Mirage: An RNS-Based Photonic Accelerator for DNN Training

Cansu Demirkiran*
Boston University
Boston, MA, USA
cansu@bu.edu

Guowei Yang
Boston University
Boston, MA, USA
guoweiy@bu.edu

Darius Bunandar
Lightmatter
Boston, MA, USA
darius@lightmatter.co

Ajay Joshi
Boston University
Boston, MA, USA
joshi@bu.edu

***Abstract*—Photonic computing is a compelling avenue for performing highly efficient matrix multiplication, a crucial operation in Deep Neural Networks (DNNs). While this method has shown great success in DNN inference, meeting the high precision demands of DNN training proves challenging due to the precision limitations imposed by costly data converters and the analog noise inherent in photonic hardware. This paper proposes *Mirage*, a photonic DNN training accelerator that overcomes the precision challenges in photonic hardware using the Residue Number System (RNS). RNS is a numeral system based on modular arithmetic—allowing us to perform high-precision operations via multiple low-precision modular operations. In this work, we present a novel micro-architecture and dataflow for an RNS-based photonic tensor core performing 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 systolic arrays, *Mirage* achieves more than $23.8\times$ faster training and $32.1\times$ lower EDP in an iso-energy scenario and consumes $42.8\times$ lower power with comparable or better EDP in an iso-area scenario.**

I. Introduction

Photonic computing has shown remarkable promise in accelerating Deep Neural Network (DNN) inference by enabling high throughput and energy-efficient matrix/vector operations [16], [35], [36], [45], [52]–[54], [60], [67]. However, photonic computing—similar to other analog computing methods—suffers from precision issues, making it unsuitable for DNN training.

Photonic DNN accelerators are typically centered around a matrix-vector multiplication (MVM) unit based on Mach Zehnder Interferometers (MZIs) [16], [52], Micro-Ring Resonators (MRRs) [35], [36], [45], [60], or a combination of the two [53], [54]. These units perform a set of dot products in parallel in the analog domain for executing general matrix-matrix multiplication (GEMM) operations. Operations in the MVM unit require switching between digital and analog domains before and after each analog operation. These conversions are performed by using digital-to-analog and analog-to-digital converters (DACs and ADCs). In the analog domain, the values can only be represented as fixed-point (FXP) numbers whose precision is limited by the precision of DACs and ADCs, as the values are encoded in physical properties.

To better understand this limitation, assume an input vector and a weight vector—that are to be multiplied in the analog domain—pass through DACs with b_i-bit and b_w-bit precision, respectively. The DACs convert these values into b_i-bit and b_w-bit signed integers. A dot product between two h-long vectors with b_w-bit and b_i-bit elements then produces a result with $b_{\text{out}}=b_i+b_w+\log_2(h)-1$ bits of information. To capture this output fully, an ADC with bit precision $b_{\text{ADC}} \geq b_{\text{out}}$ is needed. Unfortunately, the energy consumption of ADCs increases *exponentially* with bit precision and can easily make the high energy efficiency promise of photonic computing for DNNs no longer feasible. To avoid this, the common practice is to use ADCs with $b_{\text{ADC}}<b_{\text{out}}$ and lose $b_{\text{out}}-b_{\text{ADC}}$ bits of information on every dot product result [49]. In addition, vector size h, which is determined by the size of the photonic core, is typically smaller than the matrix sizes in large DNN layers. This requires the matrices to be tiled into multiple smaller blocks and the GEMM operation to be executed tile-by-tile. Each tiled-MVM operation produces a partial output to the final GEMM output where the aforementioned information loss is induced on every partial output—causing the errors to accumulate while composing the final GEMM output.

While some simple DNNs are more resilient to noise, typically, as the DNN gets larger and the task becomes more complex, a degradation in accuracy is observed [17], [49]. This degradation can be quite drastic even for DNN inference, especially for higher h. Furthermore, DNN training is often more sensitive to quantization noise than inference, as gradient calculation requires a relatively higher dynamic range. Therefore, the scope of DNN acceleration with photonic hardware has been limited to DNN inference except for a few works [6], [22], [29], [44], [68] that focused on training very small DNN models and simple tasks. Generally speaking, training state-of-the-art DNN models using photonic compute cores has been a far-fetched goal due to the limited precision of analog operations.

In this paper, we present *Mirage*, a precise photonic accelerator for DNN training. *Mirage* leverages the Residue Number System (RNS), a numeral system based on modular arithmetic, to perform high-precision analog operations in the photonic hardware. In RNS, numbers are represented as a set of integer residues for a set of selected co-prime moduli—reducing the bit-width of the operands. Essentially, RNS simplifies high-precision operations into multiple low-precision operations and enables high-precision arithmetic despite using low-precision data converters. As a result, we can take advantage of the high-speed and energy-efficiency opportunities of photonic computing while achieving high DNN accuracy.

GEMM operations in the RNS space require modular dot products. *Mirage* employs a novel micro-architecture for performing modular multiply-accumulate (MAC) operations in the photonic

* Corresponding author

core by leveraging the fact that the optical phase loops around at every 2π—which is effectively a modulo 2π operation. We use phase shifters to perform modular multiplications between two operands that are encoded in the applied voltage and the length of the phase shifters. We cascade multiple modular multipliers to accumulate the multiplication results encoded in the optical phase to perform modular dot products. A set of modular dot product units (MDPUs) construct a modular MVM unit (MMVMU). We deploy a separate MMVMU for each n moduli to perform n modular MVMs in parallel. We leverage the block floating point (BFP) format to obtain a high dynamic range while enabling integer operations in the photonic core using the mantissa bits of BFP values that share an exponent.

To evaluate our design, we first perform a sensitivity analysis to choose the optimal BFP configuration and make micro-architecture decisions that lead to maximum performance in *Mirage*. We then compare the training performance of *Mirage* against systolic arrays that support FP32 and several other more efficient data formats.

Our contributions in this work are as follows:

- We propose an RNS-based computing model and dataflow for DNN training that combines BFP and RNS to achieve high accuracy in analog photonic hardware.
- We introduce a new photonic MAC unit design that efficiently performs modular operations using the optical phase, which enables RNS-based arithmetic using photonic hardware.
- Using the photonic MAC units as a building block, we architect a novel RNS-based photonic DNN training accelerator, *Mirage*. To our knowledge, *Mirage* is the first-ever photonic training accelerator that can successfully train real-world practical DNN models.

Our evaluation shows that *Mirage* can train state-of-the-art DNNs in various tasks and achieve validation accuracy comparable to FP32 training. We show that on average across several DNNs, *Mirage* achieves more than $23.8\times$ faster training and $32.1\times$ lower Energy-Delay Product (EDP) in an iso-energy scenario and consumes $42.8\times$ lower power with comparable EDP in an iso-area scenario when compared to systolic arrays using the best-performing data format.

