TOWARD ENERGY-EFFICIENT MACHINE LEARNING: ALGORITHMS AND ANALOG COMPUTE-IN-MEMORY HARDWARE

by

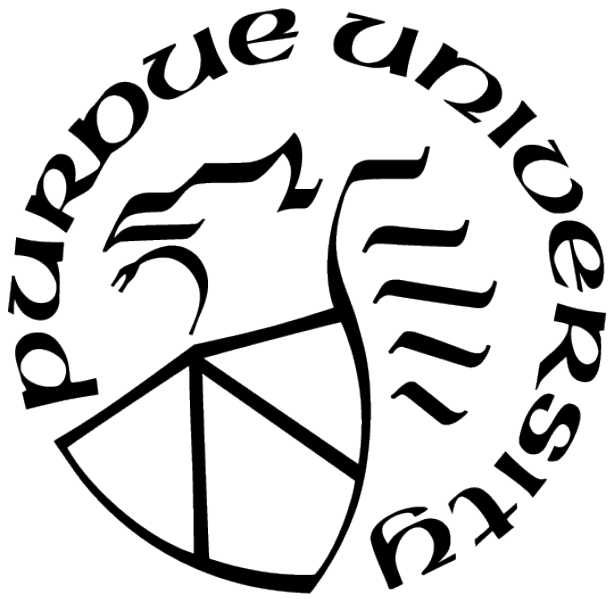
Indranil Chakraborty

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Electrical and Computer Engineering

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Kaushik Roy, Chair

School of Electrical and Computer Engineering

Dr. Anand Raghunathan

School of Electrical and Computer Engineering

Dr. Vijay Raghunathan

School of Electrical and Computer Engineering

Dr. Shreyas Sen

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Dedicated to my mother, my late father, my sister,
my brother-in-law, my niece and Deboleena

ACKNOWLEDGMENTS

I would like to express my gratitude towards my advisor Prof. Kaushik Roy for helping me shape my understanding of research and guiding me through my journey at the Nanoelectronics Research Laboratory (NRL) thus far. Kaushik has been inspirational in my development as a researcher, always keeping me in touch with the higher level aspects of my research. A remarkable quality in Kaushik is his dedication and commitment towards his students, which I have truly reaped benefits of, during my course here by engaging in regular discussions, not only about the technical aspects of research but also its philosophical values. Overall, I will forever be indebted to Kaushik for his immense support and constructive criticism which has brought me where I am today.

I would also like to thank my doctoral thesis committee members: Prof. Anand Raghunathan, Prof. Vijay Raghunathan and Prof. Shreyas Sen. I would like to take this opportunity to individually express my gratitude towards my committee members. Aside from the countless meaningful interactions I have had with Prof. Anand, I have had the opportunity to collaborate with Prof. Anand on two review papers where I have learnt immensely about framing my research to the community. Prof. Vijay has been very welcoming for any discussion on relatable topic. Prof. Shreyas has also taught me a lot about analyzing any situation better. Finally, I would like to thank my Master's advisor, Prof. Udayan Ganguly at Indian Institute of Technology, Bombay without whom I probably wouldn't have never been introduced to research or thought about doing a PhD.

I would also like to thank the NRL members for creating an environment conducive for research as well as fun activities. I have had the opportunity to know some of the best minds who would surely make a difference in our future. I reserve a special gratitude for my friend, now Prof. Abhronil Sengupta at Penn State University, for support since the day I started here. I would like to also thank Dr. Akhilesh Jaiswal for helping me start my research here. I have been grateful to have friends in NRL such as Mustafa, Nitin, Aayush, Amogh during my PhD. Finally, a big thanks to all my collaborators and fellow lab-mates at NRL for their immense support.

Last, but not the least, I would like to dedicate my doctoral journey to a few really special people in my life: My better half, Deboleena, who has been there through all the ups and downs in my life in the last three and a half years and helped me cope with the pressure and stress, my mother who has always been supportive of my decision to do a PhD despite her having to cope alone in the house, my late father who still inspires me to be a better person, my sister for her constant encouragement, my brother-in-law and my niece for making my life a happier place.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The 'Internet of Things' has increased the demand for artificial intelligence (AI)-based edge computing in applications ranging from healthcare monitoring systems to autonomous vehicles. However, the growing complexity of machine learning workloads requires rethinking to make AI amenable to resource constrained environments such as edge devices. To that effect, the entire stack of machine learning, from algorithms to hardware primitives, have been explored to enable energy-efficient intelligence at the edge.

From the algorithmic aspect, model compression techniques such as quantization are powerful tools to address the growing computational cost of ML workloads. However, quantization, particularly, can result in substantial loss of performance for complex image classification tasks. To address this, a principal component analysis (PCA)-driven methodology to identify the important layers of a binary network, and design mixed-precision networks. The proposed Hybrid-Net achieves a significant improvement in classification accuracy over binary networks such as XNOR-Net for ResNet and VGG architectures on CIFAR-100 and ImageNet datasets, while still achieving up remarkable energy-efficiency.

Having explored compressed neural networks, there is a need to investigate suitable computing systems to further the energy efficiency. Memristive crossbars have been extensively explored as an alternative to traditional CMOS based systems for deep learning accelerators due to their high on-chip storage density and efficient Matrix Vector Multiplication (MVM) compared to digital CMOS. However, the analog nature of computing poses significant issues due to various non-idealities such as: parasitic resistances, non-linear I-V characteristics of the memristor device etc. To address this, a simplified equation-based modelling of the non-ideal behavior of crossbars is performed and correspondingly, a modified technology aware training algorithm is proposed. Building on the drawbacks of equation-based modeling, a Generalized Approach to Emulating Non-Ideality in Memristive Crossbars using Neural Networks (GENIEx) is proposed where a neural network is trained on HSPICE simulation data to learn the transfer characteristics of the non-ideal crossbar. Next, a functional simulator was developed which includes key architectural facets such as tiling, and bit-slicing to analyze the impact of non-idealities on the classification accuracy of large-scale neural networks.

To truly realize the benefits of hardware primitives and the algorithms on top of the stack, it is necessary to build efficient devices that mimic the behavior of the fundamental units of a neural network, namely, neurons and synapses. However, efforts have largely been invested in implementations in the electrical domain with potential limitations of switching speed, functional errors due to analog computing, etc. As an alternative, a purely photonic operation of an Integrate-and-Fire Spiking neuron is proposed, based on the phase change dynamics of Ge2Sb2Te5 (GST) embedded on top of a microring resonator, which alleviates the energy constraints of PCMs in electrical domain. Further, the inherent parallelism of wavelength-division multiplexing (WDM) was leveraged to propose a photonic dot-product engine. The proposed computing platform was used to emulate a SNN inferencing engine for image-classification tasks. These explorations at different levels of the stack can enable energy-efficient machine learning for edge intelligence.

Having explored various domains to design efficient DNN models and studying various hardware primitives based on emerging technologies, we focus on Silicon implementation of compute-in-memory (CIM) primitives for machine learning acceleration based on the more available CMOS technology. CIM primitives enable efficient matrix-vector multiplications (MVM) through parallelized multiply-and-accumulate operations inside the memory array itself. As CIM primitives deploy bit-serial computing, the computations are exposed bit-level sparsity of inputs and weights in a ML model. To that effect, we present an energy-efficient sparsity-aware reconfigurable-precision compute-in-memory (CIM) 8T-SRAM macro for machine learning (ML) applications. Standard 8T-SRAM arrays are re-purposed to enable MAC operations using selective current flow through the read-port transistors. The proposed macro dynamically leverages workload sparsity by reconfiguring the output precision in the peripheral circuitry without degrading application accuracy. Specifically, we propose a new energy-efficient reconfigurable-precision SAR ADC design with the ability to form (n+m)-bit precision using n-bit and m-bit ADCs. Additionally, the transimpedance amplifier (TIA) –required to convert the summed current into voltage before conversion—is reconfigured based on sparsity to improve sense margin at lower output precision. The proposed macro, fabricated in 65 nm technology, provides 35.5-127.2 TOPS/W as the ADC precision varies from 6-bit to 2-bit, respectively. Building on top of the fabricated macro, we next design

a hierarchical CIM core micro-architecture that addresses the existing CIM scaling challenges. The proposed CIM core micro-architecture consists of 32 proposed sparsity-aware CIM macros. The 32 macros are divided into 4 matrix-vector multiplication units (MVMUs) consisting of 8 macros each. The core has three unique features: i) it can adaptively reconfigure ADC precision to achieve energy-efficiency and lower latency based on input and weight sparsity, determined by a sparsity controller, ii) it deploys row-gating feature to maintain SNR requirements for accurate DNN computations, and iii) hardware support for load balancing to balance latency mismatches occurring due to different ADC precisions in different compute units. Besides the CIM macros, the core micro-architecture consists of input, weight, and output memories, along with instruction memory and control circuits. The instruction set architecture allows for flexible dataflows and mapping in the proposed core micro-architecture. The sparsity-aware processing core is scheduled to be taped out next month. The proposed CIM demonstrations complemented by our previous analysis on analog CIM systems progressed our understanding of this emerging paradigm in pertinence to ML acceleration.

1. INTRODUCTION

The idea of building intelligent machines was conceptualized long before the recent excitement about Artificial Intelligence that has engulfed today's computing world. Alan Turing, in his seminal paper [1], "Computing Machinery and Intelligence" was one of the first to lay the foundation of machine intelligence. Since then, the developments in neuroscience [2] as well as emergence of digital computers have enabled the Artifical Neural Networks (ANNs), which have now become the basic building block of AI systems. Despite facing enormous hurdles in realizing large scale ANNs due to their computational complexity, the development of General Purpose Graphics Processing Units (GPGPUs) and Special Purpose Machine Learning (ML) accelerators such as TPUs [3] in the last decade has made Deep Learning (DL) pervasive in various applications such as speech recognition, predictive systems and image and video classification [4]–[7].

The motivation for energy-efficient AI systems arise from its predicted evolution in the coming years. Today, AI is comprised of various edge devices collecting data from the environment and sending it over to a cloud system for processing the data. However, with rapid proliferation of these edge devices, it becomes costly to sustain such a cloud computing model. To that effect, it is of growing interest to enable computing in these edge devices. Edge computing can overcome the latency of communication between edge and cloud as well as enhance data security.

