are kept stationary–which allows for high operation frequency. All three

dataflows are applicable to systolic arrays.

In Figure 5·6(a), we observe that different dataflows perform better for different computations and different layers in a DNN model. For example, in the first layer of AlexNet, DF1 achieves a lower latency in the forward pass while DF2 achieves a lower latency in the input gradient calculation in Mirage. A similar observation can be made for the systolic array design. So to maximize performance, in both Mirage and the systolic array, we added flexibility in the choice of the dataflow. Figure 5·6(b) shows the impact of using different dataflows as well as the added data flexibility optimizations (OPT1 and OPT2) for different DNNs. OPT1 chooses the best dataflow for each type of computation (i.e., forward pass, input gradient, and weight gradient calculation) which is kept the same for all layers. A more aggressive optimization, OPT2, picks the best dataflow for each GEMM operation separately for each layer. This dataflow scheduling is done once and offline for a DNN via analytical performance estimations.

In Figure 5·6(b), for Mirage, we observe that DF1 performs better for all models except Transformer in which DF2 achieves a better performance. In all DNNs, the flexible dataflows (OPT1 and OPT2) bring minor to no benefit in Mirage. For the systolic array, however, there exists more variety in the performance of different dataflows. On average across all reported models, OPT1 boosts the performance by 11.7% and OPT2 boosts the performance by 12.5% over the best-performing dataflow. Although the OPT1 and OPT2 optimizations do not improve the performance of Mirage by much, we believe that it is important to consider this performance boost in systolic arrays to have the best possible baseline for comparison. For this purpose, we use OPT2 for both Mirage and systolic arrays for the rest of the analysis.

5.6 Accuracy Results

We evaluated Mirage's accuracy in commonly deployed CNNs for image classification on the ImageNet dataset (Deng et al., 2009), in YOLO-v2 (Redmon and Farhadi, 2017) for object detection on the PASCAL VOC2012 dataset (Everingham et al., 2010), and in a transformer (Vaswani et al., 2017) model for machine translation on the IWSLT14 German-English dataset (Bojar et al., 2014). The CNN models were trained for 60 epochs using the SGD optimizer with a batch size of 256 and a learning rate starting from 0.01 and scaled down by 10 after each 20 epoch. YOLO-v2 was trained for 120 epochs using the SGD optimizer with an initial learning rate of 10^{-4}, and scaled down by 10 after epochs 60 and 90. The 12-layer transformer model with 12 heads and a hidden size of 768 was trained for 150 epochs using the Adam optimizer with a learning rate of 10^{-4}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Table 5.1 shows the accuracy results for Mirage and several other data formats. The accuracy results for the data formats other than Mirage in Table 5.1 were obtained from the work by Zhang et al. (Zhang et al., 2022b). For Mirage, we used the exact same training parameters for fair comparison. It can be seen that Mirage can provide comparable validation accuracy to FP32 training for all benchmarks. All other reported data formats except for INT8 ($2-5\%$ accuracy degradation) achieve similar accuracy.

5.7 Performance, Power, and Area Results

Table 5.2 compares the energy consumption and area per MAC operation and clock rate for Mirage's RNS-MMVMUs against systolic arrays with various data formats. While we could synthesize the digital INT units at 1 GHz, FP units are typically more complex circuits with longer critical paths than INT MAC units forcing us to reduce the clock frequency to 500 GHz. Compared to other data formats, the main advantage of Mirage is the high clock speed of 10 GHz with comparable or less energy consumption per MAC. In addition to its speed advantage, Mirage also provides a lower energy consumption per

Table 5.1: Validation accuracy of Mirage and various data formats (Zhang et al., 2022b).

Model	Mirage	FP32	bfloat16	INT8	INT12	HFP8	FMAC
ResNet18	68.51	68.6	68.55	65.53	68.51	68.53	68.52
ResNet50	75.15	75.17	75.12	71.01	75.03	75.07	75.11
MobileNet v2	68.20	68.27	68.22	65.97	68.16	68.11	68.17
VGG16	69.5	69.74	69.71	64.5	69.33	69.62	69.78
YOLO v2	73.2	73.36	73.32	61.12	73.07	72.88	73.28
Transformer	35.4	35.41	35.39	29.18	35.27	35.38	35.4

Table 5.2: Performance, power, and area analysis of MAC units

	Mirage	FP32	bfloat16	HFP8	INT12	INT8	FMAC
pJ/MAC	0.21	12.42	3.20	1.47	0.71	0.42	0.11
mm^2/MAC	0.12	9.6E-3	3.5E-3	1.4E-3	7.7E-4	4.1E-4	N/A
$f(Hz)$	10G	500M	500M	500M	1G	1G	500M

MAC ($2-59.1\times$) compared to all data formats besides FMAC (Zhang et al., 2022b) ($\sim 2\times$ higher).

While optical MAC units can reach higher speed and energy efficiency, they typically fall short in computational density as optical devices such as phase shifters and MRRs have a much larger area footprint than digital CMOS gates. Therefore, Mirage has a larger area footprint per MAC operation and is less area-efficient than its electronic counterparts.

Figure 5·7 compares the hardware performance of Mirage against systolic arrays with MAC units using various data formats when training various DNNs. In Figure 5·7, the energy/power consumption of systolic arrays only consists of MAC units while for Mirage, we consider the energy/power consumption of lasers, photonic devices, TIAs, DACs and ADCs, RNS and BFP conversion units, and FP32 accumulators. The analyses in the figure include iso-energy (per MAC) designs (left) and iso-area designs (right). We report results for Mirage with 8 RNS-MMVMUs, each with 3 16×32 MMVMUs (See Section 5.5 for the justification of the design choices). For the iso-energy analysis, we scaled the number

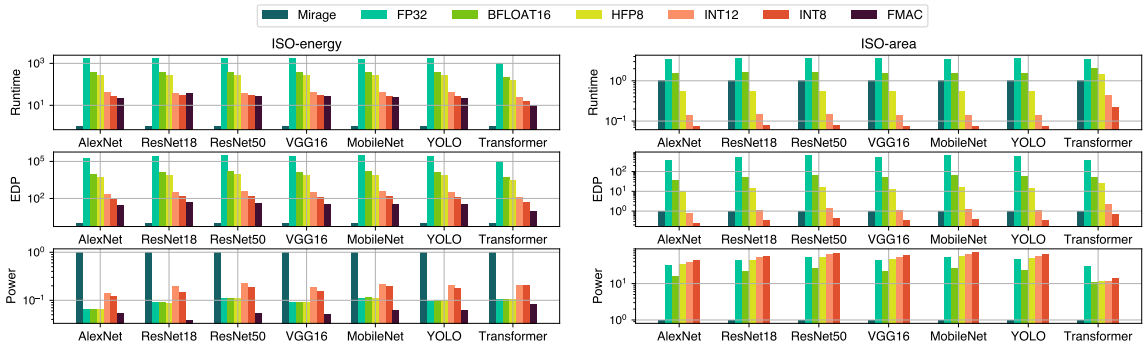

Figure 5·7: Normalized training runtime, EDP and power comparison of Mirage (eight 16×32 arrays) against systolic arrays using MAC units with various data formats. The plots on the left-hand side show the iso-energy results where the number of MAC units in the systolic arrays is scaled to consume the same energy per MAC operations using the numbers in Table 5.2. The plots on the right-hand side show iso-area results where the number of MAC units in the systolic arrays is scaled to take up the same area as Mirage. As we do not have the area footprint of the FMAC units, we do not show the FMAC numbers in the iso-area results.

of MAC units in systolic arrays to match the energy consumption per MAC operation of Mirage for different data formats using the numbers in Table 5.2. Similarly, in the iso-area analysis, we increase the number of MAC units in systolic arrays to take up the same area as Mirage. We observed that increasing size leads to long latencies to load up the new tile and causes the systolic array performance to go down significantly. To avoid this performance drop in systolic arrays, while increasing the number of MAC units, we kept the 16×32 array size fixed and used multiple systolic arrays instead. While INT8 cannot meet the high accuracy criteria, it is shown for completeness (See Section 5.6).

In the iso-energy analysis, the best-performing data format among the systolic array designs is FMAC (Zhang et al., 2022b). Given the same energy per MAC budget, on average across the reported DNNs, Mirage achieves a 23.8× lesser runtime and 32.1× lower EDP than the systolic array with FMAC units. However, in this case, Mirage consumes 17.2× higher power consumption. Compared to the systolic array with FP32 MAC units, on average, Mirage provides 3.5× lesser runtime and 521.7× lower EDP while consuming

42.8× less power.

The iso-area results show that the most efficient datatype that achieves high accuracy, INT12, achieves 5.4× better runtime than Mirage on average due to the large area footprint of Mirage. However, while being slower in the iso-area scenario, Mirage has 42.8× lower power consumption and 1.27× lower EDP compared to INT12. Mirage has 3.5× lesser runtime, 521.7× lower EDP and 42.8× lower power consumption compared to FP32 for the iso-area scenario.

Overall, the results indicate that there exists a tradeoff between runtime, area, and power consumption. Compared to digital systolic arrays, given the same energy budget, Mirage can perform faster DNN training, however comes with a higher power consumption and area footprint. In contrast, given the same area budget, Mirage has lower power consumption with comparable or better EDP.

Figure 5·8 shows the peak power and area breakdown for Mirage. It can be seen that SRAM accesses consume most of the power (61.2%) in Mirage. This is mainly because we store all data in FP32 and perform frequent SRAM operations. To reduce this cost, more efficient data formats (FP16, BFP, etc.) can be chosen to store data and perform nonlinearities—which would reduce the total data storage requirements and the energy consumption per SRAM access.

Figure 5·8 (right) shows that most of the area is occupied by photonic devices and SRAM. All the components take up 476.6 mm^2 in total, 234 mm^2 for the photonic and 242.7 mm^2 for the electronic chiplet. As the photonic and electronic chiplets are stacked via 3D integration, the total area can be considered as the largest of the two chiplets (242.7 mm^2).

Compared to NVIDIA GPUs, Mirage can achieve ∼18.5× better EDP per epoch when training ResNet-50 according to MLPerf Training v4.0 results (Reddi et al., 2020). It should be noted that the GPU system considered consists of eight NVIDIA H100 GPUs

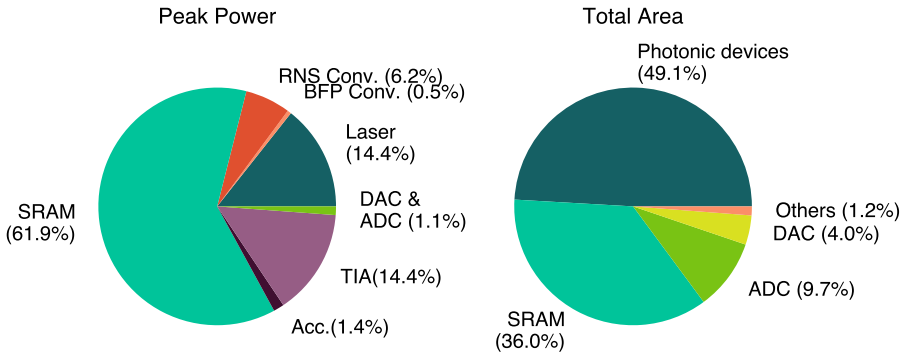

Figure 5·8: Peak power consumption and area breakdown for Mirage. The total peak power consumption is 19.95 W and the total area is 476.6mm^2.

and two host CPUs. Our results do not include the energy and latency overhead of the host CPUs and the corresponding operations (e.g., weight updates), so the actual performance gain is expected to be slightly less than reported. In addition, there are differences between the MLPerf implementation and our training script, including the optimizer type and parameters and the number of training epochs. Therefore, we compare the EDP of the GPU system and Mirage on a per-epoch basis. Despite these differences in the two results, as the accelerator's latency and energy consumption dominate the overall training process, we anticipate that Mirage will outperform the GPU system by more than an order of magnitude in EDP.

5.8 Evaluation Methodology

5.8.1 Accuracy Modeling

We modeled the accuracy of Mirage in PyTorch using customized GEMM layers. In all models, we swapped each GEMM operation, i.e., convolution and linear layers, with our customized BFP versions for a given b_m and g. Once the values are converted to the BFP format, BNS-RNS and RNS-BNS conversions and the chosen moduli set have no direct impact on the DNN accuracy as long as the RNS operations are guaranteed to stay within the RNS range. So we omit these conversions in the accuracy model for faster training

experiments.

In our customized GEMM layers, the tensors are first flattened and grouped. For each BFP group, we calculate the shared exponent and align the mantissae for a given b_m and g. We then perform the convolution or linear operation and collect the result. This BFP conversion is done for all GEMM layers during both forward and backward passes of each layer. While the gradients are calculated in BFP, we make the weight updates in FP32. For this purpose, we store the weights in FP32 instead of BFP and call them within the optimizer right before the parameter update step. After updating the weights, we switch back to the BFP format before the next forward pass.