II. Background

A. DNN Training

DNN training consists of two main steps: forward pass and backward pass. In a DNN, the input X to the $(\ell{+}1)$-th layer during a forward pass is the output generated by the previous ((ℓ)-th) layer:

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

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. 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:

$$\frac{\partial \mathcal{L}}{\partial X^{(\ell)}} = W^{(\ell)^T}\frac{\partial \mathcal{L}}{\partial O^{(\ell)}}, \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial O^{(\ell)}}X^{(\ell)^T}. \tag{3}$$

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:

$$W^{(\ell)}_{i+1} = W^{(\ell)}_{i} - \eta \Delta W^{(\ell)}_{i}, \tag{4}$$

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.

B. Data Formats for DNNs

The efforts to optimize DNN training or inference mainly revolve around improving the efficiency of MAC operations and matrix multiplications. To accelerate MAC operations, scalar quantization and FXP multiplications have been extensively explored for DNN inference and training [7], [14], [24], [28], [30], [70]. In addition, special FP formats have been proposed to improve the hardware performance over FP32 and FP16 formats. Examples include Brain Float (BFLOAT16) [63], HFP8 [59], and TensorFloat [58].

In our work, we use the BFP format, which provides a middle ground between FXP and FP formats. BFP has been used for DNN inference [8], [15], [57] and training [19], [69] as it is less costly than the FP formats and achieves better accuracy than the FXP formats with the same bit-width. BFP format splits tensors into groups and assigns an exponent to each group that is shared by the elements within the group. This representation allows integer operations between groups using only sign and mantissa bits while preserving the dynamic range through a shared exponent.

In the analog domain, the lack of FP arithmetic and the lack of support for large-bit-width FXP numbers limit the range of applications that can benefit from analog computing. Combining BFP with analog hardware is a promising solution as BFP enables low-precision integer operations while providing a wider dynamic range than conventional integer arithmetic. This idea of using BFP formats for analog computing was first proposed by Basumallik et al. [8] for DNN inference. In our work, we combine BFP and RNS to perform DNN training in photonic hardware.

C. Bit Precision in Conventional Analog Cores

Fig. 1(a) shows a diagram for the dataflow in a conventional analog core performing a single MVM operation. Independent of the technology, in an analog MVM core, inputs and weights in a DNN layer are passed through DACs and are encoded in an analog property (e.g., phase, amplitude, etc.). After analog dot products are performed, the output data are passed through ADCs. Here, the precision of the analog operation is determined by (1) the precision of DACs, (2) the precision of ADCs, and (3) the signal-to-noise ratio (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, for 8-bit DACs, the output will require more than 16 bits, calling for an ADC with $b_{\text{ADC}}{\geq}16$ to ensure no information loss. Fig. 1(b) shows the approximate energy consumption per conversion for ADCs and DACs with different bit precision [40]. As seen in the figure, ADC energy consumption is much higher (two orders of magnitude) than DAC energy consumption. In addition, ADC energy consumption increases

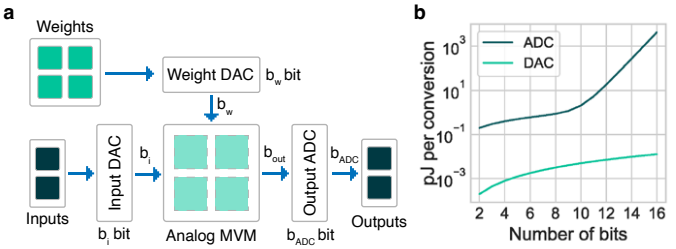

Fig. 1. (a) Dataflow for a conventional analog core. (b) Energy consumption per conversion in ADCs and DACs with varying bit precision. The energy per conversion numbers are estimated using equations formulated by Murmann [40].

exponentially with bit precision—roughly $4\times$ higher energy per conversion for each additional bit. For the abovementioned 8-bit example, a single A-to-D conversion would require $\geq$1 nJ energy. Considering the low energy consumption of the MAC operations performed in the analog domain (tens-to-hundreds of fJ/MAC), high-precision ADCs can easily dominate the total energy consumption. In addition to data converter precision, the SNR required to ensure the integrity of the analog operations also increases exponentially with the increasing bit precision, in turn calling for higher power consumption and posing a limitation to achievable precision in analog cores.

While such high-precision analog operations are impractical, using low precision causes information loss on partial outputs. Prior works [17], [49] show that this information loss due to low-precision ADCs can cause a drastic accuracy loss for inference, even for 8-bit analog operations—which is a widely accepted bit precision for DNN inference in digital hardware. This difference between analog and digital hardware stems from that in digital hardware, the quantization step is typically done after scaling by a static value (e.g., a pre-determined constant) or a dynamic value dependent on the input (e.g., the maximum value in a tensor or a shared exponent as in BFP) that helps minimize the quantization effects and preserve the dynamic range. In analog hardware, while this scaling can be done at the quantization step before processing a layer (before DACs), the output of the analog MAC operations is quantized by the ADCs without any scaling in the analog domain—causing a more significant loss than the pre-layer quantization. Here, a higher h increases the required bit precision to capture the partial outputs (b_{out})—resulting in higher information loss on the partial outputs and in turn, more severe accuracy degradation, especially for large DNNs handling complex tasks. While the accuracy degradation in inference can be recovered for some models through sophisticated training methods such as quantization-aware training [33], [65], the impact of the abovementioned information loss on training accuracy is mostly irreparable.

D. The Residue Number System (RNS)

The RNS represents an integer as a set of smaller integers called residues, which are obtained by performing a modulo operation on the said 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 $i\in\{1,...,n\}$. X can be uniquely reconstructed from the residues and the corresponding moduli using the Chinese Remainder Theorem (CRT):

$$X=\sum_{i=1}^{n}(x_i M_i T_i)_M, \tag{5}$$

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 addition and multiplication. Therefore, we can perform the GEMM operations in DNNs in the RNS space as long as we guarantee the output of the dot products stays 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.

E. Device Metrics and Noise Sources in Silicon Photonics

1) Modulation Mechanisms and Device Tradeoffs: In silicon photonics, efficient reprogrammability is a critical property for devices such as MZIs and MRRs—which are the building blocks of the computing units. Modulation in such devices can be obtained through different mechanisms creating different tradeoffs between device metrics such as modulation bandwidth, optical loss, and device size—which impacts the scalability of the design. These mechanisms can broadly be grouped into three: thermo-optic, free-carrier-dispersion-based, and nano/micro-opto-electro-mechanical systems (N/MOEMS)-based devices. Thermo-optic devices are widely used in silicon photonics due to their simple fabrication process and high modulation efficiency, but their modulation speed is typically limited to a few KHz. In addition, the heaters used for modulation dissipate significant power and can easily cause thermal crosstalk. The free-carrier-dispersion effect can be used to design high-speed modulators. While these modulators can easily reach tens of GHz of bandwidth, they typically are lossy and require longer device lengths. Lastly, N/MOEMS-based devices have recently emerged as a viable alternative to the other two mechanisms. These devices [3], [21], [47] provide a moderate modulation frequency (up to a few hundred MHz) and low optical loss with negligible static power consumption.