Thus far, we have established that AI at the edge is an absolute necessity. However, the growing computational complexity of DL models poses a serious challenge toward their deployment in resource constrained edge devices. Fig. 1.1 shows the exponential trends of computational complexity in ML applications over the last few years [8]. There is a need, therefore, to devise solutions at various levels of the stack which would include designing algorithms that enable low complexity neural network models as well as building low-power hardware based on both CMOS and emerging technologies. At the highest level, its necessary to adapt techniques such as model compression, including quantization and pruning, to design DNN models with reduced number of parameters and arithmetic operations. At the lowest level, we should focus on hardware acceleration of DNN models. The main

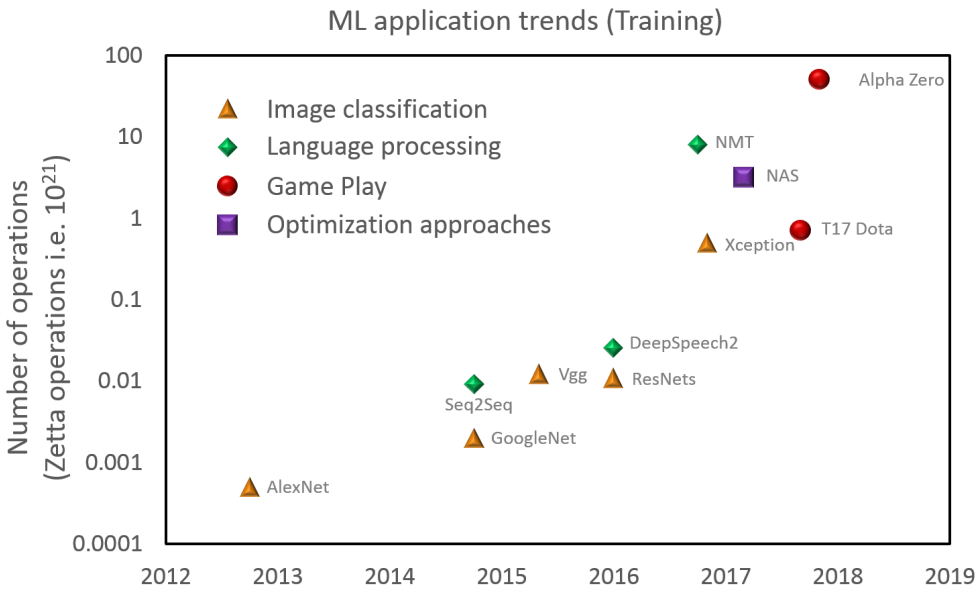


Figure 1.1. Trends of ML applications in terms of computational complexity

computational kernel in a DNN is matrix multiplication, which can be data intensive. It is imperative to accelerate this kernel through parallelism and data reuse and today's digital accelerators [9], [10] go a long way in overcoming the memory bandwidth bottleneck faced by traditional computing systems when performing large-scale matrix multiplications. Despite significant developments in digital DNN accelerators, there has been a growing interest in the enabling computing within the memory array itself by simultaneously activating multiple rows of an array. Such a computing paradigm is known as analog Compute-in-Memory (CIM) where the arithmetic of matrix multiplication is performed in the bit-lines of the memory array [11]–[14]. Such a technique can utilize the full bandwidth of on-chip memories. Further, when implemented on emerging non-volatile memory, analog CIM systems can enable highly dense spatial architectures capable of significantly improvement over digital implementation of DNN accelerators. [15], [16]

In this research, we explore solutions at these aforementioned levels of the design stack, from algorithms to devices, which can pave the way for enabling AI at the edge. First, in order to tackle the exploding computational complexity in DNN models, researchers have explored quantization of the activations and weights of the models. However, in DNNs, different layers can have different quantization limits and hence a heterogeneous quantization across

layers can achieve optimal energy-accuracy tradeoffs. Therefore, in Chapter 2, we propose a Principal Component Analysis (PCA) driven design of mixed precision neural networks [17] where we determine the significance of the layers in Binary Neural Networks [18] using PCA and increase the bit precision of the weights and activations of the significant layers.

Having explored the design of low complexity DL models, we focus on energy-efficient hardware primitives based on non-volatile memory (NVM) based analog compute-in-memory (CIM) through Chapters 3 to 6. Analog CIM can potentially offer massively parallel computations thus leading to higher energy efficiency than its digital counterparts for accelerating ML workloads. To that effect, in Chapter 3, we introduce the background of analog CIM using memristive crossbars [19]–[21]. Despite the promises of analog CIM primitives toward achieving high density and energy efficiency compared to digital accelerators, the inherent analog nature of computing can introduce computational errors due to various device-circuit non-idealities in the system. These non-idealities can arise from parasitic and peripheral resistances, or from non-linear device characteristics. In Chapter 4, we develop mitigation strategies [22] to counteract computational errors in analog CIM primitives. In Chapter 5, we build a novel neural-network based modeling technique GENIEx [23], and develop a simulation framework around GENIEx to evaluate large-scale DNN models on state-of-art CIM accelerators in presence of non-idealities.

Next, we focus on implementation of silicon demonstration of an analog CIM processing system for DNN acceleration. In Chapter 6, we develop an in-memory computing platform based CMOS SRAM memories [24]. In Chapter 7, we demonstrate an implementation of 8T SRAM based CIM in TSMC 65nm technology [25]. The proposed CIM macro can leverage data sparsity to reduce the peripheral ADC overhead and achieve energy efficiency. We further propose a multi-macro sparsity-aware CIM processing core which is designed and implemented in TSMC 65nm technology for an upcoming tape-out. The proposed processing core addresses challenges in scaling up a sparsity-aware analog CIM macro such as low signal-to-noise ratio (SNR) and sparsity imbalance. Finally, in Chapter 8, we explore spike-based photonic computing systems of the future based on phase change materials embedded in photonic devices [26], [27]. Through thorough exploration and investigation of various solutions across the stack, to the computational complexity in DNN models, this research

has established the ground for low-complexity DNN models and corresponding demonstration of hardware acceleration based on analog CIM. It has paved the way for future research in building low-complexity models and large-scale DNN accelerators based on analog CIM.

2. CONSTRUCTING ENERGY-EFFICIENT MIXED-PRECISION NEURAL NETWORKS THROUGH PRINCIPAL COMPONENT ANALYSIS FOR EDGE INTELLIGENCE

2.1 Introduction

The recent advent of 'Internet of Things' (IOT) has deeply impacted our lives by enabling connectivity, communication and autonomous intelligence. With rapid proliferation of connected devices, the amount of data that needs to be processed is ever increasing. These data collected from numerous distributed devices are usually noisy, unstructured and heterogeneous [28]. Deep learning succeeds in reliably processing such complex and large volumes of data where conventional machine learning techniques fail [29]. Thus, it has become the driving force behind ubiquitous Artificial Intelligence (AI), and we see the pervasiveness of deep learning in various applications such as speech recognition, predictive systems and image and video classification [4]–[7].

Traditionally, IOT devices act as data collecting interfaces that feed the deep learning models deployed in centralized cloud computing systems. However, such systems have their own issues and vulnerabilities. In real-time application such as self-driving cars, the latency of communication between IOT devices and the cloud can pose a serious safety risk. As more IOT devices connect to the cloud, it strains the available shared bandwidth for communication. Furthermore, rising concerns around data privacy and over-centralization of information has propelled the need for decentralized user-specific systems [30], [31]. Edge computing [32] is a promising alternative that enables IOT devices to process data, thus reducing communication overhead and latency and ensuring decentralization of data. The facilitation of on-chip analytics offered by edge computing can prove to be pivotal for autonomous platforms such as drones and self-driving cars as well as smart appliances. In addition, smart edge devices can play a significant role in healthcare monitoring systems and medical applications. Intelligent edge devices can be further leveraged for swarm intelligence based applications. However, computing in these resource constrained edge devices comes

with its own challenges. Deep learning models are usually large in size and computationally intensive, thus making them difficult to implement in low-power and memory-constrained IOT devices. Thus, there is a need to design deep learning models which can perform effectively while requiring less memory and less computations.

One approach toward compressing neural network models is to modify the network architecture itself, such that it has fewer parameters, such as SqueezeNet [33]. Another method of compression is pruning which aims to reduce redundancies in over-parameterized networks. To that effect, researchers have investigated several network pruning techniques, both during training [34], [35] and inference [36], [37].

A different technique of model compression is representing weights and activations with reduced precision. Quantized networks [38] help achieve reduction in energy consumption as well as improve memory compression compared to full-precision networks. Binary neural networks [18] are an extreme case of quantization where the activations and weights are reduced to binary representations. These networks drastically reduce the energy consumption by replacing the expensive multiply and accumulate (MAC) operations with simple add or subtract operations. This massive reduction in memory usage and computational cost make them particularly suitable for edge computing. However, despite these benefits, the networks suffer from performance and scalability issues, especially for complex pattern recognition tasks. Several training algorithms [39] have been proposed to optimize network performance to achieve state-of-art accuracy in extremely quantized neural networks. Although such training methodologies recover the performance hit caused by binarizations weights alone, they fail to completely counter the degradation caused by binarizing both weighs and activations.

In this chapter, we present Hybrid-Net, a mixed-precision network topology fashioned by the combination of binary and high-precision inputs and activations in different layers of a network. We use Principal Component Analysis (PCA) to determine significance of layers based on the ability of a layer to expand data into higher dimensional space, with the ultimate aim of linear separability. Viewing a neural network as an iterative projection of input onto a successively higher dimensional manifold at each layer, until the data is eventually linearly separable allows us to identify layers that contribute relevant transformations. Following the

algorithm in [40], we find the 'significant dimensions' in a layer as the number of dimensions that cumulatively explain 99% of the total variance of the output activation map generated by that layer. Since we want the data to be expanded into higher dimensions at each layer, we deem the layers at which significant dimensions increase from the previous layer as significant. Following the identification of significant layers, we increase the bit-precision of the inputs and weights of those layers, keeping the rest of the layers entirely binary. Traditionally, PCA has been used primarily as a dimensionality reduction technique. It was also recently used to identify redundancies in different layers of a neural network and prune out the redundant features[40]. We propose a methodology where we use PCA in a reverse manner, i.e., to increase the precision of the important layers. Hybrid-Net remarkably improves the performance of extremely quantized neural networks, while keeping the activations and weights of the most of the layers binary. This ensures low energy consumption and high memory compression of extremely quantized neural networks while achieving significantly enhanced classification performance compared to binary networks such as XNOR networks. This work not only achieves signficant progress in the challenge of quantizing neural networks to binary representations but also paves way for optimized yet highly accurate quantized networks suitable for enabling intelligence at the edge.