5.8.2 Hardware Performance, Power and Area

Photonic Devices and Lasers

The latency of the photonic RNS-MMVMUs in Mirage for GEMM operations is calculated through an in-house simulator. This simulator calculates the number of tiles and the number of operations per tile within a DNN layer given the hardware configuration and layer shapes. For each tile, the reprogramming of phase shifters (similar design to Baghdadi et al. (Baghdadi et al., 2021), internally simulated) takes 5 ns during which the photonic compute core is inoperable. Once the values in the phase shifters are settled, one RNS-MMVM operation is completed every 0.1 ns (10 Giga MVMs per second). This operation rate is based on the modulation bandwidth of the MRRs (Ohno et al., 2021). ADCs (Xu et al., 2016) achieve $\geq$10 GS/s sampling rate so they do not cause a latency overhead when the operations are pipelined.

The photonic core power consumption includes the laser source power and MRR tuning power. The laser power injected into the MMVMUs needs to ensure that a target SNR, which is dependent on the modulus value, is achieved. For a modulus m, we should be able to differentiate m phase levels ($\log_2 m$ bits), i.e., SNR$>m$ where the noise includes shot and thermal noise mentioned in Section 2.2.5. From the photodetector, we back-calculate the

required laser power that can maintain an adequate SNR accounting for all the optical losses on the optical path. The phase shifters have a $V_{\pi L}$ = 0.002 V·cm modulation efficiency and 1.6 dB/mm loss.

The tuning cost of the phase shifters is negligible (a few fJ/bit). Each MRR has a radius of 10 μm and a total (insertion and propagation) loss of 0.2 dB when coupled with the light (Ohno et al., 2021). MRRs use electro-optical tuning and have a very small power consumption of 0.3 pW for switching (Ohno et al., 2021). This power dissipation is $\sim 10^7\times$ smaller than thermo-optic shifters which resolves the thermal crosstalk problem in MRRs (Ohno et al., 2021). Each 180 degree bend waveguide has a 5 μm radius and 0.01 dB insertion loss (Bahadori et al., 2019). The laser-to-chip coupler has a 0.2 dB loss (Hu et al., 2023) and the laser has a 20% efficiency (Mourou et al., 2013). The length of the phase shifters varies depending on the modulus value in the selected set $\{2^k-1, 2^k, 2^k+1\}$ where $k = 5$ (choice of k will be justified in Section 5.5). Using the Eq (5.3) and device metrics ($V_{\pi L}$ = 0.002 V·cm and V_{bias}=1.08V), the total phase shifter length for the largest moduli 33 can be calculated as 0.57 mm. With the MRRs included, the total horizontal length of a single MMU becomes 0.8 mm.

Digital Circuitry

The output signal of the photonic core is converted to the electrical domain through photodetectors and TIAs. The photodetectors have a 1.1 A/W responsivity. The TIAs consume 75 fJ/bit (Rakowski et al., 2018). Each 6-bit DAC with 20 GS/s sample rate consumes 136 mW power and takes up 0.072 mm^2 area (Kim et al., 2018). Each 6-bit ADC with 24 GS/s sample rate consumes 23 mW power and takes up 0.03 mm^2 area (Xu et al., 2016). As DACs in Mirage are used only once for each tile and ADCs perform conversions every 0.1 ns (10<24 GS/s), the power consumption of DACs and ADCs is amortized over the total training time. The bit precision required for DACs and ADCs is determined by the moduli set $\{2^k-1, 2^k, 2^k+1\}$ where $k = 5$ (choice of k will be justified in Section 5.5). For

m=2^k+1 with k=5, $\lceil \log_2 m \rceil$=6 so we use the 6-bit DACs and ADCs as is. For 2^k and 2^k-1, $\lceil \log_2 m \rceil$=5 so we scale the energy consumption down by 1 bit (Murmann, 2021). All three SRAM arrays (activation, weights, and gradients) are generated using the SRAM compiler for TSMC 40 nm technology node (TSMC, 2024). In Mirage, we employ three SRAM arrays, each with 8 MB size, consisting of 32 kB memory banks with an access latency of $\leq$1 ns. The BFP-FP and BNS-RNS conversion circuits are implemented in RTL and synthesized using Cadence Genus (Cadence, 2024) and the TSMC 40 nm library with a clock rate of 1 GHz. Each BFP-FP unit consumes 1.32 pJ and has a 1318.4μm^2 area footprint. Each BNS-RNS unit consumes 0.17 pJ with a 231.7μm^2 area footprint Each RNS-BNS unit consumes 0.48 pJ energy per conversion and requires 1545.8μm^2 area (Hiasat, 2019).

For comparison, we use systolic arrays that support several data formats including FP32 (Hauser, 2019), BFLOAT16 (Wang and Kanwar, 2019), HPF8 (Sun et al., 2019), INT8, INT12, and FMAC (Zhang et al., 2022b). We chose systolic arrays for their common usage in DNN acceleration and their superior performance over CPUs and GPUs (Chen et al., 2016; Jouppi et al., 2020). We implemented MAC units with the abovementioned data formats in RTL except FMAC for which the energy number is obtained from the recent work by Zhang et al. (Zhang et al., 2022b). The power and area per MAC unit are collected through synthesis using Cadence Genus and the TSMC 40 nm library.

5.9 Related Work

Besides the related work mentioned in Section 4.6, it is noteworthy to mention RRAM-based PiM designs targeting similar precision limitations using multiple low-bit cells to compose higher-bit results. For example, PRIME (Chi et al., 2016) uses two 3-bit cells to achieve 6-bit precision. Another example, PipeLayer (Song et al., 2017), uses four 4-bit cells to achieve 16-bit precision through shift-and-add operations for DNN inference and

training. While this approach is similar to using RNS in terms of composing high-precision from low-precision arithmetic, each b-bit MAC still produces $\geq 2b$-bit result. In RNS, bit precision does not grow during operations. Compared to PipeLayer, Mirage is 14.4× more power-efficient (OPs/s/W) while being 8.8× less area efficient (OPs/s/mm^2).

In addition, over the years, several photonic DNN inference accelerators have been proposed. While Mirage is designed for DNN training, it can still be used for DNN inference. For completeness, we compare Mirage's performance while running DNN inference against existing photonic and electronic DNN inference accelerators. This comparison is shown in Table 5.3. Mirage achieves a better IPS than all accelerators (by $1.12-8.4\times$ compared to photonic and by $176-1,856\times$ compared to electronic accelerators) except for ADEPT (3.37× slower) and TPU v3 (3.12× slower). Mirage provides a better power efficiency (IPS/W) than all photonic (by $2.1-15.4\times$) and electronic (by $1.74-84.7\times$) accelerators except for ADEPT (2.48× lower). Mirage is more area-efficient (IPS/mm^2) than all photonic (by $1.03-4.36\times$) and electronic (by $2.38-93.9\times$) accelerators except for ADEPT and HolyLight (1.16× and 8.32× lower, respectively). It should be noted that these photonic inference accelerators in Table 5.3 provide a much lower dynamic range than Mirage (∼8 bit vs. ∼15 bit in Mirage). Even for DNN inference, these works can typically achieve high accuracy only through quantization-aware training. With the same methods applied, in Mirage, a lower b_m and a moduli set with a much smaller M can be utilized, resulting in significantly better hardware performance.

5.10 Discussion

Over the years, researchers have developed many ADC designs including traditional ΔΣ ADCs, Flash ADCs, Successive Approximation Register (SAR) ADCs, and hybrid ADCs. In recent years, the architecture trends have shifted mainly towards hybrid and SAR-based architectures combined with advanced techniques such as pipelining and time-interleaving

Table 5.3: Mirage versus DNN inference accelerators.

Accelerator	ResNet50			AlexNet		
	IPS	IPS/W	IPS/mm^2	IPS	IPS/W	IPS/mm^2
Mirage	10,474	1,540.6	43.2	64,963	1,904.5	267.67
ADEPT	35,698	1,587.99	50.57	217, 201	7,476.78	307.64
Albireo-C (Shiflett et al., 2021)	N/A	N/A	N/A	7,692	344.17	61.46
DNNARA (Peng et al., 2020)	9,345	100	42.05	N/A	N/A	N/A
HolyLight (Liu et al., 2019)	N/A	N/A	N/A	50,000	900	2,226.11
Eyeriss (Chen et al., 2016)	N/A	N/A	N/A	35	124.80	2.85
Eyeriss v2 (Chen et al., 2019)	N/A	N/A	N/A	102	174.80	N/A
TPU v3 (Jouppi et al., 2020)	32,716	18.18	18.00	N/A	N/A	N/A
UNPU (Lee et al., 2018)	N/A	N/A	N/A	346	1,097.50	21.62
Res-DNN (Samimi et al., 2020)	N/A	N/A	N/A	386.11	427.78	N/A

to push the envelope further. Typically, for low-speed ADCs ($f_s \leq 10^8$ samples/sec), a widely accepted limit is the minimum possible energy spent on a class-B switch capacitor circuit, i.e., $E_{\text{ADC}} \geq 8kT \times \text{SNR}$, whereas most high-speed ADCs are technology-limited (Murmann, 2022). This indicates that there can be some room for improvement in high-speed data converters with further technology scaling. However, with technology scaling significantly slowing down, specialization and new designs can only provide limited opportunities to improve energy efficiency. Therefore, we believe that our work holds an important role in terms of enabling energy-efficient next-generation analog hardware.

It is noteworthy that in our design, data converters consume only 1.1% of the overall power consumption–which is contrary to a typical analog accelerator where data converter power consumption is a dominating component. This is mainly because the reduced bit-precision of DACs/ADCs results in an exponential decrease in their power consumption. While the decreasing bit-precision also reduces the required SNR during analog operations, the increased phase shifter length and optical loss prevent the laser power from going down exponentially. In addition, the power consumption of other components (SRAM arrays, TIAs, accumulators, etc.) increases due to the increasing component count with the use of

multiple moduli. This results in a significant reduction in the relative contribution of data converter power.

Overall, the results indicate that there exists a tradeoff between runtime, area, and power consumption. Compared to digital systolic arrays, given the same energy budget, Mirage can perform faster DNN training, however comes with a higher power consumption and area footprint. In contrast, given the same area budget, Mirage has lower power consumption with comparable or better EDP.

5.11 Chapter Summary

In this chapter, we introduced Mirage, an electro-photonic DNN training accelerator that overcomes these precision challenges using the RNS. The RNS-based dataflow introduced in Chapter 4 allows for high-precision operations through multiple low-precision modular operations. In Mirage, we employed novel photonic modular arithmetic units based on cascaded phase shifters to support this dataflow and perform modular operations in the analog domain. By combining RNS with photonics, we showed that Mirage can achieve high energy efficiency without sacrificing precision, enabling the training of state-of-the-art DNNs with accuracy comparable to FP32 training. Our evaluation showed that, on average, Mirage can achieve over 23.8× faster training and 32.1× lower EDP compared to SAs in an iso-energy scenario, and consumes 42.8× less power with comparable or better EDP in an iso-area scenario.

Chapter 6

Summary and Future Work

In this chapter, we summarize the findings and major contributions of this thesis, the limitations of the completed work, and future research directions.

6.1 Summary of Contributions

In this thesis, we pointed out the lack of system-level analysis and a clear roadmap for designing hybrid electronic-photonic accelerators capable of training and deploying state-of-the-art DNNs despite the promise of the prior works. Furthermore, we focused on the precision limitation in photonic hardware, which has often been overlooked in earlier studies. We underlined that today's DNNs require higher precision to maintain accuracy, even during inference, making training a distant goal in photonics. To this end, we explored the opportunities and challenges of photonic computing for DNN acceleration with a pragmatic approach, offering solutions to these challenges.

In Chapter 3, we detailed the design and evaluation of a full electronic-photonic accelerator for DNN inference, ADEPT. ADEPT leveraged a photonic computing unit for GEMM operations, a vectorized digital electronic ASIC for non-GEMM operations, and SRAM arrays for storing DNN parameters and activations. Unlike previous works on photonic DNN accelerators, we adopted a system-level perspective, demonstrating that while the gains were substantial, they needed to be tempered relative to prior expectations. The aim was to encourage architects to explore photonics technology pragmatically, considering the system as a whole to understand its applicability in accelerating modern DNNs. We

discussed design steps and optimizations to minimize the overhead of electronic devices. Our evaluation showed that this complete accelerator system provided, on average, 5.73× higher throughput per watt compared to traditional SAs in a full system, and at least 6.8× and 2.5× better throughput per watt compared to state-of-the-art electronic and photonic accelerators, respectively.