2) Sources of Analog Noise: Shot noise and thermal noise are the two main sources of noise in analog photonic cores [23]. Shot noise occurs due to the statistical variation in the number of photons or electrons. It can be approximately represented by a Gaussian distribution as

$$I_S=\mathcal{N}(0,\,2q_e I_{\text{D}}\Delta f), \tag{6}$$

where q_e is the elementary charge, I_D is the photodetector current, and Δf is the bandwidth. Thermal noise results from the resistor in the trans-impedance amplifier (TIA) circuitry, which can be modeled as

$$I_T=\mathcal{N}\left(0,\,\frac{4k_B T}{R}\Delta f\right), \tag{7}$$

where k_B is the Boltzman constant, T is the temperature, and R is the feedback resistor of the TIA. In the presence of noise, to achieve a desired bit precision b, one should be able to reach 2^b separable levels, i.e., SNR $\geq 2^b$. It is a common practice to increase the optical input power to suppress the noise until the SNR is large enough to achieve b bits [16], [53].

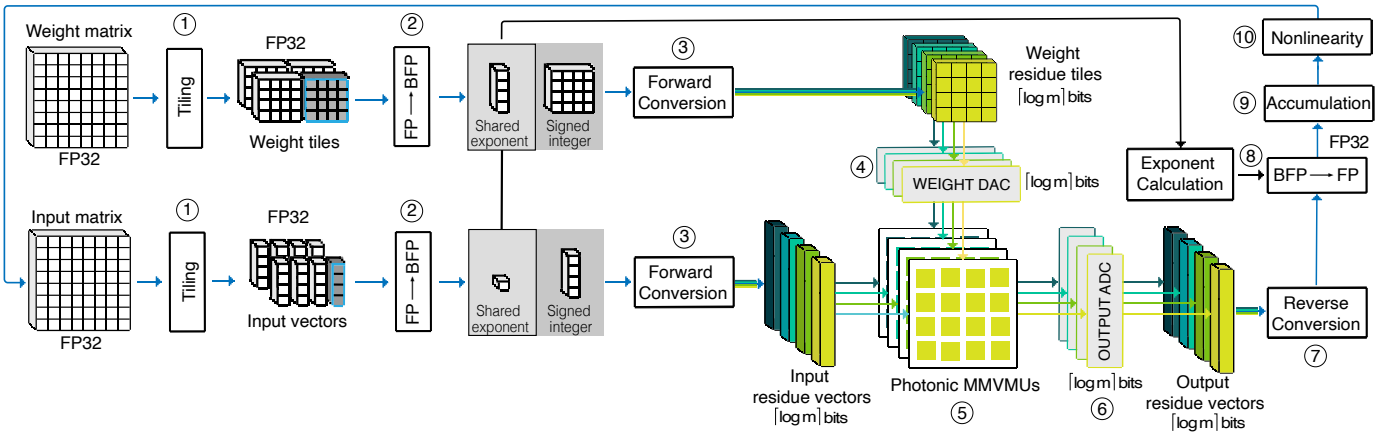

Fig. 2. Mirage's RNS-based dataflow for a single tiled-MVM operation as part of a forward pass. We show a four-moduli case in this figure as an example.

III. RNS-Based Dataflow in *Mirage*

RNS is closed under addition and multiplication allowing a GEMM operation to be performed in the RNS space. Using RNS, Eq. (1) can be rewritten as:

$$\vec{X}^{(\ell+1)} = f^{(\ell)}\left(\text{CRT}\left(\left|\left|W^{(\ell)}\right|_{\mathcal{M}}\left|X^{(\ell)}\right|_{\mathcal{M}}\right|_{\mathcal{M}}\right)\right), \tag{8}$$

where $\mathcal{M}$ represents a selected set of n co-prime moduli. 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 back to a single output matrix in BNS by using Eq (5). The same approach applies to the GEMM operations in the backward pass as stated in Eqs. (2) and (3).

Fig. 2 shows the dataflow for a tiled-GEMM operation as part of the forward pass in *Mirage*. The dataflow follows these steps:

① The FP32 input and FP32 weight matrices (flattened if necessary) are tiled. Tile size is equal to the size of the MMVMUs.

② The FP32 values are converted into BFP. In an MVM operation with BFP values, the input vector and each row of the weight tile represent a group[1]. 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. 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 IV-B for details).

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 IV-A).

⑤ Analog modular MVM operations are performed in the 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 IV-A).

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. (5). 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) and (3). Once the weight gradients are obtained, the weight values are updated according to Eq. (4). Here, we perform all the GEMM operations in BFP,

[1]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.

however, we store the weights in FP32 in the memory and perform the weight updates in FP32.

IV. *Mirage* MICRO-ARCHITECTURE

In this section, we present the micro-architecture of *Mirage* enabling the RNS-based dataflow explained in Section III. We first describe the design of our proposed photonic modular arithmetic units, discuss the moduli selection process, and lastly, explain the accelerator design and how different components in *Mirage* interact.

A. Photonic Modular Arithmetic Units

In *Mirage*, we perform tiled-MVM operations in the RNS domain, as shown in Fig. 2, 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.

1) Modular Multiplication Unit (MMU): In a typical dual-rail Mach-Zehnder Modulator (MZM) (see Fig. 3(a)), the phase difference between input and output is

$$\Delta\Phi = \frac{VL}{V_{\pi\cdot L}}, \tag{9}$$

where $V_{\pi\cdot L}$ is the phase shifter's modulation efficiency, which is 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 using the attenuation caused by the phase shift on the input signal. However, in RNS, a modular multiplication $|xw|_m$ is needed. In *Mirage*, we obtain this behavior via *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 cannot be changed at runtime. Therefore, we use separate phase shifters for each digit of the binary operand, where the length of the shifter is proportional to the weight of the binary digit as shown in Fig. 3(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., $V^{(d)} = (wV_0) \wedge x^{(d)}, \forall d \in \{0, ..., b-1\}$.

Fig. 3(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[2]. 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

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

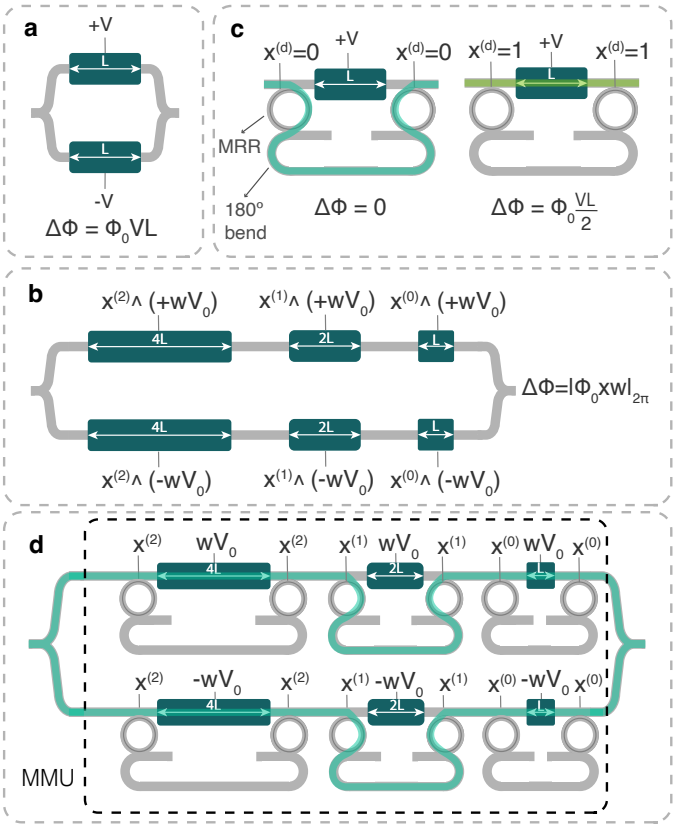

Fig. 3. (a) Simple MZM with phase shifters with length L and applied voltage V. (b) 3-bit modular multiplication using cascaded phase shifters. (c) Routing light using MRR switches. (d) 3-bit modular multiplication using MRR switches.