2.2 Related Work

Various techniques have been proposed to improve the performance of quantized networks. Fully binary networks [18], [39] are constructed by replacing the activations with their sign. However, these networks usually suffer from significant degradation in accuracy, especially for larger datasets such as CIFAR-100 and ImageNet. One intuitive way of recovering quantization errors is using wider networks [41] but it comes at the cost of increased energy consumption. There have been efforts focusing on gradient calculations for approximated sign functions to ameliorate the effect of binarization [42]. More general quantization schemes have also been explored for weights and activations [43], [44]. Although weight quantization can be compensated by training the network with quantized weights [38], it has been observed that input quantization pose a serious challenge to classification performance

for precisions lower than 4 bits. One approach that addresses this challenge involve clipping the activations by setting an upper bound. Although this approach seems to be heuristic, recent efforts have focused on using trainable quantization that can be dynamically manipulated[45], [46]. One such approach involves parameterized clipping where the clipping level is dynamically adjusted through gradient descent [47]. Another approach proposed the use of batch-normalization layers after ReLU activations to bound the activation values for effective quantization [48]. Note, that most of these works focus on optimizing the activations when the quantization precision is 2 bits or more. Binary networks with both 1-bit activations and weights, despite offering the most benefits in terms of computation cost and memory compression, still suffer from significant degradation in performance compared to full-precision networks.

An alternative path towards improving the accuracy of binary neural networks focuses on network design techniques. To that effect, improved input representations through shortcut connections in deep networks can significantly improve performance of binary neural networks without any increase in computation cost [42]. This is because shortcut connections are usually identity in nature and do not comprise of expensive MAC operations. Combinations of different kinds of input precisions have also been explored across different layers to circumvent the significant decrease in classification accuracy of such binarized networks [49]. There has been considerable effort in making the search for optimum neural architecture more sophisticated through efficient design space exploration [50]. A theoretical approach towards predicting layer-wise precision requirement has been also explored [51]. Our work differs from most of the current efforts in quantized neural networks as it lies in the realm of hybrid network design for more optimal performance of neural networks where most of the layers still have 1-bit weights and activations. This motivates us to propose an algorithm to identify important layers and judiciously reinforce those particular layers with higher bit-precision representation. To follow such a motivation, it is necessary to understand the significance of layers, which we explain in the next section.

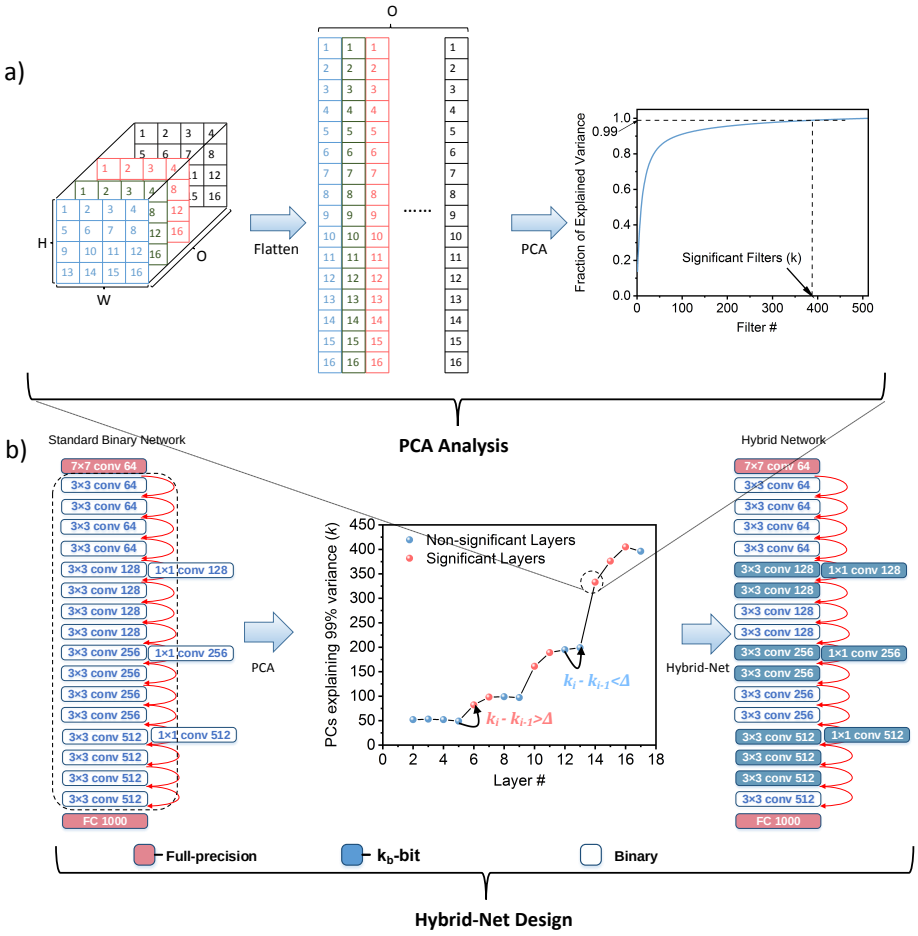


Figure 2.1. (a) The PCA function in Algorithm. 1 for a particular layer showing flattening of an instance of a 4-D tensor and subsequent PCA analysis yielding the plot showing how the cumulative explained variance accrues with the number of filters in a particular layer. Number of Significant filters, k is defined as the number of filters at which the cumulative sum reaches a threshold (say 0.99). (b) The Main() function in Algorithm. 1 is explained as we take a standard binary network (leftmost, with first and final layers having full-precision weights and activations) and perform the aforementioned PCA function on each binary layer (shown in white). The resulting plot (middle) shows the layer-wise variation in k in a ResNet-18 on ImageNet for example. A layer is considered significant (shown by red markers), when k increases by at least Δ. The new Hybrid-Net (rightmost) is designed by increasing the precision of weights and activations for the significant layers (shown in blue).

2.3 PCA-driven Hybrid-Net Design

A Hybrid-Net is a neural network that employs two different bit precisions for its weights and activations. The base network is of low precision, for example 1 bit, and certain layers are selected and set to a higher bit precision. For selecting the layers, we use Principal Component Analysis (PCA) on the output feature maps of each of the layers. The input to any layer is binarized and convolved with the weight filters. We perform PCA on the resulting output tensor. This is performed individually on the output tensor of every layer. Given any set of correlated variables, such as the feature maps, PCA does an orthogonal transformation to map them to uncorrelated variables called Principal Components(PCs), which also form the orthogonal basis set for these tensors. Each of these resulting basis vectors identify directions of varying variance in the data, and are ordered in decreasing manner, with the first vector in the direction of highest variance. We perform such PCA on each convolutional layer in a standalone fashion based on the transformation that it applies on its input. It is applied only on the linear portion of the network before non-linear activation. The aim of PCA is to identify redundancy and how many filters in that layer are necessary to give us a near perfect reconstruction of the output. Based on the redundancy data obtained from the PCA, we define our own significance metric to obtain significant layers from the network.

In a neural network, each layer applies a transformation on its input and projects it to a new feature space of ideally higher or same dimension with the objective of achieving linear separability. PCA provides the ability to study the directions of maximum variance in the input data. The pre-ReLU activation map generated by a filter is considered to composed of many instances of that particular filter. Performing PCA and finding the number of filters needed to explain a pre-defined cumulative percentage of variance identifies the number of significant dimensions for each layer. More the number of principal components needed to preserve a significant percentage, say T, of the total variance in the input, lesser is the redundant information carried by those tensors, and higher is the significant dimensionality of those tensors. Ideally, we want the number of PC's required to explain $T\%$ of the total variance of the feature space to increase as we move deeper into the network in order to extract

more uncorrelated, unique features from the data, and project it into a higher dimensional space that will eventually lead to linear separability at the classifier layer. Thus, the layers for which the number of PCs explaining variance in the output data is more than that in the input data, contribute to significant transformations on the input data. Note, the significant dimensionality of layers does not always monotonically increase. However, regardless of the trend, it provides us a way of judging the pliability of a layer to binarization. In this section, we propose a methodology to identify these significant layers and subsequently design mixed-precision networks by increasing the bit-precision of those layers.

2.3.1 PCA-driven identification of significant layers

We perform our analysis on activations of each layer, which provide a notion of activity of each filter in that layer. Let us consider the activation matrix of the L^{th} layer, X_L. Layer L has O filter banks, each containing I filters of size $k \times k$. I and O are the number of input and output channels. The first element of the ith output map of X_L is the result of convolution of the first $k \times k \times I$ sized input patch with the ith filter bank. The rest of the elements of any particular output map of X_L can be obtained by striding over the entire input. Thus, if we consider M as the size of a minibatch, X_L is a 4-dimensional tensor of size $M \times H \times W \times O$ where H and W are the height and width of the each output map. If we flatten the 4-D data $X_L \in \mathbb{R}^{M \times H \times W \times O}$ into a 2-D matrix $Y_L \in \mathbb{R}^{M*H*W \times O}$, we would obtain $M * H * W$ samples, each containing O elements equivalent to the number of filter banks. This process is shown in Fig. 2.1 (a).