Chapter 4 tackled the precision challenge in analog computing by using RNS to perform high-precision operations from multiple low-precision operations, thus avoiding the need for high-precision data converters and preventing information loss. The study showed that the RNS-based approach could achieve $\geq 99\%$ of FP32 accuracy in state-of-the-art DNN inference using only 6-bit arithmetic and training with 7-bit FXP arithmetic. These findings suggested that employing RNS could drastically reduce the energy consumption of analog accelerators for the same precision. In addition, we incorporated RRNS to improve the fault tolerance of the dataflow against the noise and errors inherent in analog hardware.

Chapter 5 introduced Mirage, a photonic DNN training accelerator based on the RNS framework discussed in Chapter 4. Mirage employed a novel micro-architecture to support RNS-based dataflow and modular arithmetic in the analog domain. By combining RNS and photonics, we showed that Mirage could achieve high energy efficiency without compromising precision and successfully train state-of-the-art DNNs with accuracy comparable to FP32 training. The study showed that, on average, across several DNNs, Mirage achieved more than 23.8× faster training and 32.1× lower EDP in an iso-energy scenario and consumed 42.8× lower power with comparable or better EDP in an iso-area scenario.

Overall, this thesis demonstrated that while performance expectations for photonic accelerators should be moderated compared to prior works evaluating photonic cores in isolation, photonics technology remains a promising candidate for next-generation AI hardware.

6.2 Current Limitations and Future Research Directions

In this section, we discuss the limitations of current electro-photonic DNN accelerator systems and the research conducted as part of this dissertation. We highlight several key challenges in the field and propose future research directions to address these issues.

6.2.1 Device-Level Challenges

The characteristics of optical components directly influence the performance and scalability of the photonic compute circuits. Improving the quality, efficiency, and density of photonic devices is an active research field. As mentioned in Section 2.2.4, there exist tradeoffs between different device metrics, making it highly challenging to achieve low optical loss, high modulation bandwidth, short device lengths with CMOS-level voltages, and low reprogramming costs in the same photonic device.

Heterogeneous Material Integration

Expanding beyond the conventional silicon platform, recent advances in emerging materials and novel devices leveraging electro-optic effects such as Pockels effect, Kerr effect, and quantum-confined stark effect present new avenues for improving the performance of PICs (Ning et al., 2024). For example, heterogeneous modulators utilizing III-V materials (Han et al., 2017), 2D materials (Sorianello et al., 2018), organic materials (Kieninger et al., 2018), and ferroelectric materials (Petraru et al., 2002) show significant enhancements in various device metrics. These devices can achieve modulation bandwidths of GHz or more, along with high modulation efficiency and low power consumption. However, integrating these technologies is complex and not without challenges. The integration and packaging issues and unique challenges of each material offer future research opportunities for developing more compact and energy-efficient photonic devices.

Operating at Visible Wavelengths

Device characteristics can also be enhanced by adjusting the operating wavelength (Shiflett, 2022). Most silicon photonic devices use the O-band (1260-1360 nm) and C-band (1530-1565 nm). Shifting to visible or ultraviolet wavelengths offers a potential approach for achieving smaller device sizes (Liang et al., 2021). However, this shift presents new challenges as silicon becomes opaque at these shorter wavelengths. As a result, current visible-wavelength PICs are often fabricated using silicon nitride (SiN). Although SiN is transparent at visible wavelengths, SiN waveguides suffer from higher propagation losses at visible wavelengths compared to silicon waveguides at infrared wavelengths (Sacher et al., 2019). Additionally, current SiN modulators primarily rely on the thermo-optic effect in active devices (Yong et al., 2022; Sacher et al., 2019) and cannot support carrier-depletion modulation (Shiflett, 2022), significantly limiting their modulation bandwidth. Nevertheless, recent advancements have demonstrated the potential for high-speed SiN modulators utilizing electro-optic modulation based on the Pockels effect (Alexander et al., 2018), showing promise for next-generation PICs operating in the visible spectrum.

6.2.2 Routing and Integration Challenges

Photonic Routing

Electrical circuits are commonly connected using Manhattan directions, with metal wires making 90 degree turns. However, waveguides in photonic circuits cannot accommodate such abrupt bends. Even with high-contrast silicon strip waveguides, bends must have radii of several micrometers, complicating the routing of photonic circuits. These larger bend radii make Manhattan-style routing impractical, necessitating smoothly curved connections at various angles for photonic circuits. This requirement for larger bends and the flexibility of all-angle connections significantly complicates the routing constraints for multiple waveguides.

Moreover, while CMOS circuits have long been manufactured with multiple metal layers, photonic circuits have traditionally been limited to planar designs. SiN has been proposed as a CMOS-compatible material for creating multilayer 3D PICs (Sacher et al., 2018). This approach could substantially increase integration density and reduce the number of lossy waveguide crossings in planar circuits with future advancements in SiN PICs.

Integration with Electronics

The integration of PICs with electronic ICs is another promising area of exploration. Silicon photonics enables the monolithic integration of photonic devices with CMOS electronics, showing significant potential in recent studies (Stojanović et al., 2018; Giewont et al., 2019). Moreover, advanced packaging techniques such as 3D (Ramey, 2020) and 2.5D integration can be advantageous as they allow for the use of different technology nodes for the individual electronic and photonic ICs that provide the best performance.

On-chip Laser Integration

While off-chip light sources offer better temperature stability and high light-emitting efficiency, they typically suffer from significant coupling losses between the off-chip light source and the silicon chip, as well as high packaging costs. In contrast, an on-chip light source can achieve higher integration density, compact size, and superior energy efficiency (Zhou et al., 2015). However, silicon's low emission efficiency (Liang and Bowers, 2010) encouraged the use of off-chip light sources and delayed the development of on-chip lasers.

To address this, various promising alternatives have been explored such as erbium-related light sources (Kenyon, 2005; Yerci et al., 2010), Germanium-on-Silicon (Ge-on-Si) lasers (Liu et al., 2010), and III-V-based lasers (Fang et al., 2006; Sun et al., 2009). Electrically pumped Erbium-related lasers have not yet been demonstrated due to their low power efficiency and problems such as energy back-transfer and device instability (Zhou et al.,

2015). Ge-on-Si lasers offer potential for optoelectronic integration but face challenges like high threshold currents (Sukhdeo et al., 2013) and redshift effects (Zhou et al., 2015). III-V-based hybrid silicon lasers, produced via bonding techniques, show the best performance but encounter heat dissipation issues and limitations for mass production (Van Campenhout et al., 2007). Alternatively, III-V quantum-dot lasers grown directly on silicon (Tanabe et al., 2012) appear promising for future low-cost, high-yield, temperature-insensitive, large-scale monolithic integration.

6.2.3 Data Conversion Challenges

Many prior works, including this thesis, have demonstrated that optical MAC operations are significantly faster and more energy-efficient compared to their electrical counterparts. However, the necessary conversions between digital and analog, as well as between electrical and optical domains, dominate the energy consumption in electro-photonic systems, posing a major constraint on their overall energy efficiency.

To mitigate the number of conversions, current systems aim to maximize data reuse through techniques such as optical broadcasting (Tait et al., 2017; Zhu et al., 2024) for spatial sharing of encoded signals and time-integration (Zhu et al., 2024; Li et al., 2023) for temporal accumulation in the analog domain (Ning et al., 2024). In addition to minimizing the number of conversions, lowering bit precision is crucial for reducing the energy consumption of these conversions. The RNS-based approach proposed in this thesis seeks to minimize the bit precision of data converters without compromising accuracy, thereby alleviating this energy bottleneck.

Another promising solution is to combine the digital-to-analog and electrical-to-optical steps (or optical-to-electrical and analog-to-digital steps) using photonic data converters. These converters can offer high sampling rates, precision, and low distortion while being less susceptible to jitter and EM noise (Shastri et al., 2021). Photonic compute circuits can greatly benefit from this approach, as photonic data converters provide a reduced foot-

print, high sampling rates, and low power consumption. Previous works have demonstrated 2-bit photonic DAC structures based on optical intensity weighting of multiwavelength signals, modulated with MRRs (Sun et al., 2018) and MZMs (Patel et al., 2015). However, achieving high-speed operation with high ER in these designs remains a significant challenge (Shastri et al., 2021). A potential solution for scaling to higher bit numbers is the use of coherent parallel photonic DACs (Meng et al., 2021; Shastri et al., 2021).

6.2.4 Scaling Challenges

In Chapter 3, we highlighted the necessity of large SRAMs in photonic DNN acceleration, but the SRAM sizes are constrained by several factors. Large SRAMs suffer from high leakage power and low yield, making them less efficient. Additionally, as models grow increasingly large and reticle size stays limited, scaling out to multiple chiplets becomes inevitable.

Scaling up to multiple photonic chiplets creates new challenges as these chiplets require significantly higher bandwidth compared to their electronic counterparts. To address these challenges, exploring advanced integration techniques such as wafer-scale approaches, 2.5D, and 3D integration becomes critical. Innovations in interconnect technologies (e.g., silicon photonics for high-bandwidth, low-latency communication (Harris et al., 2022)) and advanced packaging techniques to enhance thermal management will be essential.

Moreover, developing design automation tools for multi-chiplet electronic-photonic systems can help optimize the layout and connectivity of chiplets, ensuring maximum efficiency and performance.

6.2.5 Performance Modeling Challenges

The lack of robust simulation tools for electro-photonic systems poses a significant challenge, primarily due to the complexity of integrating multiple components that must function together. It is essential to develop comprehensive simulator frameworks that can ef-

fectively evaluate the performance of such computing systems. An ideal simulator should support the integration of new optical hardware, provide automatic algorithms for hardware mapping, and offer detailed evaluations of chip-level performance (Ning et al., 2024). These tools are vital for ensuring fair and straightforward comparisons across various optical circuit designs, helping to identify system bottlenecks and guide further optimizations.

To integrate optical computing hardware into mainstream AI software stacks, it is also necessary to create specialized compilers and instruction sets tailored for photonic computing architectures (Ferreira De Lima et al., 2020). Existing deep learning frameworks like PyTorch and TensorFlow can still be used as front-end interfaces, while the compiler must be adapted to generate machine code optimized for accelerators, ensuring effective use of photonic hardware (Ning et al., 2024).

6.2.6 Challenges around RNS-Based DNN Computation

Forward and Reverse Conversion

RNS-based DNN computation presents several challenges that must be addressed to optimize its effectiveness. One significant issue is the expense associated with forward and reverse conversions between BNS and RNS performed before and after every matrix operation. Conversion methods relying on special moduli are proven to be the most efficient among various methods. For example, the cost of these conversion circuits was 6.2% of the overall power consumption in Mirage as reported in Section 5.7. However, the special moduli set approach restricts the range of moduli that can be used, often leading to less optimal choices from an accelerator performance perspective. Alternatively, when methods supporting arbitrary moduli are used such as CRT, this energy and latency overhead significantly increases and becomes the dominating factor. Therefore, developing cost-effective conversion techniques for arbitrary moduli or reducing the number of conversions could significantly enhance accelerator performance.

To reduce the number of forward and reverse conversions, prior works proposed com-

puting DNN inference fully in the RNS space (Samimi et al., 2020; Salamat et al., 2018). As mentioned in Section 4.6, these methods use approximations for nonlinear operations and periodically need ranging operations to avoid overflow. This fully-RNS approach limits us to use all-integer operations, however, it requires forward and reverse conversion only once before and after each pass, eliminating the need for using special moduli sets. Scaling operations are still possible in RNS, yet, the cost of extra operations and the impact of the lack of FP numbers on inference and training accuracy should be investigated.

Extending the RNS Approach

The RNS-based approach can reduce data converter energy and provide a sufficient dynamic range for DNN inference and training, especially since full FP32 precision is typically unnecessary, However, it encounters limitations when higher precision is required. In such cases, the bit-width of the moduli must be increased, causing the RNS-based approach to suffer from inefficiencies similar to those of traditional analog methods. Addressing these challenges is crucial for improving the performance and scalability of RNS-based DNN computations for high-precision computing.

The precision limitation can be eliminated by combining the RNS framework with a positional number system (PNS), allowing it to work freely with arbitrary precision. One can represent an integer value as D separate digits where each digit is represented as a set of residues in the RNS domain and has an RNS range of M. This hybrid scheme can achieve $D \log_2 M$ bit precision where D can be liberally increased without increasing the bit precision of the data converters. Different from the RNS-only scheme, the hybrid scheme requires overflow detection and carry propagation from lower digits to higher digits. More details are provided in Appendix A.2.

Besides the PNS approach, some works explored hierarchical RNS (Djath et al., 2019; Yassine, 1992; Hollmann et al., 2018) methodologies to break down the large moduli of an RNS into smaller moduli within residue number subsystems. Hierarchical RNS involves

multiple levels where each level has a distinct moduli set and the residues of a level are represented in a residue number subsystem. This approach allows hierarchical RNS to achieve large dynamic ranges using only a few distinct, small moduli. This approach, however, requires performing multiple forward and reverse conversions for each level to convert numbers between BNS and RNS.