$0 \wedge V = 0$, 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. 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

$$\Delta\Phi_{\text{total}} = \Big|\Big(\sum_{d=0}^{b-1}(2^d x^{(d)} w \frac{2\pi}{m})\Big)\Big|_{2\pi} = \frac{2\pi}{m}\big|(xw)\big|_m. \tag{10}$$

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.

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

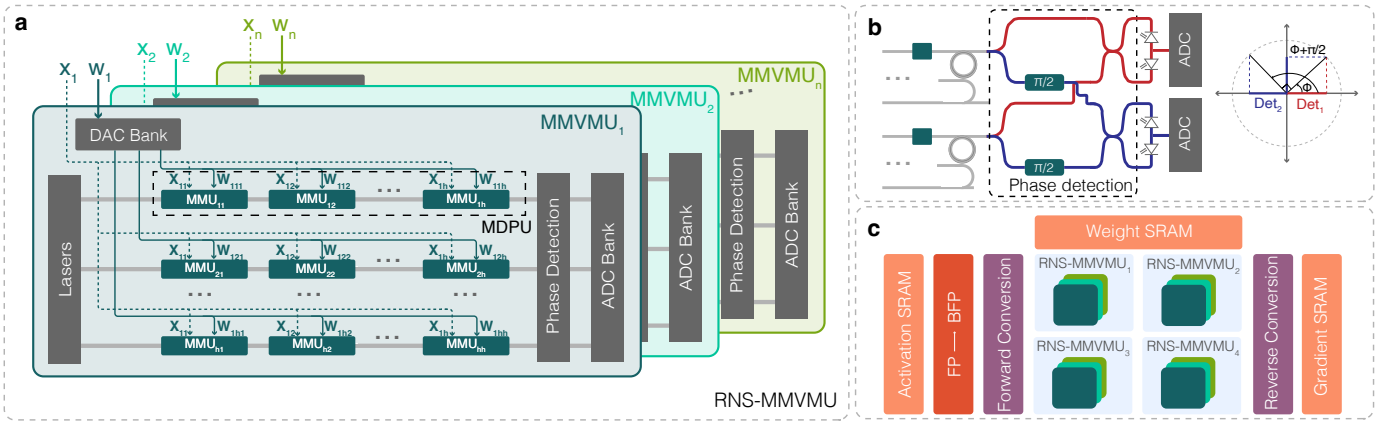

Fig. 4. (a) RNS-based MMVM Unit (RNS-MMVMU) micro-architecture. (b) Phase detection unit. The top arms of the two rows detect the amplitude of the incoming signals directly while the bottom arms apply $\pi/2$ radians phase shift and detect the amplitude. Phase detection is done by using these two amplitude values. (c) Main components of *Mirage* architecture with four RNS-MMVMUs and three moduli as an example.

(V_{bias}) is fully applied. This requires a total phase shifter length of

$$L_{\text{total}} = \frac{V_{\pi L}}{V_{\text{bias}}} \frac{\Delta\Phi_{\max}}{\pi}. \tag{11}$$

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 II-E1.

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 Fig. 3(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 II-E1, 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 Fig. 3(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 Fig. 3(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 (Fig. 3(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 (Fig. 3(c), right). Fig. 3(d) shows the same multiplication unit as Fig. 3(b) with MRR switches for the abovementioned example where $x=101$ and $w=011$.

In the modified design (Fig. 3(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 without 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 [42]. 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.

2) Modular Dot Product Unit (MDPU) and Modular MVM Unit (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

$$\Delta\Phi_{\text{total}} = \Big|\sum_{j=0}^{g-1}\Big(\sum_{i=0}^{b-1}(2^i x_j^{(i)} w_j \frac{2\pi}{m})\Big)\Big|_{2\pi} = \frac{2\pi}{m}\Big|\sum_{j=0}^{g-1}(x_j w_j)\Big|_m. \quad (12)$$

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 Fig. 4(a), the input residue vector x_i and weight residue tile w_i for each modulus $i \in \{1, ..., n\}$ 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 $\lceil \log_2 m \rceil$-bit ADCs. These values are then converted to BNS via the reverse conversion unit.

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° separation [61]. Fig. 4(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).

B. 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

$$\log_2 M \geq b_{\text{out}} = 2(b_m+1)+\log_2(g)-1, \quad (13)$$

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. (13). 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 [64]. 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 significantly. 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 [26]. 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 [26] for implementation details.

C. Mirage Accelerator Design

Fig. 4(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 Fig. 2). 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 (See Section IV-A for details about MMVMUs). 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 [16]. 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.

V. Evaluation Methodology

A. 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.

B. Hardware Performance, Power and Area

1) 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. [3], 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 [42]. ADCs [66] 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 II-E2. 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 [42]. MRRs use electro-optical tuning and have a very small power consumption of 0.3 pW for switching [42]. This power dissipation is $\sim 10^7\times$ smaller than thermo-optic shifters which resolves the thermal crosstalk problem in MRRs [42]. Each 180° bend waveguide has a 5 μm radius and 0.01 dB insertion loss [4]. The laser-to-chip coupler has a 0.2 dB loss [27] and the laser has a 20% efficiency [38]. 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 VI-A). Using the Eq (11) 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.

2) 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 57 fJ/bit [46]. Each 6-bit DAC with 20 GS/s sample rate consumes 136 mW power and takes up 0.072 mm^2 area [32]. Each 6-bit ADC with 24 GS/s sample rate consumes 23 mW power and takes up 0.03 mm^2 area [66]. 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 VI-A). 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 [40]. All three SRAM arrays (activation, weights, and gradients) are generated using the SRAM compiler for TSMC 40 nm technology node [1]. 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 [2] 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 [26].

For comparison, we use systolic arrays that support several data formats including FP32 [10], BFLOAT16 [63], HPF8 [59], INT8, INT12, and FMAC [69]. We chose systolic arrays for their common usage in DNN acceleration and their superior performance over

CPUs and GPUs [11], [31]. 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. [69]. The power and area per MAC unit are collected through synthesis using Cadence Genus and the TSMC 40 nm library.

VI. Evaluation Results

In this section, we first conduct a sensitivity analysis to investigate the impact of different design knobs in *Mirage* to decide the optimal BFP configuration and hardware parameters. We then provide accuracy and performance, power, and area results for *Mirage* and compare *Mirage* against traditional systolic arrays with MAC units based on different data formats. Lastly, for completeness, we compare the performance of *Mirage* running DNN inference against prior photonic and electronic DNN inference accelerators.