When PCA is performed over the aforementioned 2-D matrix Y_L, the singular value decomposition (SVD) of the mean normalized, symmetric matrix $Y_L^T Y_L$ generates O eigenvectors v_i and eigenvalues λ_i. The total variance in the data is given by sum of the variances of individual parameters:

$$Var = \sum_{\mathrm{i}}^{O} (\sigma_{\mathrm{ii}}^2) = Tr(Y_L^T Y_L) \tag{2.1}$$

The contribution of any component, λ_i, towards the total variance can be expressed as λ_i/Var. To calculate the number of significant components, we set a threshold value T

Algorithm 1: Hybrid-Net Design Methodology

Function `PCA(`*activations,layer,T*`)`

[M,H,W,O] ⟵ size(activations[layer]);
act_fl ⟵ flatten(activations[layer],M*H*W,O);
run PCA on *act_fl*;
tot_var ⟵ total variance;
cum_var ⟵ cumulative sum of variance;
k ⟵ num of components with *cum_var*<T**tot_var*;
return k;

Function `Main()`

Train a N-layer binary network;
Set Threshold T ;
Set Delta Δ ;
Set Delta k_b ; `// Bit Precision of significant layers`
for i ← 0 **to** $N-1$ **do**
    k[i] ⟵ $PCA(activations, \text{i}, T)$;
    if k[i] − k[i − 1] > Δ & i > 0 **then**
        Add to Significant Layer list;
    end
end
Create Hybrid-Net:
for i ← 1 **to** $N-1$ **do**
    if i ∈ Sig_{layer} **then**
        $Prec-layer$ ⟵ k_b
    else
        $Prec-layer$ ⟵ 1
    end
end
Initialize weights with same seed
Train Hybrid-Net

which is amount of variance the first k significant components are able to explain. This can be expressed as:

$$\frac{\sum_{\mathrm{i}=1}^{k}(\lambda_{\mathrm{i}}^{2})}{\sum_{\mathrm{i}=1}^{O}(\lambda_{\mathrm{i}}^{2})} = T \tag{2.2}$$

An example of a typical curve of the cumulative sum of variance for different filter numbers, obtained by PCA, is shown in Fig. 2.1 (a) (rightmost). As the PCA analysis produces the k most significant components to explain T fraction of the total variance, we proceed to identify the significant layers. We define a significant layer as the layer which transforms the input data such that the number of significant components to explain T fraction of the variance, increase from that required for the output of previous layer. Let k_{i} be the number of significant components corresponding to the i^{th} layer. Then, it can be said that layer i contributes a relevant transformation on the input data if $k_{\mathrm{i}} > k_{\mathrm{i}-1}$. It means that the layer requires more significant components to explain the variance in the data at its output than the previous layer. However, for a better control on deciding the important layers, we check the condition whether $k_{\mathrm{i}} - k_{\mathrm{i}-1} > \Delta$ to determine if the i^{th} layer is significant. This is explained in Fig. 2.1 (b) (middle) where the dots marked in red denote the significant layers where $k_{\mathrm{i}} - k_{\mathrm{i}-1} > \Delta$.

2.3.2 Hybrid-Net Design

The PCA analysis helps us identify a set of important layers in an N-layer network. We design a hybrid network, 'Hybrid-Net', where we set the bit-precision of weights and inputs of the important layers to a higher value, $k_b = 2, 4$, than the other layers which have binary weights and inputs. This is shown in Fig. 2.1 (b) (rightmost). The weights and inputs of the first and the final layers of a N-layer network are kept full-precision, according to standard practice [39], [43], [47]. The quantization algorithm for any k_b-bit quantized layer can be readily derived from XNOR-Net [39] where the quantized levels are:

$$q_{k-b}(x) = 2\left(\frac{\lfloor (2^{k_b}-1)(x+1)/2 \rfloor}{2^{k_b}-1} - \frac{1}{2}\right) \tag{2.3}$$

In a layer with k_b-bit weights and activations, $q_{k_b}(x)$ is used instead of the $sign$ function in layers with binary weights and activations. We use a slightly modified version of quantized networks, proposed in [39], where the weights have a scaling factor α instead of just being quantized. The convolution operation in between inputs X and weights W in such a network is approximated as:

$$\text{Binary:} \qquad X * W \approx (sign(X) * sign(W)) \odot \alpha \tag{2.4}$$

$$k_b\text{-bit:} \qquad X * W \approx (q_{k-b}(X) * q_{k-b}(W)) \odot \alpha \tag{2.5}$$

Here, α is the L1-norm of W and act as a scaling factor for the binary weights. In binary layers, the activation gradients are clipped such that they lie between -1 and 1. In the k_b-bit layers, we get rid of the activation gradient clipping for better representation. Each layer of a N-layer Hybrid-Net have either binary or k_b-bit weight kernels and the activations after each convolution are again quantized before passing to the next layer.

Hybrid-Net is expected to have a higher computation cost than a binary network. The parameter Δ decides the number of important layers to consider and hence a penalty is incurred due to increase in bit-precision. We can estimate the penalty in computation cost incurred due to the increase in bit-precision a in network with L_S number of significant layers as

$$Penalty = \frac{\sum\limits_{\mathrm{i} \notin Sig_{Layer}} B_\mathrm{i} + \sum\limits_{\mathrm{i} \in Sig_{Layer}} B_\mathrm{i} \times p}{\sum\limits_{\mathrm{i}=1}^{N} B_\mathrm{i}} \tag{2.6}$$

Here B_i is the computation cost of a binary layer and p is the overhead of k_b-bit computation over binary computation. We will present a detailed analysis of energy consumption and memory usage later in the manuscript (A).

For residual networks, ResNets, we include another design feature in addition to the PCA-driven Hybrid-Net, improving input representations through residual connections. This has been alluded to by Liu et al in [42] where adding identity shortcut connections at every layer improves representational capability of binary networks. In standard residual networks [6], such identity connections are added to address the vanishing gradient problem in deep

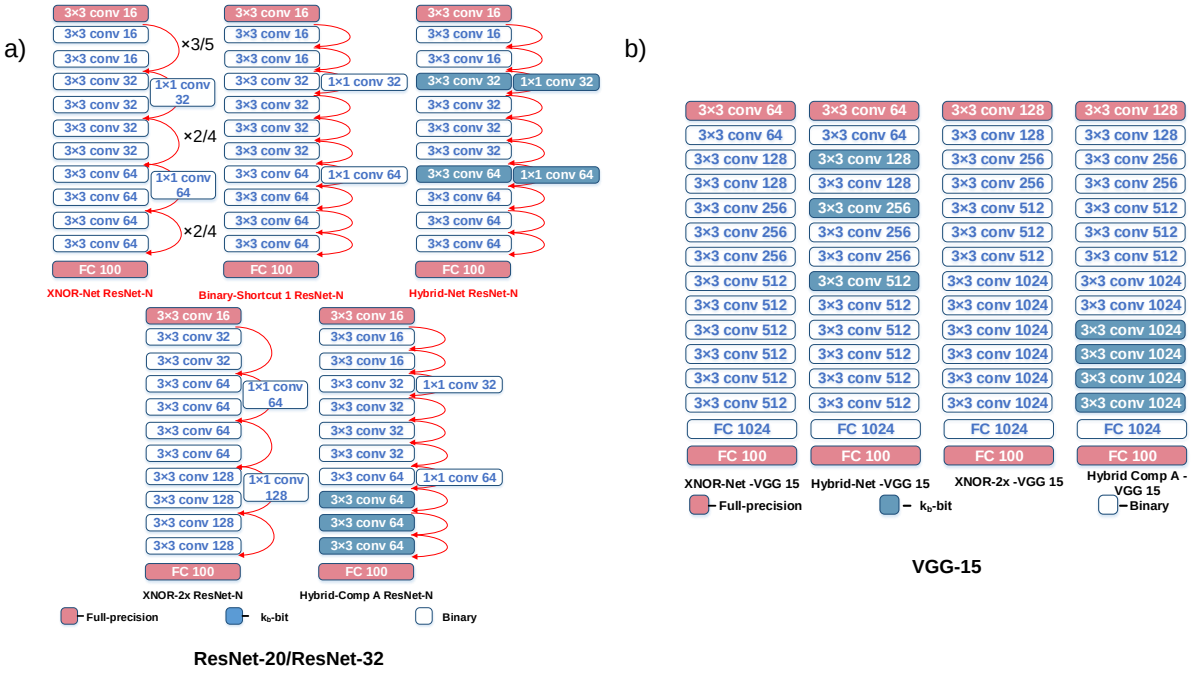


Figure 2.2. Different network configurations based on a) ResNet-20/ResNet-32 and b) VGG-15 architectures showing binary, k_b-bit and full-precision layers in each of the networks as described in Section 4.2. The width of the layer shown in this figure is for CIFAR-100 dataset

neural networks. However, in case of binary networks, these connections serve to provide an improved representation by carrying floating-point information from the previous layer. As a result, the Hybrid-Net design also considers the effect of adding such highway connections at every layer. Note, in case of convolution layers which induce a change in size of each feature map, the shortcut connections consist of 1×1 convolution weight layers to account for the change in size [6].

2.4 Experiments, Results and Discussion

2.4.1 Experiments

We evaluated the performance of all the networks described in this section 4.2 in PyTorch [52]. We perform image classification on the datasets CIFAR-100 [53] and ImageNet [54]. The CIFAR-100 dataset has 50000 training images and 10000 testing images of size 32×32

for 100 classes. For the CIFAR-100 dataset, we explore the proposed Hybrid-Net design methodology on standard network architectures, ResNet-20, ResNet-32 and VGG-15, where the training algorithm for the quantized layers has been adopted from [39]. We extended our analysis to the ImageNet dataset [54] which is the most challenging dataset pertaining to image classification tasks. It consists of 1.2 million training images and 50000 validation images divided into 1000 categories. For simplicity, we considered ResNet-18 for our ImageNet evaluation. To compare against the proposed Hybrid-Net designs, we explore different network configurations as baselines, for ResNet, shown in Fig. 2.2 (a) and VGG architectures, shown in Fig. 2.2 (b). XNOR-Net is the base skeleton network we design other networks on. Binary-Shortcut is same as XNOR-Net except it has residual connections every layer similar to [42]. Hybrid-Comp A is formed by inter-layer sectioning, i.e., dividing the network into 2 parts ($N-k$ binary and k k_b-bit layers) where N is the number of the layers between the first and last layer. The widths of the network architectures, shown in Fig. 2.2 (a) and Fig. 2.2 (b) are for CIFAR-100 dataset. For ImageNet, we have used a wider network architecture, which we describe in Table. 2.1. We have also compared our proposed Hybrid-Net designs against state-of-art quantized networks such as [43], [45], [47] for ImageNet. We performed simulations for 5 different random initializations for all networks on CIFAR-100 dataset. For simplicity, we performed simulations for 3 different random initializations on 4 selected networks and 1 initialization for the rest of the networks on ImageNet dataset. The accuracy reported for both datasets is the top-1 accuracy and the Mean $\pm$ SD accuracy provides the mean and standard deviation of accuracies obtained for different random initializations.