6.2.7 Future of Photonic Computing

Our experiments in Chapter 3 reveal that current electro-photonic systems for DNN inference offer less than an order of magnitude improvement over purely electronic systems. While a $\sim 5\times$ performance gain might seem modest, it is important to recognize that this is only the initial potential of what photonics can accomplish with today's technology. The current limitations are primarily due to the reliance on electrical components, which dominate power consumption.

To truly unlock the potential of photonics, future developments must focus on performing more operations in the optical domain, thereby reducing the need for frequent data conversions and minimizing memory access bottlenecks. Although this increases the optical loss and, consequently, laser power requirements, as mentioned in Section 6.2.1, advancements in photonic devices are expected to mitigate these issues. Emerging optical components with lower optical losses can enable the development of more scalable and efficient photonic cores, significantly reducing the impact of electrical overhead.

Moreover, replacing nonlinear operations with optical alternatives can allow systems to remain within the optical domain longer, further enhancing performance. In addition, exploring alternative memory structures, such as PCM-based memory that can operate via optical signals, offers the potential for performing memory operations in the optical domain. While these technologies still face challenges and may not yet match the density of traditional memory arrays like SRAM or DRAM, their integration into electro-photonic systems can lead to a significant step toward achieving superior performance. As we scale

towards larger systems, the power of photonic compute units can be enhanced with data movement and communication via photonics between these units. This can amplify the efficiency and performance of photonic systems, enabling them to fully realize their advantages over traditional electronic approaches.

Finally, algorithmic innovations can play a vital role in enhancing the energy efficiency of photonic systems. For instance, the RNS-based approach proposed in this thesis demonstrates how reducing bit-precision through algorithmic novelty can lead to significant performance gains, pushing photonic computing beyond the results observed in Chapter 3. These advancements suggest that, as photonics technology evolves, the performance benefits can far exceed the initial improvements, making photonics a strong candidate for replacing the CMOS technology for future computing paradigms.

6.3 Final Remarks

In conclusion, this thesis has delved into the intricate challenges and promising opportunities within photonic computing for deep learning inference and training. Through the design and evaluation of ADEPT, we have underscored the need for a more realistic approach to evaluating photonic computing. This work offers practical insights for designing electro-photonic systems and highlights the significant potential of these systems for next-generation DNN inference hardware. Furthermore, our innovative application of the RNS has effectively addressed precision challenges, a critical limitation not only in photonics but across various analog technologies. This RNS-based methodology has brought the once far-fetched goal of using photonic computing for state-of-the-art DNN training tasks within reach. This work opens the path for the development of next-generation analog systems for AI and other computationally intensive, high-precision tasks, such as cryptography and scientific computing. Overall, we believe that our contributions offer valuable insights and inspiration for researchers and engineers seeking to harness the power of pho-

tonics technology in shaping the future of computing, advancing the rapidly evolving field of AI, and pioneering other next-generation computing paradigms.

Appendix A

Appendix

A.1 Error distribution in the RRNS code space

For an RRNS$(n+k,n)$ with n non-redundant moduli, i.e., $\{(m_1,m_2,...,m_n\}$ and k redundant moduli, i.e., $\{m_{n+1},m_{n+2},...,m_{n+k}\}$, the probability distributions, i.e., p_c, p_d, and p_u, of different types of errors, i.e., Case 1, Case 2, and Case 3 that were mentioned in the RRNS for Fault Tolerance subsection are related to the Hamming distance distribution of the RRNS code space. In an RRNS$(n+k,n)$, every integer is represented as $n+k$ residues (r_i where $i \in \{1,...,n+k\}$) and this vector of $n+k$ residues is considered as an RRNS codeword. A Hamming distance of $\eta \in \{0,1,...,n+k\}$ between the original codeword and the erroneous codeword indicates that η out of $n+k$ residues are erroneous. The erroneous codewords create a new vector space of $n+k$-long vectors where at least one r_i is replaced with $r_i' \neq r_i$ with $i \in \{1,...,n+k\}$ and $r_i' < m_i$. This vector space includes all the RRNS$(n+k,n)$ codewords as well as other possible $n+k$-long vectors that do not overlap with any codeword in the RRNS code space. A vector represents a codeword and is in the RRNS code space if and only if it can be converted into a value within the legitimate range $[0,M)$ of the RRNS$(n+k,n)$ by using the CRT. The number of all vectors that have a Hamming distance η from a codeword in RRNS$(n+k,n)$ can be expressed as

$$V_\eta = \sum_{Q\binom{n+k}{\eta}} \prod_{i=1}^{\eta} (m_i - 1), \tag{A.1}$$

where $Q\binom{n+k}{\eta}$ represents one selection of η moduli from $n+k$ moduli while $\sum_{Q\binom{n+k}{\eta}}$ represents the summation over all distinct $\binom{n+k}{\eta}$ selections. The number of codewords that are in the RNS code space with a Hamming distance of $\eta \in \{0,1,...,n+k\}$ can be expressed as

$$D_\eta = \sum_{h=0}^{\eta-1-k} (-1)^h \binom{n+k-\eta+h}{n+k-\eta} \zeta(n+k, \eta-h), \tag{A.2}$$

for $k+1 \leq \eta \leq n+k$. For $1 \leq \eta \leq k$, $D_\eta = 0$ and $D_0 = 1$. $\zeta(n+k,\eta)$ represents the total number of non-zero common divisors in the legitimate range $[0,M)$ for any $n+k-\eta$ moduli out of the $n+k$ moduli of the RRNS$(n+k,n)$ code and can be denoted as

$$\zeta(n+k,\eta) = \sum_{Q\binom{n+k}{n+k-\eta}} \left\lfloor \frac{M-1}{m_{i_1} m_{i_2} ... m_{i_{(n+k-\eta)}}} \right\rfloor, \tag{A.3}$$

where $(m_{i_1}, m_{i_2}, ..., m_{i_\lambda})$ with $1 \leq \lambda \leq n+k$ is a subset of the RRNS$(n+k,n)$ moduli set.

An undetectable error occurs only if a codeword with errors overlaps with another codeword in the same RRNS space. Given the distance distributions for the vector space V and the codespace D (Eqs. (A.1), (A.2), respectively), the probability of observing an undetectable error (p_u) for RRNS$(n+k,n)$ can be computed as

$$p_u = \sum_{\eta=k+1}^{n+k} \frac{D_\eta}{V_\eta} p_E(\eta), \tag{A.4}$$

where $p_E(\eta)$ is the probability of having η erroneous residues in a codeword which can be calculated as

$$p_E(\eta) = \sum_{Q\binom{n+k}{\eta}} p^\eta (1-p)^{(n+k-\eta)}, \tag{A.5}$$

for a given error probability in a single residue, p.

Eq. (A.2) indicates that for up to $\eta = k$ erroneous residues $D_\eta = 0$, and so an erroneous codeword cannot overlap with another codeword in the RRNS code space. This guarantees

the successful detection of the observed error. If the Hamming distance of the erroneous codeword is $\eta \leq \lfloor \frac{k}{2} \rfloor$, the error can be corrected by the majority logic decoding mechanism. In other words, the probability of observing a correctable error is equal to observing less or equal to $\lfloor \frac{k}{2} \rfloor$ errors in the residues and can be calculated as

$$p_c = \sum_{\eta=0}^{\lfloor \frac{k}{2} \rfloor} p_E(\eta) = \sum_{\eta=0}^{\lfloor \frac{k}{2} \rfloor} \Big(\sum_{Q\binom{n+k}{\eta}} p^{\eta}(1-p)^{(n+k-\eta)} \Big). \tag{A.6}$$

All the errors that do not fall under the undetectable or correctable categories are referred to as detectable but not correctable errors with a probability p_d where $p_d = 1 - (p_c + p_d)$. The equations in this section were collected from the work conducted by (Yang and Hanzo, 2001b).

To model the error in the RNS core for the analysis shown in Fig. 4·6, p_c, p_d, and p_u are computed for a given RRNS$(n+k,n)$ and p value using Eqs. (A.4) and (A.6). Given the number of error correction attempts, p_{err} is calculated according to Eq. (4.3). Random noise is injected at the output of every tiled-MVM operation using a Bernoulli distribution with a probability of p_{err}.

A.2 Extended Dynamic Range via Positional Number System

By combining RNS and PNS, an integer value Z can be represented as D separate digits, z_d where $d \in \{0,1,...,D-1\}$ and $0 \leq z_d < M$:

$$Z = \sum_{d=0}^{D-1} z_d M, \tag{A.7}$$

and can provide up to $D \log_2 M$ bit precision. This hybrid scheme requires carry propagation from lower digits to higher digits, unlike the RNS-only scheme. For this purpose, one can use two sets of moduli, primary and secondary, where every operation is performed for both sets of residues. After every operation, overflow is detected for each digit and carried

over to the next higher-order digit.

Let us define and pick n_p primary moduli m_i where $i \in \{1,...,n_p\}$ and n_s secondary moduli m_j where $j \in \{1,...,n_s\}$, and $m_i \neq m_j \ \forall \ \{i,j\}$. Here $M = M_p \cdot M_s = \prod_{i=1}^{n_p} m_i \cdot \prod_{j=1}^{n_s} m_j$ is large enough to represent the largest possible output of the operations performed in this numeral representation and M_p and M_s are co-prime.

In this hybrid number system, operations for each digit are independent of one another and can be parallelized except for the overflow detection and carry propagation. Assume $z_d = z_d|_{p;s}$ consists of primary and secondary residues and is a calculated output digit of an operation before overflow detection. z_d can be decomposed as $z_d|_p = Q_d|_p M_p + R_d|_p$ where $Q_d|_p$ and $R_d|_p$ are the quotient and the remainder of the digit, with respect to the primary RNS. To detect a potential overflow in the digit z_d, a base extension from primary to secondary RNS is performed on $z_d|_p$ and the base extended residues are compared with the original secondary residues of the digit, $z_d|_s$. If the residues are the same, this indicates that there is no overflow, i.e., $Q_d|_{p;s} = 0$, and both primary and secondary residues are kept without any carry moved to the next higher digit. In contrast, if the base-extended secondary residues and the original secondary residues are not the same, there exists an overflow (i.e., $Q_d|_{p;s} \neq 0$). In the case of overflow, the remainder of the secondary RNS, $R_d|_s$, is calculated through a base extension from primary to secondary RNS on $R_d|_p$ where $R_d|_p = z_d|_p$. $Q_d|_s$ can then be computed as $Q_d|_s = (z_d|_s - R_d|_s)M_p^{-1}$ where $|M_p \cdot M_p^{-1}|_{M_s} \equiv 1$. $Q_d|_p$ is calculated through base extension from the secondary to primary RNS on the computed $Q_d|_s$. The full quotient $Q_d|_{p;s}$ is then propagated to the higher-order digit.

Algorithm 1 shows the pseudo-code for handling an operation $\square$ using the extended RNS representation. The operation can be replaced by any operation that is closed under RNS. It should be noted that $z_d|_{p;s}$ cannot always be computed as $x_d|_{p;s} \square y_d|_{p;s}$. For operations such as addition, each digit before carry propagation is computed by simply adding the same digits of the operands, i.e., $z_d|_{p;s} = x_d|_{p;s} + y_d|_{p;s}$. However, for multiplication,

each digit of $z_d|_{p;s}$ should be constructed as in long multiplication. The multiplication of two numbers in the hybrid number system with D_x and D_y digits requires $D_x D_y$ digit-wise multiplications and the output will result in $D_z = D_x + D_y$ digits in total. Similarly, a dot product is a combination of multiply and add operations. If two vectors with h elements where each element has D_x and D_y digits, the output will require in $D_z = D_x + D_y + \log_2 h$ digits.

Algorithm 1 **Pseudocode for performing a $\square$ operation using the hybrid number system.** Here, x and y are the input operands of $\square$. z_d represents the digits of the output where $d \in \{1,...,D_z\}$, $z_d|_p$ are the primary residues, and $z_d|_s$ are the secondary residues. Primary and secondary residues together are referred to as $z'_d|_{p;s}$. Q is the quotient and R is the remainder where $z_d = Q_d M_p + R_d$. p2s() and s2p() refer to base extension algorithms from primary to secondary residues and from secondary to primary residues, respectively.