A. 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.

1) BFP Parameters: The BFP configuration, i.e., b_m and g choice, significantly impacts the accuracy and hardware performance of *Mirage*. Fig. 5(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. (13), k_{min}=4 when b_m=3. Similarly, k_{min}=5 for b_m=4, and k_{min}=6 for b_m=5. In Fig. 5(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 Fig. 5(b), b_m=4 when g=16 provides the best energy efficiency among all the options that yield high accuracy in Fig. 5(a). Considering these results, in *Mirage*, we choose b_m=4 and g=16 and use these values for the rest of the experiments.

2) RNS-MMVMU 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 the number of optical channels (MDPUs) for higher throughput. Additionally,

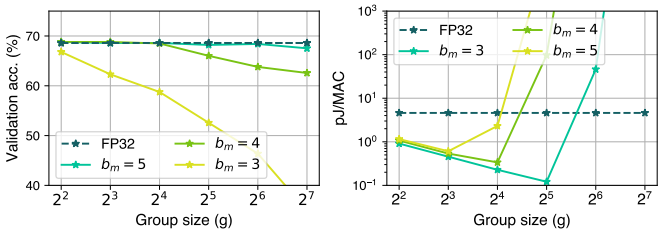

Fig. 5. (a) ResNet18 validation accuracy on Imagenet after training from scratch for 60 epochs and (b) energy per MAC operation (pJ/MAC) for varying b_m and g. This analysis includes energy consumed by lasers and tuning circuitry, TIAs, DACs and ADCs, FP-BFP, and RNS-BNS conversions. Here, ResNet18 is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

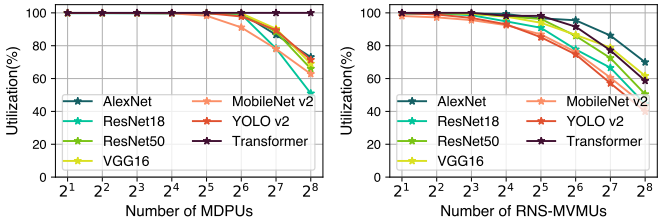

Fig. 6. (a) Number of MDPUs versus spatial utilization (%). (b) Number of RNS-MMVM units versus spatial utilization (%).

we can utilize multiple RNS-MMVMUs on the same chip to further improve parallelism. Fig. 6 (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 Fig. 6 (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.

3) Dataflow Choice: Dataflow choice has been shown to have a critical impact on the performance of DNN hardware accelerators [11]. For DNN inference, the typical dataflows can be listed as weight stationary, input stationary, and output stationary. The performance 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 calcula-

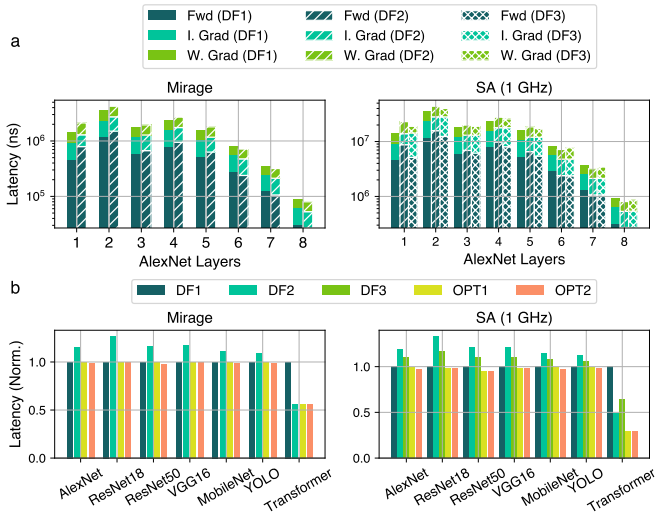

Fig. 7. (a) Latency per step for each layer of AlexNet for *Mirage* (left) and a 1 GHz digital systolic array (right). (b) Latency per step for different DNNs and impact of dataflow for *Mirage* (left) and a 1 GHz digital systolic array (right). The numbers for all dataflows are normalized to the DF1 results for all models.

tion, 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.

Fig. 7(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 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 IV-A, 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 Fig. 7(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. Fig. 7(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 Fig. 7(b), for *Mirage*, we observe that DF1 performs better

TABLE I
VALIDATION ACCURACY OF MIRAGE AND VARIOUS DATA FORMATS [69].

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 II
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

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.

B. Accuracy Evaluation

We evaluated *Mirage*'s accuracy in commonly deployed CNNs for image classification on the ImageNet dataset [18], in YOLO-v2 [48] for object detection on the PASCAL VOC2012 dataset [20], and in a transformer [62] model for machine translation on the IWSLT14 German-English dataset [9]. 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 I shows the accuracy results for *Mirage* and several other data formats. The accuracy results for the data formats other than *Mirage* in Table I were obtained from the work by Zhang et al. [69]. 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.

C. Performance, Power, and Area Evaluation

Table II compares the energy consumption and area per MAC operation and clock rate for *Mirage*'s RNS-MMVMUs against

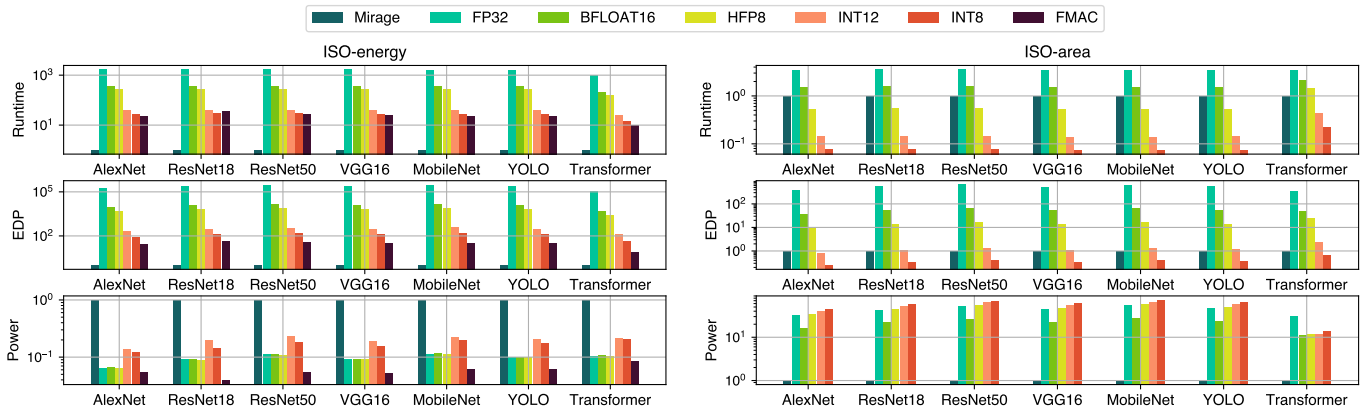

Fig. 8. Normalized training runtime, EDP and power comparison of *Mirage* (eight 16×32 arrays) against systolic arrays using MAC units with various data formats. The plots on the left-hand side show the iso-energy results where the number of MAC units in the systolic arrays is scaled to consume the same energy per MAC operations using the numbers in Table II. 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.