Energy efficiency and Memory compression

We have briefly alluded to the possible penalty incurred due to increasing the bit-precision of certain layers in a network. To identify its effect with respect to the entire network metrics and further illustrate the benefits of the proposed Hybrid-Nets, we perform a storage and computation analysis to calculate the energy efficiency and memory compression of the

Table 2.1. Networks architectures for ImageNet classification task

ResNet - 18
7×7 conv 64 stride 2
3×3 maxpool stride 2
3×3 conv 64 stride 1 (× 4)
3×3 conv 128 stride 2
3×3 conv 128 stride 1 (× 3)
3×3 conv 256 stride 2
3×3 conv 256 stride 1 (× 3)
3×3 conv 512 stride 2
3×3 conv 512 stride 1 (× 3)
Linear 1000

proposed networks. For any two networks A and B, the energy efficiency and memory compression of Network A with respect to Network B can be defined as:

$$\begin{aligned} \text{Energy Efficiency } (E.E) &= \frac{E_A}{E_B} \\ \text{Memory Compression } (M.C) &= \frac{M_A}{M_B} \end{aligned} \tag{2.7}$$

where E_A and E_B are the energy consumed by Network A and Network B respectively, M_A and M_B are the memory used for storing the weights of Network A and Network B, respectively. We estimate energy efficiency (E.E) and memory-compression (M.C) with respect to an full-precision network and normalize it with respect to an XNOR-Net network which is an entirely binary network except the first and final layer. Thus, the normalized E.E ($E.E_{Norm}$) and normalized M.C ($M.C_{Norm}$) of any network A can be written as:

$$\begin{aligned} E.E(A) &= \frac{\sum_{\mathrm{i}} E_{\mathrm{i}}(FP)}{\sum_{\mathrm{i}} E_{\mathrm{i}}(A)} \\ E.E_{Norm}(A) &= \frac{E.E(A)}{E.E(XNOR)} \\ M.C(A) &= \frac{\sum_{\mathrm{i}} M_{\mathrm{i}}(FP)}{\sum_{\mathrm{i}} M_{\mathrm{i}}(A)} \\ M.C_{Norm}(A) &= \frac{M.C(A)}{M.C(XNOR)} \end{aligned} \tag{2.8}$$

Here, $E_{\mathrm{i}}(FP)(M_{\mathrm{i}}(FP))$ is the energy (memory) consumed by the i^{th} layer of a network with full-precision weights and activations whereas $E_{\mathrm{i}}(A)(M_{\mathrm{i}}(A)$ is the energy (memory) consumed by the i^{th} layer of any network A under consideration.

2.4.2 Results - PCA

We perform PCA analysis on the activations of each convolutional layer and extract the number of principal components required to explain a T fraction of variance in the data. The design parameters such as T and subsequently Δ are heuristically. For all analysis, we fix $T = 99\%$ as this makes the increases in significant components, k across various layers clearly distinguishable. The Δ values are chosen based on the variation in k across layers. A higher Δ value yields less number of significant layers. For clarity, we perform our analysis for various Δ values.

ResNet Architectures - CIFAR-100

For ResNet architectures, we perform the PCA on a plain version of a binary network devoid of any residual connections. We decided to do this to isolate the effect of the convolution layers on the activations, instead of having residual connections. This is done because we focus on the quantization of the filters of the layers and the residual additions may distort the output feature space and hence the information we seek from it. Fig. 2.3 (a) and (b) shows the variation in the number of filters required to explain $T = 99\%$ with different layers for ResNet-20 and ResNet-32 architectures respectively. As expected, the maximum change in k occurs when the number of output channels increase. However, we observe a trend in both networks, that the layers just after the output channels increase from, say 16 to 32 or 32 to 64, attribute for the maximum change in the number of significant filters. Based on our criteria for significant layers, discussed in Section 3.1, we fix a $\Delta = 1$ for Resnet-20 and $\Delta = 4$ for ResNet-32 to identify the layers where the number of significant components undergo a change more than Δ. Fig. 2.2 (a) also shows those layers marked by red dots. Note, by varying the Δ, more or less number of layers can be considered as significant. After

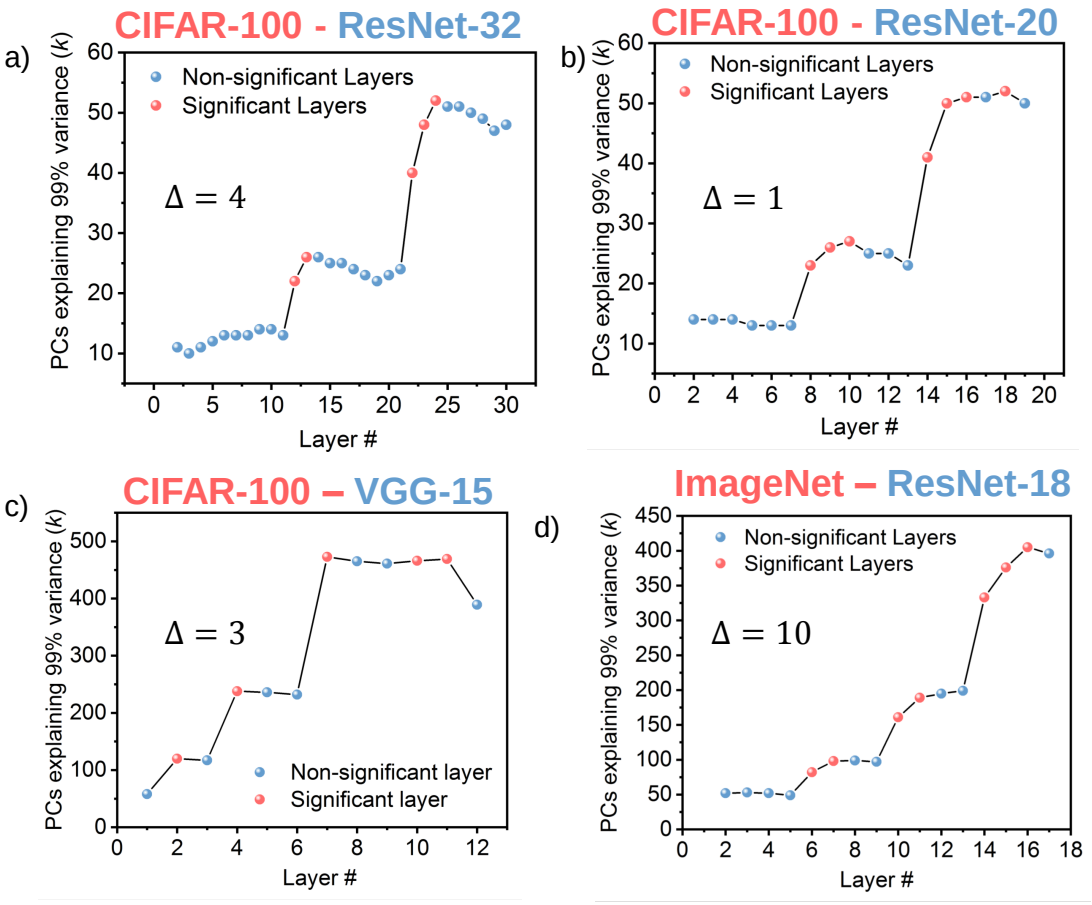


Figure 2.3. PCA analysis plot showing the number of principal components required to explain 99% variance at the output of convolutional layers across different layers for a) ResNet-32 ($\Delta = 4$), b) ResNet-20 ($\Delta = 1$), and c) VGG-15 ($\Delta = 3$) on CIFAR-100 dataset and d) for ResNet-18 ($\Delta = 30$) on ImageNet dataset

performing this analysis on a plain version of the ResNet architecture, we perform network simulations on the standard version with residual connections.

VGG Architectures - CIFAR-100

For VGG architecture, we perform the PCA of a binary network which has binary weights and activations for all layers except the first and the last. Fig. 2.3 (c) shows the plot showing how the number of filters required to explain $T = 99\%$ of the variance changes with different layers for a VGG-15 architecture. We observe that the number of significant filters mostly

Table 2.2. Significant Layers identified by PCA analysis

CIFAR-100	
Network Arch	Significant layers
ResNet-20 ($\Delta = 1$)	8, 9, 10, 14, 15, 16, 18
ResNet-20 ($\Delta = 2$)	8, 9, 14, 15
ResNet-32 ($\Delta = 4$)	12, 13, 22, 23, 24
VGG-15 ($\Delta = 3$)	3, 5, 8, 11, 12
VGG-15 ($\Delta = 10$)	3, 5, 8
ImageNet	
Network Arch	Significant layers
ResNet-18 ($\Delta = 30$)	6, 10, 14, 15
ResNet-18 ($\Delta = 20$)	6, 10, 11, 14, 15, 16
ResNet-18 ($\Delta = 10$)	6, 7, 10, 11, 14, 15, 16

increase when the number of filter bank increases at a particular layer. For rest of the layers, it remains fairly constant. As the PCA plot shows very little variation across layers, we consider a relatively lower $\Delta = 3$ with respect to the number of filters. We mark the significant layers by red dots in Fig. 2.3 (c). Table. 2.2 lists the different combination of significant layers obtained from ResNet and VGG architectures through the PCA analysis for different Δ values for CIFAR-100 dataset. Note, we did not choose a lower Δ for ResNet-32 as it would have included many layers which would increase the computation cost without a significant benefit in accuracy.

ResNet Architectures - ImageNet

We further perform PCA analysis on ResNet-18 architecture for the ImageNet dataset. Fig. 2.3 (d) shows the plot showing how the number of filters required to explain $T = 99\%$ of the variance changes with different layers for ResNet-18 for $\Delta = 10$. The significant layers identified by our proposed methodology are marked with red dots. We observe a similar trend as in case of CIFAR-100, that maximum increase in the number of significant filters, k, occur in the first few layers after every change in filter size. We perform the PCA analysis for $\Delta = 10, 20, 30$ to identify the significant layers, listed in Table. 2.2.