$Q_{-1}|_{p;s} = 0$
for d in $(0, D_z)$ **do**
    $z'_d|_{p;s} = (x|_{p;s} \square\, y|_{p;s})_d$
end for
for d in $(0, D_z)$ **do**
    $z_d|_{p;s} = z'_d|_{p;s} + Q_{d-1}|_{p;s}$
    $R_d|_p = z_d|_p$
    $R_d|_s = \text{p2s}(R_d|_p)$
    if $R_d|_s = z'_d|_s$ **then**
        $Q_d|_{p;s} = 0$
    else
        $Q_d|_s = (z'_d|_s - R_d|_s) M_p^{-1}$
        $Q_d|_p = \text{s2p}(Q_d|_s)$
    end if
end for

References

Ahmadifar, H. and Jaberipur, G. (2015). A new residue number system with 5-moduli set:{22 q, 2 q±3, 2 q±1}. *The Computer Journal*, 58(7):1548–1565.

Akiyama, S., Baba, T., Imai, M., Akagawa, T., Takahashi, M., Hirayama, N., Takahashi, H., Noguchi, Y., Okayama, H., Horikawa, T., and Usuki, T. (2012). 12.5-Gb/s operation with 0.29-V·cm VπL using silicon Mach-Zehnder modulator based-on forward-biased pin diode. *Optics Express*, 20(3):2911–2923.

Al-Qadasi, M., Chrostowski, L., Shastri, B., and Shekhar, S. (2022). Scaling up silicon photonic-based accelerators: Challenges and opportunities. *APL Photonics*, 7(2).

Alexander, K., George, J. P., Verbist, J., Neyts, K., Kuyken, B., Van Thourhout, D., and Beeckman, J. (2018). Nanophotonic pockels modulators on a silicon nitride platform. *Nature communications*, 9(1):3444.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR.

Anderson, A., Vasudevan, A., Keane, C., and Gregg, D. (2017). Low-memory gemm-based convolution algorithms for deep neural networks. *arXiv preprint arXiv:1709.03395*.

Ansys (2024). [Online]. Available: https://www.ansys.com/products/photonics.

Baghdadi, R., Gould, M., Gupta, S., Tymchenko, M., Bunandar, D., Ramey, C., and Harris, N. C. (2021). Dual slot-mode noem phase shifter. *Optics Express*, 29(12):19113–19119.

Bahadori, M., Nikdast, M., Cheng, Q., and Bergman, K. (2019). Universal design of waveguide bends in silicon-on-insulator photonics platform. *Journal of Lightwave Technology*, 37(13):3044–3054.

Bandyopadhyay, S., Hamerly, R., and Englund, D. (2021). Hardware error correction for programmable photonics. *Optica*, 8(10):1247–1255.

Bandyopadhyay, S., Sludds, A., Krastanov, S., Hamerly, R., Harris, N., Bunandar, D., Streshinsky, M., Hochberg, M., and Englund, D. (2023). A photonic deep neural network processor on a single chip with optically accelerated training. In *2023 Conference on Lasers and Electro-Optics (CLEO)*, pages 1–2. IEEE.

Bangari, V., Marquez, B. A., Miller, H., Tait, A. N., Nahmias, M. A., De Lima, T. F., Peng, H.-T., Prucnal, P. R., and Shastri, B. J. (2019). Digital electronics and analog photonics for convolutional neural networks (deap-cnns). *IEEE Journal of Selected Topics in Quantum Electronics*, 26(1):1–13.

Bankman, D. and Murmann, B. (2015). Passive charge redistribution digital-to-analogue multiplier. *Electronics Letters*, 51(5):386–388.

Bankman, D. and Murmann, B. (2016). An 8-bit, 16 input, 3.2 pj/op switched-capacitor dot product circuit in 28-nm fdsoi cmos. In *2016 ieee asian solid-state circuits conference (a-sscc)*, pages 21–24. IEEE.

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. (2018). Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31.

Bao, Q., Zhang, H., Ni, Z., Wang, Y., Polavarapu, L., Shen, Z., Xu, Q.-H., Tang, D., and Loh, K. P. (2011). Monolayer Graphene as a Saturable Absorber in a Mode-locked Laser. *Nano Research*, 4(3):297–307.

Basumallik, A., Bunandar, D., Dronen, N., Harris, N., Levkova, L., McCarter, C., Nair, L., Walter, D., and Widemann, D. (2022). Adaptive block floating-point for analog deep learning hardware. *arXiv preprint arXiv:2205.06287*.

Bell, T. E. (1986). Optical Computing: A Field in Flux. *IEEE Spectrum*, 23(8):34–38.

Bernstein, L., Sludds, A., Hamerly, R., Sze, V., Emer, J., and Englund, D. (2021). Freely Scalable and Reconfigurable Optical Hardware for Deep Learning. *Scientific Reports*, 11(1):3144.

Bogaerts, W., De Heyn, P., Van Vaerenbergh, T., De Vos, K., Kumar Selvaraja, S., Claes, T., Dumon, P., Bienstman, P., Van Thourhout, D., and Baets, R. (2012). Silicon Microring Resonators. *Laser & Photonics Reviews*, 6(1):47–73.

Bohr, M. (2007). A 30 year retrospective on dennard's mosfet scaling paper. *IEEE Solid-State Circuits Society Newsletter*, 12(1):11–13.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Cadence (2024). Genus Synthesis Solution. [Online]. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html.

Caulfield, H. J. (1987). Parallel N4 Weighted Optical Interconnections. *Applied Optics*, 26(19):4039–4040.

Chang, J., Sitzmann, V., Dun, X., Heidrich, W., and Wetzstein, G. (2018). Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Scientific reports*, 8(1):12324.

Chen, C. and Joshi, A. (2013). Runtime management of laser power in silicon-photonic multibus noc architecture. *IEEE Journal of Selected Topics in Quantum Electronics*, 19(2):3700713–3700713.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., and Temam, O. (2014a). Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Computer Architecture News*, 42(1):269–284.

Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., et al. (2014b). Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE.

Chen, Y.-H., Krishna, T., Emer, J. S., and Sze, V. (2016). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138.

Chen, Y.-H., Yang, T.-J., Emer, J., and Sze, V. (2019). Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308.

Cheng, Q., Kwon, J., Glick, M., Bahadori, M., Carloni, L. P., and Bergman, K. (2020). Silicon photonics codesign for deep learning. *Proceedings of the IEEE*, 108(8):1261–1282.

Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y., and Xie, Y. (2016). Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 27–39.

Chih, Y.-D., Lee, P.-H., Fujiwara, H., Shih, Y.-C., Lee, C.-F., Naous, R., Chen, Y.-L., Lo, C.-P., Lu, C.-H., Mori, H., et al. (2021). 16.4 an 89tops/w and 16.3 tops/mm 2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 252–254. IEEE.

Choquette, J. (2023). Nvidia hopper h100 gpu: Scaling performance. *IEEE Micro*, 43(3):9–17.

Choquette, J., Gandhi, W., Giroux, O., Stam, N., and Krashinsky, R. (2021). Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35.

Clements, W. R., Humphreys, P. C., Metcalf, B. J., Kolthammer, W. S., and Walmsley, I. A. (2016). Optimal Design for Universal Multiport Interferometers. *Optica*, 3(12):1460–1465.

Cottle, E., Michel, F., Wilson, J., New, N., and Kundu, I. (2020). Optical convolutional neural networks–combining silicon photonics and fourier optics for computer vision. *arXiv preprint arXiv:2103.09044*, :.

Courbariaux, M., Bengio, Y., and David, J.-P. (2014). Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.

Darvish Rouhani, B., Lo, D., Zhao, R., Liu, M., Fowers, J., Ovtcharov, K., Vinogradsky, A., Massengill, S., Yang, L., Bittner, R., et al. (2020). Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point. *Advances in neural information processing systems*, 33:10271–10281.

Dehlaghi, B. and Chan Carusone, A. (2016). A 0.3 pj/bit 20 gb/s/wire parallel interface for die-to-die communication. *IEEE Journal of Solid-State Circuits*, 51(11):2690–2701.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Djath, L., Bigou, K., and Tisserand, A. (2019). Hierarchical approach in rns base extension for asymmetric cryptography. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pages 46–53. IEEE.

Dong, P., Qian, W., Liang, H., Shafiiha, R., Feng, N.-N., Feng, D., Zheng, X., Krishnamoorthy, A. V., and Asghari, M. (2010). Low power and compact reconfigurable multiplexing devices based on silicon microring resonators. *Optics Express*, 18(10):9852–9858.

Drumond, M., Lin, T., Jaggi, M., and Falsafi, B. (2018). Training dnns with hybrid block floating point. *Advances in Neural Information Processing Systems*, 31.

Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., Feng, X., Chen, Y., and Temam, O. (2015). Shidiannao: Shifting vision processing closer to the sensor. In *Proceedings of the 42nd annual international symposium on computer architecture*, pages 92–104.

Epoch AI (2023). Key trends and figures in machine learning. [Online]. Available: https://epochai.org/trends.

Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., and Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Proceedings of the 38th annual international symposium on Computer architecture*, pages 365–376.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338.

Fang, A. W., Park, H., Cohen, O., Jones, R., Paniccia, M. J., and Bowers, J. E. (2006). Electrically pumped hybrid algainas-silicon evanescent laser. *Optics express*, 14(20):9203–9210.

Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., and Talay, S. (2011a). Large-scale fpga-based convolutional networks. *Scaling up machine learning: parallel and distributed approaches*, 13(3):399–419.

Farabet, C., Martini, B., Corda, B., Akselrod, P., Culurciello, E., and LeCun, Y. (2011b). Neuflow: A runtime reconfigurable dataflow processor for vision. In *CVPR 2011 WORKSHOPS*, pages 109–116.

Farabet, C., Poulet, C., Han, J. Y., and LeCun, Y. (2009). Cnp: An fpga-based processor for convolutional networks. In *2009 International Conference on Field Programmable Logic and Applications*, pages 32–37. IEEE.

Feldmann, J., Youngblood, N., Karpov, M., Gehring, H., Li, X., Stappers, M., Le Gallo, M., Fu, X., Lukashchuk, A., Raja, A. S., et al. (2021). Parallel convolutional processing using an integrated photonic tensor core. *Nature*, 589(7840):52–58.

Feldmann, J., Youngblood, N., Wright, C. D., Bhaskaran, H., and Pernice, W. H. (2019). All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature*, 569(7755):208–214.

Feng, Y., Thomson, D. J., Mashanovich, G. Z., and Yan, J. (2020a). Performance analysis of a silicon noems device applied as an optical modulator based on a slot waveguide. *Optics Express*, 28(25):38206–38222.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., et al. (2020b). Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.

Ferreira De Lima, T., Tait, A. N., Mehrabian, A., Nahmias, M. A., Huang, C., Peng, H.-T., Marquez, B. A., Miscuglio, M., El-Ghazawi, T., Sorger, V. J., et al. (2020). Primer on silicon neuromorphic photonic processors: architecture and compiler. *Nanophotonics*, 9(13):4055–4073.

Filipovich, M. J., Guo, Z., Al-Qadasi, M., Marquez, B. A., Morison, H. D., Sorger, V. J., Prucnal, P. R., Shekhar, S., and Shastri, B. J. (2022). Silicon photonic architecture for training deep neural networks with direct feedback alignment. *Optica*, 9(12):1323–1332.

Fujikata, J., Takahashi, S., Takahashi, M., Noguchi, M., Nakamura, T., and Arakawa, Y. (2016). High-performance mos-capacitor-type si optical modulator and surface-illumination-type ge photodetector for optical interconnection. *Japanese Journal of Applied Physics*, 55(4S):04EC01.

Gao, M., Pu, J., Yang, X., Horowitz, M., and Kozyrakis, C. (2017). Tetris: Scalable and efficient neural network acceleration with 3d memory. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–764.

Garello, K., Yasin, F., and Kar, G. S. (2019). Spin-orbit torque mram for ultrafast embedded memories: From fundamentals to large scale technology integration. In *2019 IEEE 11th International Memory Workshop (IMW)*, pages 1–4. IEEE.

Garg, S., Lou, J., Jain, A., Guo, Z., Shastri, B. J., and Nahmias, M. (2023). Dynamic precision analog computing for neural networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 29(2: Optical Computing):1–12.

Gargini, P. A. (2023). Overcoming semiconductor and electronics crises with irds: Planning for the future. *IEEE Electron Devices Magazine*, 1(3):32–47.

Gholami, E., Farshidi, R., Hosseinzadeh, M., and Navi, K. (2009). High speed residue number system comparison for the moduli set $\{2n\text{-}1, 2n, 2n+ 1\}$. *Journal of communication and computer*, 6(3):40–46.

Giewont, K., Nummy, K., Anderson, F. A., Ayala, J., Barwicz, T., Bian, Y., Dezfulian, K. K., Gill, D. M., Houghton, T., Hu, S., Peng, B., Rakowski, M., Rauch, S., Rosenberg, J. C., Sahin, A., Stobert, I., and Stricker, A. (2019). 300-mm Monolithic Silicon Photonics Foundry Technology. *IEEE Journal of Selected Topics in Quantum Electronics*, 25(5):1–11.