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 MAC ($2-59.1\times$) compared to all data formats besides FMAC [69] ($\sim 2\times$ 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.

Fig. 8 compares the hardware performance of *Mirage* against systolic arrays with MAC units using various data formats when training various DNNs. In Fig. 8, 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 VI-A for the justification of the design choices). For the iso-energy analysis, we scaled the number 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 II. 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 VI-B).

In the iso-energy analysis, the best-performing data format among the systolic array designs is FMAC [69]. Given the same energy per MAC budget, on average across the reported DNNs, *Mirage* achieves a $23.8\times$ lesser runtime and $32.1\times$ lower EDP than the systolic array with FMAC units. However, in this case, *Mirage* consumes $17.2\times$ higher power consumption. Compared to the systolic array with FP32 MAC units, on average, *Mirage* provides $3.5\times$ lesser runtime and $521.7\times$ lower EDP while consuming $42.8\times$ less power.

The iso-area results show that the most efficient datatype that achieves high accuracy, INT12, achieves $5.4\times$ 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 $42.8\times$ lower power consumption and $1.27\times$ lower EDP compared to INT12. *Mirage* has $3.5\times$ lesser runtime, $521.7\times$ lower EDP and $42.8\times$ 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.

Fig. 9 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. It is also 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

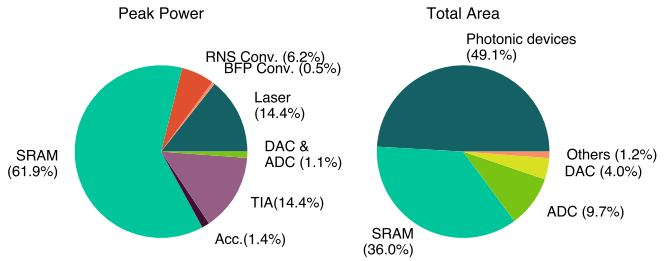

Fig. 9. Peak power consumption and area breakdown for Mirage. The total peak power consumption is 19.95 W and the total area is 476.6mm^2.

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.

Fig. 9 (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).

D. Mirage As An Inference Accelerator

The focus of this paper is DNN training. However, *Mirage* can also be used to accelerate DNN inference as inference operations are a subset of training operations. 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 III. *Mirage* achieves a better throughput in terms of inferences per second (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 ($3.37\times$ slower) and TPU v3 ($3.12\times$ slower). *Mirage* provides a better power efficiency (IPS per Watt) than all photonic (by $2.1-15.4\times$) and electronic (by $1.74-84.7\times$) accelerators except for ADEPT ($2.48\times$ lower). *Mirage* is more area-efficient (IPS per 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 ($1.16\times$ and $8.32\times$ lower, respectively). It should be noted that these photonic inference accelerators in Table III provide a much lower dynamic range than *Mirage* ($\sim$8 bit vs. $\sim$15 bit in *Mirage*). Even for DNN inference, these works can typically can achieve high accuracy only through quantization-aware training [16]. 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.

E. Managing Noise and Process Variations in Mirage

In photonic cores, analog noise, process variations, and fabrication errors can cause noise and errors in the residues. While our accuracy experiments only consider errors due to tiling and quantization, we take shot noise, thermal noise, and all the optical losses along the path into consideration during our PPA analysis and back calculate the required laser power to achieve the desired bit precision (See Section V for details).

In addition to these noise sources, process variations can cause bias in phase shifters and drifts in the resonant wavelength in MRRs

TABLE III
MIRAGE VS 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 [53]	N/A	N/A	N/A	7,692	344.17	61.46
DNNARA [45]	9,345	100	42.05	N/A	N/A	N/A
HolyLight [35]	N/A	N/A	N/A	50,000	900	2,226.11
Eyeriss [11]	N/A	N/A	N/A	35	124.80	2.85
Eyeriss v2 [12]	N/A	N/A	N/A	102	174.80	N/A
TPU v3 [31]	32,716	18.18	18.00	N/A	N/A	N/A
UNPU [34]	N/A	N/A	N/A	346	1,097.50	21.62
Res-DNN [51]	N/A	N/A	N/A	386.11	427.78	N/A

resulting in errors during operations. Prior works proposed various methods such as careful parameter choices during fabrication [37], [60], novel device modifications [55], and error correction methods [5], [25] for MRR- and MZI-based designs to minimize or calibrate away these errors. These methods can be leveraged in *Mirage* similar to other photonic hardware. Moreover, previous works show that increasing DAC precision helps encode values more precisely and reduces the encoding errors in the photonic devices to achieve the desired bit precision [16]. The output precision (b_{out}) in an MDPU is limited by how well one can encode input values onto phase shifters and MRRs. The total error at the output of an MDPU can be calculated by considering all errors accumulated along the optical path as the signal passes through photonic devices. There are h MMUs in an MDPU. Each MMU includes a group of phase shifters representing a $\lceil \log_2 m \rceil$-bit value and $2\lceil \log_2 m \rceil$ MRRs controlling the route of the signal. The precision at the output of an h-long MDPU can then be quantified by adding the errors in quadrature as

$$\Delta\Phi_{\text{out}} = \sqrt{h\Delta\varepsilon_{\text{PS}}^2 + 2h\lceil \log_2 m \rceil \Delta\varepsilon_{\text{MRR}}^2}\,, \tag{14}$$

where $\Delta\varepsilon_{\text{PS}}$ is the encoding error per phase shifter (phase shifters within a single MMU are considered together) and $\Delta\varepsilon_{\text{MRR}}$ is the encoding error per MRR. $\Delta\Phi_{\text{out}}$ is calculated for the worst-case scenario where the light goes through all the phase shifters. It should be noted that $\Delta\varepsilon_{\text{PS}}$ and $\Delta\varepsilon_{\text{MRR}}$ quantities should be measured in the fabricated silicon photonic wafer to precisely estimate the compound noise and the error margins during analog operations. However, for this study, we can conservatively assume $\Delta\varepsilon_{\text{PS}} \leq 2^{-b_{\text{DAC}}}$ and $\Delta\varepsilon_{\text{MRR}} \leq 0.3\%$ [42]. To achieve a b_{out}-bit output precision, $\Delta\Phi_{\text{out}} \leq 2^{-b_{\text{out}}}$ should be guaranteed. Our calculations show that $b_{\text{DAC}} \geq 8$ satisfies this inequality for $b_{\text{out}} \geq \log_2 m$ when $h = 16$, which is adequate for achieving high accuracy in *Mirage*. Given that DACs and ADCs together consume only 1.1% of the overall power and DACs are used only when a new tile is loaded to the phase shifters, slightly increasing DAC precision (from 6-bit to 8-bit) does not cause a significant change in the overall power consumption. With 8-bit DACs [41], energy and average power consumption of *Mirage* increases only by $1.09\times$ on average among the evaluated DNNs compared to *Mirage* with 6-bit DACs.