2.4.3 Image Classification Results - CIFAR-100

ResNet Architectures

The ResNet-N architecture consists of N-1 convolution layers and a fully-connected classifier. As discussed before, the first convolution layer and the classifier have full precision inputs and weights. For CIFAR-100 dataset, we consider $N = 20$ and $N = 32$. Further, we consider a slightly modified version of ResNet, where we add identity shortcut connections at every layer instead of every two layers for better input representation, as discussed earlier. We increase the bit-precision of weights and inputs of the layers obtained from PCA analysis to bit precisions $k_b = 2$ and $k_b = 4$ to form Hybrid-Net (k_b, k_b). The rest of the layers have binary representations for weights and inputs. We also compare the proposed Hybrid-Net with Hybrid-Comp A (k_b, k_b) (k), which is formed by splitting the entire network into $N - k$ binary and k k_b-bit sections. Table. 2.4 shows that accuracy, energy efficiency and memory compression of the proposed Hybrid-Net based on ResNet-20 and ResNet-32 in comparison to XNOR-Net and other kinds of hybrid networks discussed in Fig. 2.2.

We observe that the proposed Hybrid-Net achieves a much superior trade-off between accuracy, energy efficiency and memory compression compared to other kinds of hybridization techniques. Moreover, in case of both ResNet-20 and ResNet-32, Hybrid-Net increases the classification accuracy by 10-11% compared to a XNOR-Net with minimal degradation in efficiency and compression. For example, Hybrid-Net (2,2) ($\Delta = 1$) based on ResNet-20 can be expected to have only $36\% \pm 0.03\%$ of the accuracy degradation of a XNOR network with respect to a full precision network while only costing 13% extra energy. While quantizing the entire network to 2-bit inputs and weights (Quantized (2,2)) achieves a slightly higher accuracy, we show that the our principle of increasing the bit-precision of few significant layers captures most of the increase in accuracy from an XNOR-Net to a 2-bit networks. Hybrid-Net thus consumes 14% less energy and 12% less memory for ResNet-20 than a 2-bit network with a performance within 2% of the latter. For ResNet-32, the benefits of Hybrid-Net is even pronounced where it consumes 24% less energy and 26% less memory than a 2-bit network while achieving accuracy within 4% of the latter. Hybrid-Net thus ensures a signficant improvement in accuracy over a binary network without making the entire network

Table 2.3. Comparison of different networks on CIFAR-100

ResNet-20				
FP Accuracy - 69.49%				
$E.E(XNOR)$ **- 16.35**, $M.C(XNOR)$ **- 17.26**				
Network Type	*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	$E.E_{Norm}$	$M.C_{Norm}$
XNOR	50.50	50.23 ± 0.21	1	1
Binary- Shortcut 1	54.16	53.92 ± 0.21	0.99	1
Hybrid-Net (2,2) (Δ=1)	62.84	62.5 ± 0.24	0.87	0.77
Hybrid-Net (2,2) (Δ=2)	60.93	60.53 ± 0.39	0.93	0.88
Hybrid-Net (4,4) (Δ=1)	63.88	63.38 ± 0.49	0.7	0.53
Hybrid-Net (4,4) (Δ=2)	61.62	61.53 ± 0.1	0.82	0.7
Quantize(2,2)	65.81	65.19 ± 0.49	0.73	0.65
Hybrid-Comp A (2,2)(k=6)	62.36	61.79 ± 0.34	0.88	0.71
XNOR2x	63.03	62.81 ± 0.14	0.39	0.33
Resnet-32				
FP Accuracy - 70.62%				
$E.E(XNOR)$ **- 18.42,** $M.C_{Norm}$ **- 20.44**				
Network Type	*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	$E.E_{Norm}$	$M.C_{Norm}$
XNOR	53.89	53.48 ± 0.27	1	1
Binary- Shortcut 1	58.98	58.23 ± 0.61	0.99	1
Hybrid-Net (2,2) (Δ=4)	64.34	63.75 ± 0.39	0.94	0.87
Hybrid-Net (4,4) (Δ=4)	64.45	64.28 ± 0.18	0.84	0.69
Quantize(2,2)	68.04	67.73 ± 0.21	0.7	0.61
Hybrid-Comp A (2,2)	62.41	62.15 ± 0.2	0.91	0.76
XNOR2x	65.20	65.11 ± 0.07	0.38	0.31

2-bit. We also show that Hybrid-Net achieves a higher accuracy than Hybrid-Comp A networks while consuming less energy for both ResNet-20 and ResNet-32, thus demonstrating the effectiveness of the design methodology.

Table 2.4. Comparison of different networks on CIFAR-100 for VGG-15

VGG-15				
FP Accuracy - 68.31%				
$E.E(XNOR)$ **- 21.77,** $M.C_{Norm}$ **- 26.24**				
Network Type	*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	$E.E_{Norm}$	$M.C_{Norm}$
XNOR	54.30	54.23 ± 0.1	1	1
Hybrid-Net (2,2) (Δ=3)	61.81	61.67 ± 0.08	0.84	0.75
Hybrid-Net (2,2) (Δ=10)	60.13	59.87 ± 0.25	0.93	0.92
Hybrid-Net (4,4) (Δ=3)	63.38	63.12 ± 0.15	0.64	0.5
Hybrid-Net (4,4) (Δ=10)	60.37	60.06 ± 0.18	0.81	0.8
Quantize(2,2)	68.90	68.63 ± 0.28	0.65	0.55
Hybrid-Comp A (2,2) (k=3)	58.01	57.46 ± 0.38	0.85	0.72
XNOR2x	58.24	57.35 ± 0.54	0.29	0.3

VGG architecture

We further extend our analysis to VGG architectures. We considered VGG-15, which consists of 13 convolutional and 2 fully-connected layers as shown in Fig. 2.2 (b). We kept one of the fully-connected layer binary to preserve energy-efficiency. Table 2.4 lists the accuracy, energy efficiency and memory compression results for VGG-15 on CIFAR-100 for different networks. We consider $\Delta = 3$ and $\Delta = 10$ for our analysis and for each of the network configurations we use $k_b = 2, 4$ for the significant layers. We observe that Hybrid-Net achieves 13% higher accuracy than a XNOR-Net with minimal degradation in efficiency. When we make the inputs and weights of the entire network 2-bit (Quantize (2,2)), we achieve an even higher accuracy. We also show that Hybrid-Net with 2-bit layers achieve a better performance than Hybrid-Comp A for iso-efficiency in energy and memory. Similar to trends in ResNet, we observe that making the significant layers 4-bit while keeping the rest of the layers binary improves performance, however, at the cost of energy-efficiency. An entirely 2-bit network proves to be a more efficient solution. In summary, even for VGG architecture, we show that Hybrid-Net achieves 7.44% higher classification accuracy compared to a XNOR network while keeping most of the layers binary.

Table 2.5. Comparison of different networks on ImageNet

Resnet-18					
FP Accuracy - 69.15%					
$E.E(XNOR)$ **- 8.57**, $M.C(XNOR)$ **- 13.35**					
Network Type		*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	$E.E_{Norm}$	$M.C_{Norm}$
XNOR		50.33	–	1	1
Binary-Shortcut 1		54.36	54.15 ± 0.15	1	1
Bi-Real Net [42]		56.9	–	1	1
Hybrid-Net (2,2) ($\Delta = 30$)		60.38	59.75 ± 0.44	0.96	0.87
Hybrid-Net (2,2) ($\Delta = 20$)		61.95	61.89 ± 0.04	0.93	0.8
Hybrid-Net (2,2) ($\Delta = 10$)		62.73	–	0.92	0.8
Hybrid-Net (4,4) ($\Delta = 30$)		61.70	60.54 ± 0.84	0.89	0.7
Quantize (2,2)	XNOR-kbit	64.51	–	0.84	0.71
	DoReFA [43]	62.6	–		
	PACT [47]	67	–		
	LQ-Nets [45]	64.9	–		
Hybrid [49]		54.9	–	0.68	1
Hybrid-Comp A (2,2) (k=4)		59.47	–	0.94	0.77

2.4.4 Image Classification Results - ImageNet

We evaluate the proposed Hybrid-Net design Table. 2.5. We observe that the XNOR network suffers a significant degradation in accuracy from a full-precision network. Even the Binary-Shortcut 1 network with residual connections at every layer fail to recover the classification accuracy. With improved quantization schemes and weight update mechanisms, Bi-Real Net [42] has shown a slightly higher accuracy. Compared to these binary networks, we observe that the proposed Hybrid-Net, considering both 2-bit and 4-bit weights and activations achieves upto over 10 % higher accuracy than corresponding XNOR network. In particular, Hybrid-Net (2,2) ($\Delta = 20$) can be expected to have only 38.6% ± 0.002% of the accuracy degradation of a XNOR network with respect to a full-precision network while only costing 7% extra energy. Quantizing the activations and weights of the entire network to 2-bits can further increase the accuracy by 1-2% but at the cost of a 15-20% increase

in energy consumption than Hybrid-Net. Note, that we have provided results as baselines for different input quantization algorithms, such as DoReFA-Net [43], LQ-Net [45], and PACT [47], for the Quantize (2,2) network although we have used the XNOR quantization (described in Section 3.2) in this chapter. We also show Hybrid-Net (2,2) achieves upto 7.4% higher accuracy with respect to other methods of hybridization [49]. This work shows that increasing the bit-precision of a few significant layers can remarkably boost the performance of binary neural networks without making the entire network higher precision. Note, that using improved quantization schemes [42], [47] can potentially further increase the accuracies of the proposed Hybrid-Nets.

2.4.5 Statistical Analysis

We have mentioned in an earlier subsection that we perform simulation for various random initializations. To understand the amount of variations in the results, we performed simulations for two cases:

(a) Fixed Optimal Solutions: We define a binary base network with any random initialization. We run a PCA analysis and obtain a optimal Hybrid-Net. Next, we train the Hybrid-Net with various random initializations.