Global Foundries (2024). GF22nm FD-SOI Technology. [Online]. Available: https://gf.com/technology-platforms/fdx-fd-soi/.

Google Cloud (2024). Tpu v5p. [Online]. Available: https://cloud.google.com/tpu/docs/v5p.

Gregory, R. T. and Matula, D. W. (1975). Base conversion in residue number systems. In *1975 IEEE 3rd Symposium on Computer Arithmetic (ARITH)*, pages 117–125. IEEE.

Gschwend, D. (2020). Zynqnet: An fpga-accelerated embedded convolutional neural network. *arXiv preprint arXiv:2005.06892*.

Guo, M., Mao, J., Sin, S.-W., Wei, H., and Martins, R. P. (2020). A 5 GS/s 29 mW Interleaved SAR ADC With 48.5 dB SNDR Using Digital-Mixing Background Timing-Skew Calibration for Direct Sampling Applications. *IEEE Access*, 8:138944–138954.

Gupta, R., Pal, S., Kanade, A., and Shevade, S. (2017). Deepfix: Fixing common c language errors by deep learning. In *Proceedings of the aaai conference on artificial intelligence*, volume 31.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. (2015). Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR.

Hamerly, R., Bandyopadhyay, S., and Englund, D. (2021a). Accurate Self-Configuration of Rectangular Multiport Interferometers. *arXiv*, abs/2106.03249:.

Hamerly, R., Bandyopadhyay, S., and Englund, D. (2021b). Stability of Self-Configuring Large Multiport Interferometers. *arXiv*, abs/2106.04363:.

Han, J.-H., Boeuf, F., Fujikata, J., Takahashi, S., Takagi, S., and Takenaka, M. (2017). Efficient low-loss ingaasp/si hybrid mos optical modulator. *Nature Photonics*, 11(8):486–490.

Harris, N. C., Bunandar, D., Joshi, A., Basumallik, A., and Turner, R. (2022). Passage: A wafer-scale programmable photonic communication substrate. In *2022 IEEE Hot Chips 34 Symposium (HCS)*, pages 1–26. IEEE Computer Society.

Harris, N. C., Ma, Y., Mower, J., Baehr-Jones, T., Englund, D., Hochberg, M., and Galland, C. (2014). Efficient, compact and low loss thermo-optic phase shifter in silicon. *Optics express*, 22(9):10487–10493.

Hauser, J. R. (2019). Bsg-external/hardfloat. [Online]. Available: https://github.com/bsg-external/HardFloat.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, Y., Sainath, T. N., Prabhavalkar, R., McGraw, I., Alvarez, R., Zhao, D., Rybach, D., Kannan, A., Wu, Y., Pang, R., Liang, Q., Bhatia, D., Shangguan, Y., Li, B., Pundak, G., Sim, K. C., Bagby, T., Chang, S.-y., Rao, K., and Gruenstein, A. (2019). Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

Hiasat, A. (2019). A residue-to-binary converter with an adjustable structure for an extended rns three-moduli set. *Journal of Circuits, Systems and Computers*, 28(08):1950126.

Hobbhahn, M., Heim, L., and Aydos, G. (2023). Trends in machine learning hardware. [Online]. Available: https://epochai.org/blog/trends-in-machine-learning-hardware.

Hollmann, H. D., Rietman, R., de Hoogh, S., Tolhuizen, L., and Gorissen, P. (2018). A multi-layer recursive residue number system. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1460–1464. IEEE.

Horowitz, M. (2014). Computing's Energy Problem (and What We Can Do About It). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, . .

Hsu, K., Brady, D., and Psaltis, D. (1988). Experimental Demonstrations of Optical Neural Computers. In Anderson, D., editor, *Neural Information Processing Systems*. American Institute of Physics.

Hu, M., Strachan, J. P., Li, Z., Grafals, E. M., Davila, N., Graves, C., Lam, S., Ge, N., Yang, J. J., and Williams, R. S. (2016). Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *Proceedings of the 53rd annual design automation conference*, pages 1–6.

Hu, R., Sun, L., Zhang, Z., Sun, Q., Pan, Y., and Su, Y. (2023). Ultrabroadband and compact 2$\times$ 2 3-db coupler based on trapezoidal subwavelength gratings. *Optics Express*, 31(14):23542–23550.

Huang, H.-Y., Chen, X.-Y., and Kuo, T.-H. (2021). A 10-GS/s NRZ/Mixing DAC With Switching-Glitch Compensation Achieving SFDR gt; 64/50 dBc Over the First/Second Nyquist Zone. *IEEE Journal of Solid-State Circuits*, ():1–1.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Huggingface. Huggingface/transformers: State-of-the-art machine learning for pytorch, tensorflow, and jax. [Online]. Available: https://github.com/huggingface/transformers.

Hughes, T. W., Minkov, M., Shi, Y., and Fan, S. (2018). Training of photonic neural networks through in situ backpropagation and gradient measurement. *Optica*, 5(7):864–871.

Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, and Lingli Wang (2016). A High Performance FPGA-based Accelerator for Large-scale Convolutional Neural Networks. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9, . .

Intel (2024). Intel® xeon® gold 6242 processor (22m cache, 2.80 ghz) product specifications. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/192440/intel-xeon-gold-6242-processor-22m-cache-2-80-ghz.html.

Intel Corporation (2021). Intel® advanced vector extensions 512. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/avx-512-solution-brief.html.

Intel Corporation (2023). 4th gen intel® xeon® scalable processors. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/processors/xeon-accelerated/4th-gen-xeon-scalable-processors-product-brief.html.

Jackson, J. D. (1975). *Classical Electrodynamics*. Wiley, New York, NY.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Jain, S., Ranjan, A., Roy, K., and Raghunathan, A. (2017). Computing in memory with spin-transfer torque magnetic ram. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(3):470–483.

James, J. and Pe, A. (2015). Error correction based on redundant residue number system. In *2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–5. IEEE.

Jayatilleka, H., Shoman, H., Chrostowski, L., and Shekhar, S. (2019). Photoconductive heaters enable control of large-scale silicon photonic ring resonator circuits. *Optica*, 6(1):84–91.

Jenkins, W. K. (1980). Complex residue number arithmetic for high-speed signal processing. *Electronics Letters*, 16:660.

Jha, A., Huang, C., and Prucnal, P. R. (2020). Reconfigurable all-optical nonlinear activation functions for neuromorphic photonics. *Optics letters*, 45(17):4819–4822.

Joshi, V., Le Gallo, M., Haefeli, S., Boybat, I., Nandakumar, S. R., Piveteau, C., Dazzi, M., Rajendran, B., Sebastian, A., and Eleftheriou, E. (2020). Accurate deep neural network inference using computational phase-change memory. *Nature communications*, 11(1):2473.

Jouppi, N. P., Yoon, D. H., Kurian, G., Li, S., Patil, N., Laudon, J., Young, C., and Patterson, D. (2020). A domain-specific supercomputer for training deep neural networks. *Commun. ACM*, 63(7):67–78.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kenyon, A. (2005). Erbium in silicon. *Semiconductor Science and Technology*, 20(12):R65.

Kieninger, C., Kutuvantavida, Y., Elder, D. L., Wolf, S., Zwickel, H., Blaicher, M., Kemal, J. N., Lauermann, M., Randel, S., Freude, W., et al. (2018). Ultra-high electro-optic activity demonstrated in a silicon-organic hybrid modulator. *Optica*, 5(6):739–748.

Kim, D., Kung, J., Chai, S., Yalamanchili, S., and Mukhopadhyay, S. (2016). Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory. *ACM SIGARCH Computer Architecture News*, 44(3):380–392.

Kim, S., Kim, J., Kim, M. J., Jung, W., Rhu, M., Kim, J., and Ahn, J. H. (2021). Bts: An accelerator for bootstrappable fully homomorphic encryption. *arXiv preprint arXiv:2112.15479*, :.

Kim, S.-N., Kim, W.-C., Seo, M.-J., and Ryu, S.-T. (2018). A 65-nm cmos 6-bit 20 gs/s time-interleaved dac with full-binary sub-dacs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(9):1154–1158.

Kim, W., Bruce, R. L., Masuda, T., Fraczak, G., Gong, N., Adusumilli, P., Ambrogio, S., Tsai, H., Bruley, J., Han, J.-P., et al. (2019). Confined pcm-based analog synaptic devices offering low resistance-drift and 1000 programmable states for deep learning. In *2019 Symposium on VLSI Technology*, pages T66–T67. IEEE.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342:.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Lamb, C. C. and DeBrunner, L. (1995). A table-lookup scheme for residue-to-binary conversion. In *Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 214–217. IEEE.

Lavely, A. (2022). Powering extreme-scale hpc with cerebras waferscale accelerators. Technical report, Cerebras Systems. [Online]. Available: https://8968533.fs1.hubspotusercontent-na1.net/hubfs/8968533/Powering-Extreme-Scale-HPC-with-Cerebras.pdf.

Le Gallo, M., Khaddam-Aljameh, R., Stanisavljevic, M., Vasilopoulos, A., Kersting, B., Dazzi, M., Karunaratne, G., Brändli, M., Singh, A., Mueller, S. M., et al. (2023). A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nature Electronics*, 6(9):680–693.

Le Sueur, E. and Heiser, G. (2010). Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*, pages 1–8.

Lee, J., Kim, C., Kang, S., Shin, D., Kim, S., and Yoo, H.-J. (2018). Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision. *IEEE Journal of Solid-State Circuits*, 54(1):173–185.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436.

Li, S., Yang, H., Wong, C. W., Sorger, V. J., and Gupta, P. (2023). Photofourier: A photonic joint transform correlator-based neural network accelerator. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 15–28. IEEE.

Li, X. and Zhou, L. (2020). A survey of high-speed high-resolution current steering DACs. *Journal of Semiconductors*, 41(20060024):111404.

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., and Gonzalez, J. (2020). Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR.

Liang, D. and Bowers, J. E. (2010). Recent progress in lasers on silicon. *Nature photonics*, 4(8):511–517.

Liang, G., Huang, H., Mohanty, A., Shin, M. C., Ji, X., Carter, M. J., Shrestha, S., Lipson, M., and Yu, N. (2021). Robust, efficient, micrometre-scale phase modulators at visible wavelengths. *Nature Photonics*, 15(12):908–913.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Lin, X., Rivenson, Y., Yardimci, N. T., Veli, M., Luo, Y., Jarrahi, M., and Ozcan, A. (2018). All-optical machine learning using diffractive deep neural networks. *Science*, 361(6406):1004–1008.

Lischke, S., Knoll, D., Mai, C., Zimmermann, L., Peczek, A., Kroh, M., Trusch, A., Krune, E., Voigt, K., and Mai, A. (2015). High Bandwidth, High Responsivity Waveguide-coupled Germanium p-i-n Photodiode. *Optics Express*, 23(21):27213–27220.

Liu, J., Sun, X., Camacho-Aguilera, R., Kimerling, L. C., and Michel, J. (2010). Ge-on-si laser operating at room temperature. *Optics letters*, 35(5):679–681.

Liu, W., Liu, W., Ye, Y., Lou, Q., Xie, Y., and Jiang, L. (2019). Holylight: A nanophotonic accelerator for deep learning in data centers. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1483–1488. IEEE.

Maslej, N., Fattorini, L., Perrault, R., Parli, V., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R., and Clark, J. (2024). Artificial intelligence index report 2024. [Online]. Available: https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_2024_AI-Index-Report.pdf.

Mehrabian, A., Al-Kabani, Y., Sorger, V. J., and El-Ghazawi, T. (2018). Pcnna: A photonic convolutional neural network accelerator. In *2018 31st IEEE International System-on-Chip Conference (SOCC)*, pages 169–173. IEEE.

Meng, J., Miscuglio, M., George, J. K., Babakhani, A., and Sorger, V. J. (2021). Electronic bottleneck suppression in next-generation networks with integrated photonic digital-to-analog converters. *Advanced photonics research*, 2(2):2000033.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Mikolajick, T., Schroeder, U., and Slesazeck, S. (2020). The past, the present, and the future of ferroelectric memories. *IEEE Transactions on Electron Devices*, 67(4):1434–1443.

Milanizadeh, M., Aguiar, D., Melloni, A., and Morichetti, F. (2019). Canceling thermal cross-talk effects in photonic integrated circuits. *Journal of Lightwave Technology*, 37(4):1325–1332.

Miller, D. A. B. (2013). Self-configuring Universal Linear Optical Component. *Photonics Research*, 1(1):1–15.

Miller, D. A. B. (2015). Perfect optics with imperfect components. *Optica*, 2(8):747–750.

Mirza, A., Sunny, F., Walsh, P., Hassan, K., Pasricha, S., and Nikdast, M. (2022). 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*, 41(10):3359–3372.