Lastly, redundant RNS (RRNS) can be used for error detection and correction in RNS-based systems. Demirkiran et al. [17] show that by adding redundant moduli to the original set, we can

recover from accuracy loss during RNS-based DNN inference in the presence of noise. In RRNS, the operations are performed for all moduli as regular RNS. The errors can then be detected and corrected through majority logic decoding. Adding redundant moduli to the set increases the power and area roughly linearly with the number of moduli as the number of components scales linearly with the number of moduli, while throughput stays the same.

VII. Related Work

Over the years, many photonic DNN accelerators have been proposed, which almost exclusively target DNN inference. Some of these inference accelerators are based on MRR weight banks [36], MZI arrays [16], [52], and a mixture of MZIs and MRRs [53], [54]. DNNARA [45], similar to *Mirage*, uses RNS for performing multiplication and addition operations. Unlike *Mirage*, DNNARA does not use any analog property to encode information, instead it uses photonics only to map input operands to the outputs via a one-hot encoded mapping built by using 2×2 switches. This network of switches change the route of the light uniquely to the combination of two operands (i.e., the activated input port and the states of the switches). However, each multiply and add operation requires a separate network, resulting in $O(m\log m)$ switches *per operation* for a modulus m. Although parallelism can be imposed using wavelength division multiplexing (WDM), the number of devices increases rapidly for larger moduli and as more operations are performed in parallel. In *Mirage*, each MAC operation requires fewer optical devices ($O(\log m)$), providing a more scalable approach.

Res-DNN [51] and RNSnet [50] also proposed using RNS to accelerate DNN computation in digital accelerators. While these accelerators reported promising results compared to other electronic accelerators, unlike *Mirage*, they are still bound by the speed of electronic operations which cannot match the high bandwidth of photonics. In addition, these RNS-based works propose staying in the RNS domain throughout the whole inference. While this idea is promising for reducing the RNS-BNS conversion overhead, it has several drawbacks. First, it requires performing periodic scaling operations in the RNS domain to stay in the RNS range. Second, nonlinear operations cannot be performed using RNS. This necessitates using approximations (e.g., Taylor series expansion) for nonlinear operations—which can lead to accuracy loss, especially in training. Lastly, staying in the RNS domain for the whole DNN computation limits us to use only integer arithmetic. In *Mirage*, using BFP enables us to preserve the dynamic range during operations and significantly improves the success of DNN training in analog hardware. Even for inference, these works [50], [51] use significantly higher input/weight precision than *Mirage* to achieve high accuracy ($\geq$16-bit vs. 5-bit in *Mirage*). Considering that RNS-BNS conversions only consume 6% of the power in *Mirage*, we believe that the hybrid arithmetic (RNS and FP) approach we propose is a better fit for DNN training.

Similar to photonics, ReRAM-based processing-in-memory (PIM) designs also suffer from precision limitations. To overcome this issue, some prior works used multiple low-bit cells to compose higher-bit results. For example, PRIME [13] uses two 3-bit cells to achieve 6-bit precision. Another example, PipeLayer [56], 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 $14.4\times$ more power-efficient (OPs/s/W) while being $8.8\times$ less area efficient (OPs/s/mm^2).

Lastly, a few previous works demonstrated fully optical [6] and hybrid in-situ DNN training [29], [44]. Other works combined photonics with alternative training schemes such as direct feedback alignment [22] and genetic algorithm [68]. While these works are promising in terms of showing the applicability of the photonic technology in DNN training, the demonstrations have been limited to DNNs with only a few layers, small datasets, and simple classification problems so far. To train more complex state-of-the-art DNNs, higher precision will be needed—encouraging us to look for innovative accelerator designs. In *Mirage*, by utilizing BFP and RNS, we propose a way of extending the applicability of photonic training accelerators to today's commonly used DNNs.

VIII. Discussion

Over the years, researchers have developed many ADC designs including traditional $\Delta\Sigma$ 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 techniques such as pipelining and time-interleaving 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 [39]. This indicates that there can be 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.

While *Mirage* uses photonic technology, the idea of using RNS applies to other analog technologies that suffer from precision restrictions, such as ReRAM, PCM, etc. [17]. For such accelerators, GEMM operations can be performed as is and analog modulo operations can be implemented through electrical analog circuits such as ring oscillators [43].

IX. Conclusion

In this work, we proposed *Mirage*, an RNS-based photonic accelerator for DNN training. By combining RNS and BFP, *Mirage* can successfully train state-of-the-art DNNs at least $23.8\times$ faster and with $32.1\times$ lower EDP in an iso-energy scenario and with $42.8\times$ lower power consumption in an iso-area scenario than systolic arrays. Overall, we believe that combining analog computing with RNS is a promising solution to overcome the precision challenges in photonic DNN accelerators.

ACKNOWLEDGEMENTS

This work was in part supported by the IARPA MicroE4AI program and by Lightmatter through an internship.

References

[1] "40nm technology." [Online]. Available: https://www.tsmc.com/english/dedicatedFoundry/technology/logic

[2] , "Genus Synthesis Solution," . [Online]. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html

[3] R. Baghdadi, M. Gould, S. Gupta, M. Tymchenko, D. Bunandar, C. Ramey, and N. C. Harris, "Dual slot-mode noem phase shifter," *Opt. Express*, vol. 29, no. 12, pp. 19 113–19 119, Jun 2021. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-29-12-19113

[4] M. Bahadori, M. Nikdast, Q. Cheng, and K. Bergman, "Universal design of waveguide bends in silicon-on-insulator photonics platform," *Journal of Lightwave Technology*, vol. 37, no. 13, pp. 3044–3054, 2019.

[5] S. Bandyopadhyay, R. Hamerly, and D. Englund, "Hardware error correction for programmable photonics," *Optica*, vol. 8, no. 10, pp. 1247–1255, Oct 2021. [Online]. Available: https://opg.optica.org/optica/abstract.cfm?URI=optica-8-10-1247

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

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

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

[9] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand *et al.*, "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 12–58.

[10] Bsg-External, "Bsg-external/hardfloat." [Online]. Available: https://github.com/bsg-external/HardFloat

[11] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[12] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

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

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

[15] B. Darvish Rouhani, D. Lo, R. Zhao, M. Liu, J. Fowers, K. Ovtcharov, A. Vinogradsky, S. Massengill, L. Yang, R. Bittner *et al.*, "Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point," *Advances in neural information processing systems*, vol. 33, pp. 10 271–10 281, 2020.

[16] C. Demirkiran, F. Eris, G. Wang, J. Elmhurst, N. Moore, N. C. Harris, A. Basumallik, V. J. Reddi, A. Joshi, and D. Bunandar, "An electro-photonic system for accelerating deep neural networks," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 19, no. 4, pp. 1–31, 2023.