(b)Varying Optimal Solutions: We train the base binary network with different random initializations and run PCA analysis on each case. This provides various optimal solutions of Hybrid-Net with different combinations of significant layers. Then we perform accuracy simulations for corresponding random initializations.

Fixed Optimal Solutions:

We performed simulations for all cases explained in the chapter for 5 different random initializations. For simplicity, we considered 3 random initializations for 4 networks for ImageNet. Note, the energy and memory costs depend on the network architecture and is fixed for a particular architecture. The results in Table. 2.4 and 2.5 shows the variations in accuracies are within 2% of the best accuracies.

Varying Optimal Solutions:

We trained the base binary network for ResNet-20 on CIFAR-100 for different initializations and performed PCA analysis on each of them to obtain varying optimal solutions for different Δ values. We observe that the optimal solutions overlap quite significantly. The resulting energy efficiency and memory compression metrics also do not vary remarkably. We perform accuracy analysis for each optimal solution for their corresponding random initialization to observe the variations in results. The results are presented in Table. 2.6. Based on the analysis, we can expect the Hybrid-Net (2,2) (Δ=1) to have only 37.7 $\pm$ 7E-3 % of the accuracy degradation of a XNOR network with respect to a full-precision network with only 12 $\pm$ 1.22 % extra energy cost. Similarly, Hybrid-Net (2,2) (Δ=2) is expected to have only 41 $\pm$ 0.01 % of the accuracy degradation of a XNOR network with respect to a full-precision network with only 9 $\pm$ 1.4 % extra energy cost.

Table 2.6. Analysis of effect of random initialization on varying optimal Hybrid-Net architecture (ResNet-20 on CIFAR-100)

Δ	*Init*	*Significant layers*	*Best Accuracy (%)*	*Mean $\pm$ SD Accuracy (%)*	$E.E_{Norm}$	$M.C_{Norm}$
1	1	8,9,10,14,15,16,18	62.84	62.46 $\pm$ 0.23	0.88	0.77
	2	5,8,9,14,15,16	62.14	61.59 $\pm$ 0.29	0.89	0.82
	3	7,8,9,14,15,16	62.17	61.85 $\pm$ 0.19	0.89	0.82
	4	2,8,9,10,14,15,16,18	63.4	63.15 $\pm$ 0.13	0.86	0.77
2	1	8,9,14,15	61.49	60.72 $\pm$ 0.52	0.93	0.88
	2	8,9,14,15,16	61.29	61.48 $\pm$ 0.29	0.91	0.83
	3	6,8,9,14,15,16	61.70	61.82 $\pm$ 0.53	0.89	0.82
	4	8,9,14,15,16	61.87	61.48 $\pm$ 0.29	0.91	0.83

Following this analysis, we can conclude that variation in classification accuracy due to different random initializations vary within a range of 2% for all cases considered. Further, it is true that random initializations lead to varying optimal solutions which would mean the PCA step needs to be performed for every network initialization and dataset. For edge applications, the network can be tuned for a target application during initial stages of deployment.

2.4.6 Optimality Studies

The optimality of the proposed Hybrid-Net configurations can be understood through two visualizations. First, we compare the network designed through the proposed PCA-driven methodology with arbitrarily defined network architectures with randomly chosen layers as k_b-bit precision. We have performed this analysis on ResNet-32 for CIFAR-100 dataset. For identical comparisons to the proposed network Hybrid-Net(2,2) ($\Delta = 1$), we have defined networks with randomly chosen 6 or 7 layers with k_b-bit precision. Table. 2.7 shows the networks. The numbers within the bracket signifies the layers with k_b-bit precision. The rest of the layers in this network is binary. Fig. 2.4 (a) shows the resulting plot.

Table 2.7. Network Configurations with randomly chosen layers as k_b-bit precision

Network index	*Network Configurations*	*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	*Energy*
N1	Hybrid-Net (2,2) (Delta=4)	64.34	63.75 ± 0.39	0.94
N2	Hybrid-Net (25, 26, 27, 28, 29, 30, 31)	62.70	62.62 ± 0.09	0.90
N3	Hybrid-Net (2, 11, 12, 20, 21, 23, 29)	63.43	63.00 ± 0.37	0.91
N4	Hybrid-Net (12, 17, 18, 20, 24, 25, 26)	63.81	63.46 ± 0.21	0.91
N5	Hybrid-Net (2, 5, 6, 7, 20, 28)	61.26	61.04 ± 0.24	0.92
N6	Hybrid-Net (2, 17, 22, 25, 28, 30)	63.57	63.43 ± 0.12	0.93

We observe that hybrid network configurations with similar energy efficiencies lead to different accuracies. This gives the intuition that some layers might be more significant than others. This is the premise our technique is based on where we provide a methodology to identify those significant layers. We observe that PCA-driven methodology of designing Hybrid-Net achieves a better energy-accuracy tradeoff than networks with randomly chosen layers as k_b-bit precision.

Another visualization of optimality would be a comparison of the networks with varying Δ and other baselines considered in this work. Here, we plot the best accuracies obtained for different configurations of ResNet-20 on CIFAR-100 and ResNet-18 on ImageNet described in Table. 2.4 and Table. 2.5. Fig. 2.4 (b) shows such a plot.

Note, that its difficult to comment on Pareto frontier without doing an exhaustive search of networks which can prove to be quite time expensive. In this chapter, we have focused

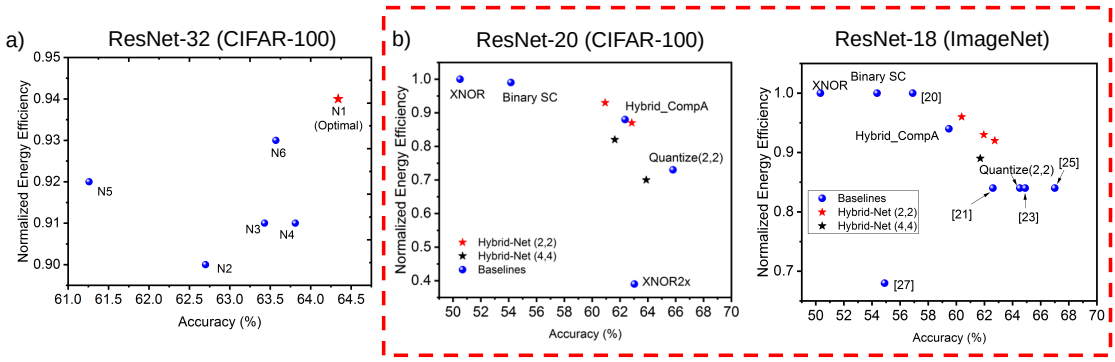


Figure 2.4. Illustration of energy-accuracy optimality of Hybrid-Net. a) Accuracy v/s Energy efficiency plot showing that PCA-driven Hybrid network design achieves more optimal tradeoffs than randomly chosen layers.. b) Normalized Energy Efficiency v/s Accuracy plot showing optimality boundaries for the considered network configurations. It shows that Hybrid-Net (2,2) networks lie right on the optimal boundary among the networks considered. Note, that PACT involves a more advanced quantization algorithm than other networks.

on improving the accuracy of extremely quantized neural networks without a significant degradation in energy efficiency. Our technique proposes a methodology to design hybrid networks through a PCA-driven significance analysis, which achieves 10% higher accuracy with less than 6-10% increase in energy consumption. Out of the network configurations considered, Hybrid-Net (2,2) configurations shows that the proposed design principle can lead us to optimality for both CIFAR-100 and ImageNet. The absolute Pareto optimality is difficult to gauge without performing an exhaustive search.

2.4.7 Discussion

The proposed Hybrid-Net design uses PCA-driven hybridization of extremely quantized neural networks, resulting in significant improvements and observations as listed. One key contribution of the proposed methodology is that we can design hybrid networks without any iterations. It does not require an iterative design space exploration to identify optimal networks. Moreover, this methodology shows that increasing the bit-precision of only the significant layers in a binary network achieves performance close to that of a network that is entirely composed of layers with higher bit-precision weights and activations. Intuitively,

a 2-bit network performs much better than a binary network. However, our analysis shows that it is not necessary to make the weights and activations of the entire network 2-bit. Hybrid-Net achieves more than $\sim 10\%$ improvement over a XNOR network, which is a fully binary network except the first and final layers, by increasing the bit-precision of less than half of the entire network. In fact, for deeper networks, like ResNet-34, this improvement is achieved with only $\sim 6\%$ increase in energy consumption from a XNOR-Net [39]. Thus, Hybrid-Net goes a long way in reaching close to high-precision accuracies with networks which are mostly binary and attain comparable energy-efficiency and memory compression to binary networks such as XNOR-Net [39] and BNN [18]. Moreover, this methodology can be extended to any network where making significant layers of the network k_{b2}-bit while keeping the rest of the network k_{b1}-bit ($k_{b2} > k_{b1}$), can potentially produce comparable performance with enhanced energy-efficiency than an entirely k_{b2}-bit network.

Secondly, the performance of Hybrid-Net is subject to the nature of the plots obtained from PCA on the binary version of the networks. For example, for ResNet architectures (Fig. 2.3 (a), (b) and (c) for CIFAR-100 and Fig. 2.3 (d) for ImageNet), we observe the number of significant components increase for the layers which are adjacent to the ones where the number of output channels increase and then decrease for the later layers which have the same number of output channels. It can be said that the later layers are not adding to the linear separability of the data and binarizing them preserve the accuracy as observed. Or in other words, the significant layers identified using our proposed methodology contribute remarkably higher to the linear separability than the other layers. This is reflected in the results where we show the performance difference between Hybrid-Net and a 2-bit network is less than 4%. However, for VGG architectures, we observe that the PCA plot remains fairly flat, which means that the identified significant layers are not remarkably different in their contribution towards linear separability of the data in comparison to the other layers. This is reflected in the performance difference ($> 6\%$) of Hybrid-Net from a 2-bit network for a VGG-15 network.

Thirdly, we observe that increasing the bit-precision of the weights and activations of the significant layers to 4-bits while keeping the rest of the layers binary is not the most energy-efficient way of improving accuracy of a network. An entire 2-bit network proves to be more

energy-efficient while performing better. It may be because the loss due to binarization can not be significantly recovered by increasing the bit-precision much higher than binary, while keeping most of the layers binary. Thus, the proposed methodology performs best when the precision of the significant layers is close to the base precision of the network (binary in our case).