Miscuglio, M., Hu, Z., Li, S., George, J., Capanna, R., Bardet, P. M., Gupta, P., and Sorger, V. J. (2020). Massively parallel amplitude-only fourier neural network. *arXiv preprint arXiv:2008.05853*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Mohan, P. A. (2002). *Residue number systems: algorithms and architectures*. Springer Science & Business Media.

Mohan, P. V. A. (2007). Rns-to-binary converter for a new three-moduli set $\{2^n + 1, 2^n, 2^n - 1\}$. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 54(9):775–779.

Mourgias-Alexandris, G., Tsakyridis, A., Passalis, N., Tefas, A., Vyrsokinos, K., and Pleros, N. (2019). An all-optical neuron with sigmoid activation function. *Optics express*, 27(7):9620–9630.

Mourou, G., Brocklesby, B., Tajima, T., and Limpert, J. (2013). The future is fibre accelerators. *Nature Photonics*, 7(4):258–261.

Murmann, B. (2021). Mixed-signal computing for deep neural network inference. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(1):3–13.

Murmann, B. (2022). Introduction to adcs/dacs: metrics, topologies, trade space, and applications. *ISSCC Short Course*.

Murmann, B. (2024). ADC Performance Survey 1997-2024. [Online]. Available: https://github.com/bmurmann/ADC-survey.

Nahmias, M. A., De Lima, T. F., Tait, A. N., Peng, H.-T., Shastri, B. J., and Prucnal, P. R. (2019). Photonic multiply-accumulate operations for neural networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(1):1–18.

Ning, S., Zhu, H., Feng, C., Gu, J., Jiang, Z., Ying, Z., Midkiff, J., Jain, S., Hlaing, M. H., Pan, D. Z., et al. (2024). Photonic-electronic integrated circuits for high-performance computing and ai accelerator. *arXiv preprint arXiv:2403.14806*.

NVIDIA Corporation (2024a). Cuda toolkit. [Online]. Available: https://developer.nvidia.com/cuda-toolkit.

NVIDIA Corporation (2024b). Nvidia dgx b200. [Online]. Available: https://www.nvidia.com/en-us/data-center/dgx-b200/.

Ohno, S., Li, Q., Sekine, N., Tang, H., Monfray, S., Boeuf, F., Toprasertpong, K., Takagi, S., and Takenaka, M. (2021). Si microring resonator optical switch based on optical phase shifter with ultrathin-inp/si hybrid metal-oxide-semiconductor capacitor. *Optics Express*, 29(12):18502–18511.

Ohno, S., Tang, R., Toprasertpong, K., Takagi, S., and Takenaka, M. (2022). Si microring resonator crossbar array for on-chip inference and training of the optical neural network. *Acs Photonics*, 9(8):2614–2622.

Ordentlich, O., Tabak, G., Hanumolu, P. K., Singer, A. C., and Wornell, G. W. (2018). A modulo-based architecture for analog-to-digital conversion. *IEEE journal of selected topics in signal processing*, 12(5):825–840.

Padmaraju, K. and Bergman, K. (2014). Resolving the thermal challenges for silicon microring resonator devices. *Nanophotonics*, 3(4-5):269–281.

Pai, S., Sun, Z., Hughes, T. W., Park, T., Bartlett, B., Williamson, I. A., Minkov, M., Milanizadeh, M., Abebe, N., Morichetti, F., et al. (2023). Experimentally realized in situ backpropagation for deep learning in photonic neural networks. *Science*, 380(6643):398–404.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR Corpus Based on Public Domain Audio Books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, . .

Patel, D., Samani, A., Veerasubramanian, V., Ghosh, S., and Plant, D. V. (2015). Silicon photonic segmented modulator-based electro-optic dac for 100 gb/s pam-4 generation. *IEEE Photonics Technology Letters*, 27(23):2433–2436.

Patel, D., Veerasubramanian, V., Ghosh, S., Samani, A., Zhong, Q., and Plant, D. V. (2014). High-speed compact silicon photonic michelson interferometric modulator. *Optics express*, 22(22):26788–26802.

Peng, J., Alkabani, Y., Sun, S., Sorger, V. J., and El-Ghazawi, T. (2020). Dnnara: A deep neural network accelerator using residue arithmetic and integrated photonics. In *Proceedings of the 49th International Conference on Parallel Processing*, pages 1–11.

Petraru, A., Schubert, J., Schmid, M., and Buchal, C. (2002). Ferroelectric batio 3 thin-film optical waveguide modulators. *Applied Physics Letters*, 81(8):1375–1377.

PowerAPI (2024). Powerapi-ng/pyrapl: A library to measure the python energy consumption of python code. [Online]. Available: https://github.com/powerapi-ng/pyRAPL.

Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., Yu, J., Tang, T., Xu, N., Song, S., et al. (2016). Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 26–35.

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rakowski, M., Ban, Y., De Heyn, P., Pantano, N., Snyder, B., Balakrishnan, S., Van Huylenbroeck, S., Bogaerts, L., Demeurisse, C., Inoue, F., Rebibis, K. J., Nolmans, P., Sun, X., Bex, P., Srinivasan, A., De Coster, J., Lardenois, S., Miller, A., Absil, P., Verheyen, P., Velenis, D., Pantouvaki, M., and Van Campenhout, J. (2018). Hybrid 14nm finfet - silicon photonics technology for low-power tb/s/mm2 optical i/o. In *2018 IEEE Symposium on VLSI Technology*, pages 221–222.

Ramey, C. (2020). Silicon photonics for artificial intelligence acceleration : Hotchips 32. In *IEEE Hot Chips 32 Symposium, HCS 2020, Palo Alto, CA, USA, August 16-18, 2020*, pages 1–26, . IEEE.

Ramon, H., Vanhoecke, M., Verbist, J., Soenen, W., De Heyn, P., Ban, Y., Pantouvaki, M., Van Campenhout, J., Ossieur, P., Yin, X., et al. (2018). Low-power 56gb/s nrz microring modulator driver in 28nm fdsoi cmos. *IEEE Photonics Technology Letters*, 30(5):467–470.

Reck, M., Zeilinger, A., Bernstein, H. J., and Bertani, P. (1994). Experimental Realization of Any Discrete Unitary Operator. *Physics Review Letters*, 73:58–61.

Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C., Anderson, B., Breughe, M., Charlebois, M., Chou, W., Chukka, R., Coleman, C., Davis, S., Deng, P., Diamos, G., Duke, J., Fick, D., Gardner, J. S., Hubara, I., Idgunji, S., Jablin, T. B., Jiao, J., John, T. S., Kanwar, P., Lee, D., Liao, J., Lokhmotov, A., Massa, F., Meng, P., Micikevicius, P., Osborne, C., Pekhimenko, G., Rajan, A. T. R., Sequeira, D., Sirasao, A., Sun, F., Tang, H., Thomson, M., Wei, F., Wu, E., Xu, L., Yamada, K., Yu, B., Yuan, G., Zhong, A., Zhang, P., and Zhou, Y. (2020). MLPerf Inference Benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459, . .

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.

Reimann, O. A. and Kosonocky, W. F. (1965). Progress in Optical Computer Research. *IEEE Spectrum*, 2(3):181 – 195.

Roelkens, G., Van Campenhout, J., Brouckaert, J., Van Thourhout, D., Baets, R., Romeo, P. R., Regreny, P., Kazmierczak, A., Seassal, C., Letartre, X., Hollinger, G., Fedeli, J., Di Cioccio, L., and Lagahe-Blanchard, C. (2007). III-V/Si Photonics by Die-to-wafer Bonding. *Materials Today*, 10(7):36–43.

Sacher, W. D., Luo, X., Yang, Y., Chen, F.-D., Lordello, T., Mak, J. C., Liu, X., Hu, T., Xue, T., Lo, P. G.-Q., et al. (2019). Visible-light silicon nitride waveguide devices and implantable neurophotonic probes on thinned 200 mm silicon wafers. *Optics express*, 27(26):37400–37418.

Sacher, W. D., Mikkelsen, J. C., Huang, Y., Mak, J. C., Yong, Z., Luo, X., Li, Y., Dumais, P., Jiang, J., Goodwill, D., et al. (2018). Monolithically integrated multilayer silicon nitride-on-silicon waveguide platforms for 3-d photonic circuits and devices. *Proceedings of the IEEE*, 106(12):2232–2245.

Salamat, S., Imani, M., Gupta, S., and Rosing, T. (2018). Rnsnet: In-memory neural network acceleration using residue number system. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–12.

Samajdar, A., Zhu, Y., Whatmough, P. N., Mattina, M., and Krishna, T. (2018). SCALE-Sim: Systolic CNN Accelerator. *arXiv*, abs/1811.02883:.

Samimi, N., Kamal, M., Afzali-Kusha, A., and Pedram, M. (2020). Res-dnn: A residue number system-based dnn accelerator unit. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(2):658–671.

Sancho, J. C. and Kerbyson, D. J. (2008). Analysis of double buffering on two different multicore architectures: Quad-core Opteron and the Cell-BE. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–12, . .

Sankaradas, M., Jakkula, V., Cadambi, S., Chakradhar, S., Durdanovic, I., Cosatto, E., and Graf, H. P. (2009). A Massively Parallel Coprocessor for Convolutional Neural Networks. In *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 53–60, . .

Scherer, D., Schulz, H., and Behnke, S. (2010). Accelerating Large-Scale Convolutional Neural Networks with Parallel Graphics Multiprocessors. In Diamantaras, K., Duch, W., and Iliadis, L. S., editors, *Artificial Neural Networks – ICANN 2010*, pages 82–91, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. (2022). Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Shafiee, A., Banerjee, S., Chakrabarty, K., Pasricha, S., and Nikdast, M. (2022). Loci: An analysis of the impact of optical loss and crosstalk noise in integrated silicon-photonic neural networks. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pages 351–355, . .

Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S., and Srikumar, V. (2016). Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26.

Shastri, B. J., Tait, A. N., Ferreira de Lima, T., Pernice, W. H. P., Bhaskaran, H., Wright, C. D., and Prucnal, P. R. (2021). Photonics for Artificial Intelligence and Neuromorphic Computing. *Nature Photonics*, 15(2):102–114.

Shen, L., Margolies, L. R., Rothstein, J. H., Fluder, E., McBride, R., and Sieh, W. (2019). Deep learning to improve breast cancer detection on screening mammography. *Scientific reports*, 9(1):12495.

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., Sun, X., Zhao, S., Larochelle, H., Englund, D., and Soljačić, M. (2017). Deep Learning with Coherent Nanophotonic Circuits. *Nature Photonics*, 11(7):441–446.

Shen, Yun, Wang, Xiaodong, Zhang, Wei, Qiu, Ciyuan, and Cheng, Xiulan (2017). Fabrication of Depletion Type Micro-ring Modulator with High Extinction Ratio and High Coupling Quality Factor. *MATEC Web Conf.*, 139:00066.

Shenoy, A. and Kumaresan, R. (1989). Fast base extension using a redundant modulus in rns. *IEEE Transactions on Computers*, 38(2):292–297.

Shi, B., Calabretta, N., and Stabile, R. (2020). Deep Neural Network Through an InP SOA-Based Photonic Integrated Cross-Connect. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(1):1–11.

Shi, Y., Oh, S., Huang, Z., Lu, X., Kang, S. H., and Kuzum, D. (2020). Performance prospects of deeply scaled spin-transfer torque magnetic random-access memory for in-memory computing. *IEEE Electron Device Letters*, 41(7):1126–1129.

Shiflett, K., Karanth, A., Bunescu, R., and Louri, A. (2021). Albireo: Energy-efficient acceleration of convolutional neural networks via silicon photonics. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 860–873. IEEE.

Shiflett, K., Wright, D., Karanth, A., and Louri, A. (2020). Pixel: Photonic neural network accelerator. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 474–487. IEEE.

Shiflett, K. D. (2022). *Photonic Deep Neural Network Accelerators for Scaling to the Next Generation of High-Performance Processing*. Doctoral dissertation, Ohio University. [Online]. Available: .

Shokraneh, F., Geoffroy-Gagnon, S., and Liboiron-Ladouceur, O. (2020). The diamond mesh, a phase-error-and loss-tolerant field-programmable mzi-based optical processor for optical neural networks. *Optics Express*, 28(16):23495–23508.

Sie, S.-H., Lee, J.-L., Chen, Y.-R., Yeh, Z.-W., Li, Z., Lu, C.-C., Hsieh, C.-C., Chang, M.-F., and Tang, K.-T. (2021). Mars: Multimacro architecture sram cim-based accelerator with co-designed compressed neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(5):1550–1562.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, L., Chen, T., Liu, W., Liu, H., Peng, Y., Yu, Z., Li, H., Shi, Y., and Dai, D. (2022). Toward calibration-free mach–zehnder switches for next-generation silicon photonics. *Photonics Research*, 10(3):793–801.