[17] C. Demirkiran, L. Nair, D. Bunandar, and A. Joshi, "A blueprint for precise and fault-tolerant analog neural networks," *arXiv preprint arXiv:2309.10759*, 2023.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[19] M. Drumond, T. Lin, M. Jaggi, and B. Falsafi, "Training dnns with hybrid block floating point," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

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

[21] Y. Feng, D. J. Thomson, G. Z. Mashanovich, and J. Yan, "Performance analysis of a silicon noems device applied as an optical modulator based on a slot waveguide," *Opt. Express*, vol. 28, no. 25, pp. 38 206–38 222, Dec 2020. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-28-25-38206

[22] M. J. Filipovich, Z. Guo, M. Al-Qadasi, B. A. Marquez, H. D. Morison, V. J. Sorger, P. R. Prucnal, S. Shekhar, and B. J. Shastri, "Silicon photonic architecture for training deep neural networks with direct feedback alignment," *Optica*, vol. 9, no. 12, pp. 1323–1332, Dec 2022. [Online]. Available: https://opg.optica.org/optica/abstract.cfm?URI=optica-9-12-1323

[23] S. Garg, J. Lou, A. Jain, Z. Guo, B. J. Shastri, and M. Nahmias, "Dynamic precision analog computing for neural networks," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 29, no. 2: Optical Computing, pp. 1–12, 2023.

[24] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*. PMLR, 2015, pp. 1737–1746.

[25] R. Hamerly, S. Bandyopadhyay, and D. Englund, "Stability of self-configuring large multiport interferometers," *Phys. Rev. Appl.*, vol. 18, p. 024018, Aug 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.18.024018

[26] A. Hiasat, "A residue-to-binary converter with an adjustable structure for an extended rns three-moduli set," *Journal of Circuits, Systems and Computers*, vol. 28, no. 08, p. 1950126, 2019.

[27] R. Hu, L. Sun, Z. Zhang, Q. Sun, Y. Pan, and Y. Su, "Ultrabroadband and compact $2\times$ 2 3-db coupler based on trapezoidal subwavelength gratings," *Optics Express*, vol. 31, no. 14, pp. 23 542–23 550, 2023.

[28] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[29] T. W. Hughes, M. Minkov, Y. Shi, and S. Fan, "Training of photonic neural networks through in situ backpropagation and gradient measurement," *Optica*, vol. 5, no. 7, pp. 864–871, 2018.

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

[31] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, "A domain-specific supercomputer for training deep neural networks," *Commun. ACM*, vol. 63, no. 7, p. 67–78, jun 2020. [Online]. Available: https://doi.org/10.1145/3360307

[32] S.-N. Kim, W.-C. Kim, M.-J. Seo, and S.-T. Ryu, "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*, vol. 65, no. 9, pp. 1154–1158, 2018.

[33] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *CoRR*, vol. abs/1806.08342, p. , 2018. [Online]. Available: http://arxiv.org/abs/1806.08342

[34] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, 2019.

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

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

[37] A. Mirza, F. Sunny, P. Walsh, K. Hassan, S. Pasricha, and M. Nikdast, "Silicon photonic microring resonators: A comprehensive design-space exploration and optimization under fabrication-process variations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 10, pp. 3359–3372, 2022.

[38] G. Mourou, B. Brocklesby, T. Tajima, and J. Limpert, "The future is fibre accelerators," *Nature Photonics*, vol. 7, no. 4, pp. 258–261, 2013.

[39] B. Murmann, "Introduction to adcs/dacs: metrics, topologies, trade space, and applications," *ISSCC Short Course*, 2022.

[40] ——, "Mixed-signal computing for deep neural network inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 3–13, 2021.

[41] A. Nazemi, K. Hu, B. Catli, D. Cui, U. Singh, T. He, Z. Huang, B. Zhang, A. Momtaz, and J. Cao, "3.4 a 36gb/s pam4 transmitter using an 8b 18gs/s dac in 28nm cmos," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3.

[42] S. Ohno, Q. Li, N. Sekine, H. Tang, S. Monfray, F. Boeuf, K. Toprasertpong, S. Takagi, and M. Takenaka, "Si microring resonator optical switch based on optical phase shifter with ultrathin-inp/si hybrid metal-oxide-semiconductor capacitor," *Opt. Express*, vol. 29, no. 12, pp. 18 502–18 511, Jun 2021. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-29-12-18502

[43] O. Ordentlich, G. Tabak, P. K. Hanumolu, A. C. Singer, and G. W. Wornell, "A modulo-based architecture for analog-to-digital conversion," *IEEE journal of selected topics in signal processing*, vol. 12, no. 5, pp. 825–840, 2018.

[44] S. Pai, Z. Sun, T. W. Hughes, T. Park, B. Bartlett, I. A. Williamson, M. Minkov, M. Milanizadeh, N. Abebe, F. Morichetti *et al.*, "Experimentally realized in

situ backpropagation for deep learning in photonic neural networks," *Science*, vol. 380, no. 6643, pp. 398–404, 2023.

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

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

[47] C. Ramey, "Silicon photonics for artificial intelligence acceleration : Hotchips 32," in *2020 IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–26.

[48] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[49] A. S. Rekhi, B. Zimmer, N. Nedovic, N. Liu, R. Venkatesan, M. Wang, B. Khailany, W. J. Dally, and C. T. Gray, "Analog/mixed-signal hardware error modeling for deep learning inference," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

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

[51] N. Samimi, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Res-dnn: A residue number system-based dnn accelerator unit," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 2, pp. 658–671, 2020.

[52] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature photonics*, vol. 11, no. 7, pp. 441–446, 2017.

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

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

[55] L. Song, T. Chen, W. Liu, H. Liu, Y. Peng, Z. Yu, H. Li, Y. Shi, and D. Dai, "Toward calibration-free mach–zehnder switches for next-generation silicon photonics," *Photon. Res.*, vol. 10, no. 3, pp. 793–801, Mar 2022. [Online]. Available: https://opg.optica.org/prj/abstract.cfm?URI=prj-10-3-793

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

[57] Z. Song, Z. Liu, and D. Wang, "Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

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

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

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

[61] M. G. Taylor, "Phase estimation methods for optical coherent detection using digital signal processing," *Journal of Lightwave Technology*, vol. 27, pp. 901–914, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:11058269

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

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

[64] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder based residue to binary number converters for (2/sup n/-1, 2/sup n/, 2/sup n/+1)," *IEEE Transactions on Signal Processing*, vol. 50, no. 7, pp. 1772–1779, 2002.

[65] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *CoRR*, vol. abs/2004.09602, p. , 2020. [Online]. Available: https://arxiv.org/abs/2004.09602

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

[67] X. Xu, M. Tan, B. Corcoran, J. Wu, A. Boes, T. G. Nguyen, S. T. Chu, B. E. Little, D. G. Hicks, R. Morandotti *et al.*, "11 tops photonic convolutional accelerator for optical neural networks," *Nature*, vol. 589, no. 7840, pp. 44–51, 2021.

[68] H. Zhang, J. Thompson, M. Gu, X. D. Jiang, H. Cai, P. Y. Liu, Y. Shi, Y. Zhang, M. F. Karim, G. Q. Lo *et al.*, "Efficient on-chip training of optical neural networks using genetic algorithm," *Acs Photonics*, vol. 8, no. 6, pp. 1662–1672, 2021.

[69] S. Q. Zhang, B. McDanel, and H. Kung, "Fast: Dnn training under variable precision block floating point with stochastic rounding," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 846–860.

[70] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.