Song, L., Qian, X., Li, H., and Chen, Y. (2017). Pipelayer: A pipelined reram-based accelerator for deep learning. In *2017 IEEE international symposium on high performance computer architecture (HPCA)*, pages 541–552. IEEE.

Song, Z., Liu, Z., and Wang, D. (2018). Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Sorianello, V., Midrio, M., Contestabile, G., Asselberghs, I., Van Campenhout, J., Huyghebaert, C., Goykhman, I., Ott, A., Ferrari, A., and Romagnoli, M. (2018). Graphene–silicon phase modulators with gigahertz bandwidth. *Nature Photonics*, 12(1):40–44.

Stojanović, V., Ram, R. J., Popović, M., Lin, S., Moazeni, S., Wade, M., Sun, C., Alloatti, L., Atabaki, A., Pavanello, F., et al. (2018). Monolithic silicon-photonic platforms in state-of-the-art cmos soi processes. *Optics express*, 26(10):13106–13121.

Stosic, D. and Micikevicius, P. (2021). Accelerating ai training with nvidia tf32 tensor cores. [Online]. Available: https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/.

Sui, X., Wu, Q., Liu, J., Chen, Q., and Gu, G. (2020). A review of optical neural networks. *IEEE Access*, 8:70773–70783.

Sukhdeo, D. S., Lin, H., Nam, D., Yuan, Z., Vulovic, B. M., Gupta, S., Harris, J. S., Dutt, B., and Saraswat, K. C. (2013). Approaches for a viable germanium laser: tensile strain, gesn alloys, and n-type doping. In *2013 Optical Interconnects Conference*, pages 112–113. IEEE.

Sun, C., Wade, M., Lee, Y., Orcutt, J., Alloatti, L., Georgas, M., Waterman, A., Shainline, J., Avizienis, R., Lin, S., Moss, B., Kumar, R., Pavanello, F., Atabaki, A., Cook, H., Ou, A. J., Leu, J., hsin Chen, Y., Asanović, K., Ram, R. J., Popovic, M., and Stojanović, V. (2015). Single-chip microprocessor that communicates directly using light. *Nature*, 528:534–538.

Sun, J., Kumar, R., Sakib, M., Driscoll, J. B., Jayatilleka, H., and Rong, H. (2018). A 128 gb/s pam4 silicon microring modulator with integrated thermo-optic resonance tuning. *Journal of Lightwave Technology*, 37(1):110–115.

Sun, X., Choi, J., Chen, C.-Y., Wang, N., Venkataramani, S., Srinivasan, V. V., Cui, X., Zhang, W., and Gopalakrishnan, K. (2019). Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. *Advances in neural information processing systems*, 32.

Sun, X., Zadok, A., Shearn, M. J., Diest, K. A., Ghaffari, A., Atwater, H. A., Scherer, A., and Yariv, A. (2009). Electrically pumped hybrid evanescent si/ingaasp lasers. *Optics letters*, 34(9):1345–1347.

Sunny, F., Mirza, A., Nikdast, M., and Pasricha, S. (2021). Crosslight: A cross-layer optimized silicon photonic neural network accelerator. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1069–1074. IEEE.

Tait, A. N., De Lima, T. F., Zhou, E., Wu, A. X., Nahmias, M. A., Shastri, B. J., and Prucnal, P. R. (2017). Neuromorphic photonic networks using silicon photonic weight banks. *Scientific reports*, 7(1):7430.

Tait, A. N., Wu, A. X., De Lima, T. F., Zhou, E., Shastri, B. J., Nahmias, M. A., and Prucnal, P. R. (2016). Microring weight banks. *IEEE Journal of Selected Topics in Quantum Electronics*, 22(6):312–325.

Tanabe, K., Watanabe, K., and Arakawa, Y. (2012). Iii-v/si hybrid photonic devices by direct fusion bonding. *Scientific reports*, 2(1):349.

Tang, J., Bishop, D., Kim, S., Copel, M., Gokmen, T., Todorov, T., Shin, S., Lee, K.-T., Solomon, P., Chan, K., et al. (2018). Ecram as scalable synaptic cell for high-speed, low-power neuromorphic computing. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 13–1. IEEE.

Tang, T., Xia, L., Li, B., Wang, Y., and Yang, H. (2017). Binary convolutional neural network on rram. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 782–787. IEEE.

Taylor, M. G. (2009). Phase estimation methods for optical coherent detection using digital signal processing. *Journal of Lightwave Technology*, 27:901–914.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Theis, T. N. and Wong, H.-S. P. (2017). The end of moore's law: A new beginning for information technology. *Computing in Science Engineering*, 19(2):41–50.

Thonnart, Y., Zid, M., Gonzalez-Jimenez, J. L., Waltener, G., Polster, R., Dubray, O., Lepin, F., Bernabé, S., Menezo, S., Parès, G., Castany, O., Boutafa, L., Grosse, P., Charbonnier, B., and Baudot, C. (2018). A 10gb/s si-photonic transceiver with 150 μw 120 μs-lock-time digitally supervised analog microring wavelength stabilization for 1tb/s/mm2 die-to-die optical networks. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 350–352, . .

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

TSMC (2024). 40nm technology. [Online]. Available: https://www.tsmc.com/english/dedicatedFoundry/technology/logic.

Van Campenhout, J., Rojo-Romeo, P., Van Thourhout, D., Seassal, C., Regreny, P., Di Cioccio, L., Fedeli, J.-M., and Baets, R. (2007). Thermal characterization of electrically injected thin-film ingaasp microdisk lasers on si. *Journal of Lightwave technology*, 25(6):1543–1548.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., et al. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Verhelst, M. and Murmann, B. (2012). Area scaling analysis of cmos adcs. *Electronics letters*, 48(6):1.

Wang, S. and Kanwar, P. (2019). Bfloat16: The secret to high performance on cloud tpus. [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus.

Wang, W., Swamy, M., Ahmad, M., and Wang, Y. (2000). A high-speed residue-to-binary converter for three-moduli (2/sup k/, 2/sup k/-1, 2/sup k-1/-1) rns and a scheme for its vlsi implementation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(12):1576–1581.

Wang, Y., Song, X., Aboulhamid, M., and Shen, H. (2002). Adder based residue to binary number converters for (2/sup n/-1, 2/sup n/, 2/sup n/+1). *IEEE Transactions on Signal Processing*, 50(7):1772–1779.

Watts, M. R., Sun, J., DeRose, C., Trotter, D. C., Young, R. W., and Nielson, G. N. (2013). Adiabatic thermo-optic mach–zehnder switch. *Optics letters*, 38(5):733–735.

Wetzstein, G., Ozcan, A., Gigan, S., Fan, S., Englund, D., Soljačić, M., Denz, C., Miller, D. A. B., and Psaltis, D. (2020). Inference in artificial intelligence with deep optics and photonics. *Nature*, 588(7836):39–47.

Wilkes, C. M., Qiang, X., Wang, J., Santagati, R., Paesani, S., Zhou, X., Miller, D. A. B., Marshall, G. D., Thompson, M. G., and O'Brien, J. L. (2016). 60dB High-extinction Auto-configured Mach–Zehnder Interferometer. *Optics Letters*, 41(22):5318–5321.

Wilson, J. (2023). The multiply and fourier transform unit: A micro-scale optical processor. [Online]. Available: https://optalysys.com/wp-content/uploads/2023/09/Multiply_and_Fourier_Transform_white_paper_12_12_20.pdf.

Wu, C., Yu, H., Lee, S., Peng, R., Takeuchi, I., and Li, M. (2021). Programmable phase-change metasurfaces on waveguides for multimode photonic convolutional neural network. *Nature communications*, 12(1):96.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation. *CoRR*, abs/2004.09602:.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, B., Zhou, Y., and Chiu, Y. (2016). A 23mw 24gs/s 6b time-interleaved hybrid two-step adc in 28nm cmos. In *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, pages 1–2.

Xu, C., Niu, D., Muralimanohar, N., Balasubramonian, R., Zhang, T., Yu, S., and Xie, Y. (2015). Overcoming the challenges of crossbar resistive memory architectures. In *2015 IEEE 21st international symposium on high performance computer architecture (HPCA)*, pages 476–488. IEEE.

Xu, H. and Shi, Y. (2018). Flat-top cwdm (de)multiplexer based on mzi with bent directional couplers. *IEEE Photonics Technology Letters*, 30(2):169–172.

Xu, X., Tan, M., Corcoran, B., Wu, J., Boes, A., Nguyen, T. G., Chu, S. T., Little, B. E., Hicks, D. G., Morandotti, R., Mitchell, A., and Moss, D. J. (2021). 11 TOPS Photonic Convolutional Accelerator for Optical Neural Networks. *Nature*, 589(7840):44–51.

Yan, T., Wu, J., Zhou, T., Xie, H., Xu, F., Fan, J., Fang, L., Lin, X., and Dai, Q. (2019). Fourier-space diffractive deep neural network. *Physical review letters*, 123(2):023901.

Yang, L.-L. and Hanzo, L. (2001a). Minimum-distance decoding of redundant residue number system codes. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, volume 10, pages 2975–2979 vol.10.

Yang, L.-L. and Hanzo, L. (2001b). Redundant residue number system based error correction codes. In *IEEE 54th Vehicular Technology Conference. VTC Fall 2001. Proceedings (Cat. No. 01CH37211)*, volume 3, pages 1472–1476. IEEE.

Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., Yang, J. J., and Qian, H. (2020). Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646.

Yassine, H. (1992). Hierarchical residue numbering system suitable for vlsi arithmetic architectures. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, volume 2, pages 811–814 vol.2.

Yassine, H. and Moore, W. (1991). Improved mixed-radix conversion for residue number system architectures. *IEE Proceedings G (Circuits, Devices and Systems)*, 138(1):120–124.

Yen, S.-M., Kim, S., Lim, S., and Moon, S.-J. (2003). Rsa speedup with chinese remainder theorem immune against hardware fault cryptanalysis. *IEEE Transactions on computers*, 52(4):461–472.

Yerci, S., Li, R., and Dal Negro, L. (2010). Electroluminescence from er-doped si-rich silicon nitride light emitting diodes. *Applied Physics Letters*, 97(8).

Yong, Z., Chen, H., Luo, X., Govdeli, A., Chua, H., Azadeh, S. S., Stalmashonak, A., Lo, G.-Q., Poon, J. K., and Sacher, W. D. (2022). Power-efficient silicon nitride thermo-optic phase shifters for visible light. *Optics express*, 30(5):7225–7237.

Yu, S., Jiang, H., Huang, S., Peng, X., and Lu, A. (2021a). Compute-in-memory chips for deep learning: Recent trends and prospects. *IEEE circuits and systems magazine*, 21(3):31–56.

Yu, S., Shim, W., Peng, X., and Luo, Y. (2021b). Rram for compute-in-memory: From inference to training. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(7):2753–2765.

Zhang, H., Thompson, J., Gu, M., Jiang, X. D., Cai, H., Liu, P. Y., Shi, Y., Zhang, Y., Karim, M. F., Lo, G. Q., et al. (2021). Efficient on-chip training of optical neural networks using genetic algorithm. *Acs Photonics*, 8(6):1662–1672.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022a). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, S. Q., McDanel, B., and Kung, H. (2022b). Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 846–860. IEEE.

Zhao, P., Xiao, K., Zhang, Y., Bian, K., and Yan, W. (2021). Ameir: Automatic behavior modeling, interaction exploration and mlp investigation in the recommender system. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2104–2110. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. (2016). Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

Zhou, T., Lin, X., Wu, J., Chen, Y., Xie, H., Li, Y., Fan, J., Wu, H., Fang, L., and Dai, Q. (2021). Large-scale neuromorphic optoelectronic computing with a reconfigurable diffractive processing unit. *Nature Photonics*, 15(5):367–373.

Zhou, Z., Yin, B., and Michel, J. (2015). On-chip light sources for silicon photonics. *Light: Science & Applications*, 4(11):e358–e358.

Zhu, H., Gu, J., Wang, H., Jiang, Z., Zhang, Z., Tang, R., Feng, C., Han, S., Chen, R. T., and Pan, D. Z. (2024). Lightening-transformer: A dynamically-operated optically-interconnected photonic transformer accelerator. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 686–703. IEEE.

Zhu, Y., Zhang, G. L., Li, B., Yin, X., Zhuo, C., Gu, H., Ho, T.-Y., and Schlichtmann, U. (2020). Countering variations and thermal effects for accurate optical neural networks. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–7, . .

Zuo, Y., Li, B., Zhao, Y., Jiang, Y., Chen, Y.-C., Chen, P., Jo, G.-B., Liu, J., and Du, S. (2019). All-optical Neural Network with Nonlinear Activation Functions. *Optica*, 6(9):1132–1137.

CURRICULUM VITAE