Finally, the precision of the first and final layers is of utmost important for extremely quantized neural networks. To study the influence of quantization on these layers, we performed experiments on our Hybrid-Net designs by making the weights of the final layer binary while having full-precision inputs as suggested in [27]. The results for ResNet-32 for CIFAR-100 for the configuration Hybrid-Net(2,2) (Δ=4) are listed below:

Table 2.8. Analysis of quantization of first and last layers in Hybrid-Net (2,2) (Delta=4) on ResNet-32 for CIFAR-100

Last Layer Configuration	*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	*Energy Consumption*	*Memory*
Full-Precision	64.34	63.75 ± 0.39	1	1
Binary weights and activations	56.93	56.83 ± 0.09	0.98	0.75
Binary weights and full-precision activations	61.07	60.36 ± 0.44	0.98	0.75
2-bit weights and activations	62.41	62.35 ± 0.05	0.98	0.76
First Layer Configuration	*Best Accuracy (%)*	*Mean ± SD Accuracy (%)*	*Energy Consumption*	*Memory*
Binary weights and activations	44.79	44.16 ± 0.74	0.85	0.98
Binary weights and full-precision activations	59.94	59.29 ± 0.57	0.93	0.98
2-bit weights and activations	60.87	60.46 ± 0.25	0.85	0.98

Energy Consumption: The energy consumption reduction after quantizing the last layer of the neural network is only 2% for CIFAR-100. On the other hand, quantizing the last layer leads to at least 2% degradation in accuracy (when the last layer is 2-bit precision). The first layer consumes more energy due to larger output map size. We performed this analysis by quantizing the first layer as well. We observe that although the energy consumption reduces by 15%, the accuracy also degrades by 3.5%.

Memory: The memory requirements of the last layer is a significant aspect. We observe that quantizing the last layer can lead to close to 25% lower memory. In case of ImageNet,

this will be even more significant and the memory reduction can be upto 2x. However, fully binarizing the last layer leads to significant accuracy degradation. Thus, keeping them higher precision (2-bit or 4-bit) will lead to better accuracy-memory tradeoffs.

In this chapter, we have considered the quantization scheme, explored in [39]. Since then, there has been a plethora of works focused on improving quantization for both inputs and weights [43], [47]. Hybrid-Net focuses on improving the performance of binary neural networks through mixed-precision network design and we believe the improved quantization schemes should further increase the accuracy of both Hybrid-Nets and entirely 2-bit or 4-bit networks.

The humongous computing power and memory requirements of deep networks stand in the way of ubiquitous use of AI for performing on-chip analytics in low-power edge devices. The significant energy efficiency offered by the compressed hybrid networks increases the viability of using AI, powered by deep neural networks, in edge devices. With the proliferation of connected devices in the IOT environment, AI-enabled edge computing can reduce the communication overhead of cloud computing and augment the functionalities of the devices beyond primitive tasks such as sensing, transmission and reception to in-situ processing.

2.5 Conclusion

Binary neural networks offer significant energy-efficiency and memory compression compared to full-precision networks. In this chapter, we propose a one-shot methodology for designing mixed-precision, hybrid networks with binary and higher bit-precision inputs and weights to improve the performance of extremely quantized neural networks in terms of classification accuracy while still achieving significant energy efficiency and memory compression. The proposed methodology uses PCA to identify significant layers in a binary network which transform the input data such that the output feature space require more significant dimensions to explain variance in data. PCA is usually exploited to perform layer-wise dimensional reduction. We use PCA in an opposite manner in order to determine which layers cause the number of signficant dimensions to increase across input and output. Next, we increase the bit-precision of the weights and activations of the significant layers and

keeping that of the other layers binary. The proposed Hybrid-Net achieves more than $\sim 10\%$ improvement over XNOR networks for ResNet and VGG network architectures on CIFAR-100 and ImageNet with only $\sim 6-10\%$ increase in energy consumption, thus ensuring more than $15-20x$ reduction in energy consumption and memory compression from full-precision networks. Memory compression along with the close match to high-precision accuracies offered by the proposed mixed-precision network design using layer-wise information allows us to explore interesting possibilities in the realm of hardware-software co-design. This chapter thus proposes an effective, one-shot methodology for designing hybrid, compressed neural networks and potentially paves the way toward using energy-efficient hybrid networks for AI-based on-chip analytics in low-power edge devices with accuracy close to full-precision networks.

3. RESISTIVE CROSSBARS AS BUILDING BLOCKS FOR MACHINE LEARNING

3.1 Introduction

Rapid advances in Machine Learning (ML) and Artificial Intelligence (AI) have touched human lives in an unprecedented way as the application space has proliferated over the last decade [55]. Deep Neural Networks (DNNs), in particular, have had enormous success in several ML tasks such as image classification [4], [6], object detection [7] and natural language processing [56]. However, the capabilities of deep learning algorithms have been accompanied by an increase in computational complexity [57]. Designers have proposed special-purpose accelerators [3], [10] to improve the time and energy required to execute DNNs. These accelerators are designed to efficiently perform matrix-vector multiplication (MVM) operations, the core computational kernel in DNNs. Despite their extensive use of on-chip scratchpads to exploit data re-use, these accelerators are eventually still limited by the off-chip memory bottleneck [58], motivating closer integration of memory and processing units as the next step in the quest for further efficiency. Taking this concept to its logical end leads to a new computing paradigm, 'in-memory' computing, wherein memory arrays that store data are also capable of performing compute operations. Although 'in-memory' computing can be explored in CMOS technology [14], [24], fundamental design conflicts in CMOS memories as well as their limited density and slowing scaling trends in CMOS are limiting factors. To that end, non-volatile memory (NVM) technologies such as PCM [59], RRAM [60], [61], Spintronics [62], *etc.* offer immense promise as an alternative to CMOS due to their high storage density and the ability to perform massively parallel in-situ MVM operations. This has led to a growing interest of exploring NVM technology as the substrate for the next generation of ML hardware [19], [63]–[66]. The basic computational fabric of NVM technologies is a two-dimensional cross-point arrangement of NVM devices, commonly known as a crossbar. Due to the intrinsic physics of these NVM devices, they can store multiple states in a highly dense memory with bit-cell area of $\sim 4F^2$ [67]. Further, owing to the resistive nature of NVM devices, crossbars can be used to perform in-situ MVM operations by programming a matrix of conductances into the 2D array of NVM devices

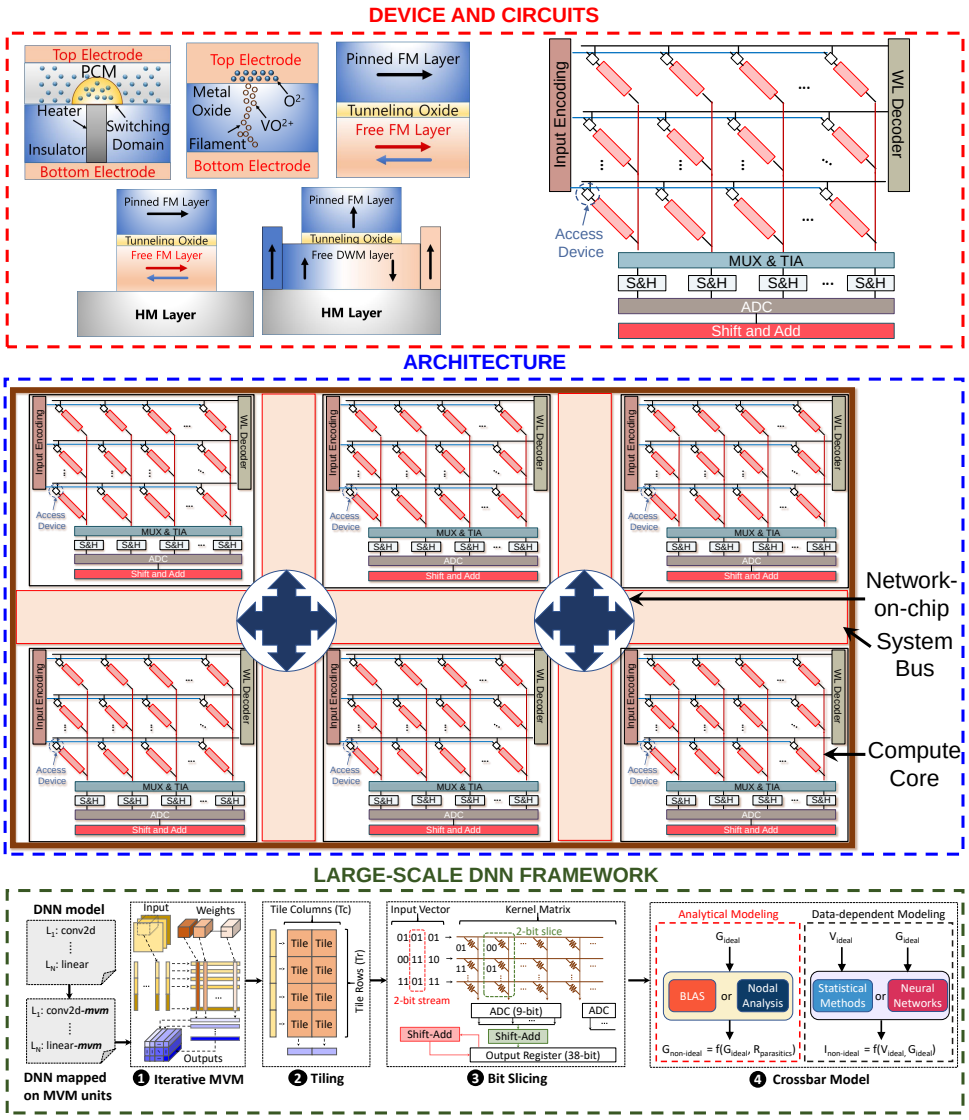


Figure 3.1. Layers of abstractions of resistive crossbar systems for DNN accelerators. First, the basic building blocks of such systems are the NVM devices based on technologies such as PCM, RRAM and Spintronics. Crossbar array with NVM devices, is augmented with peripheral circuits to enable MVM computations. Such crossbars can be used to design large-scale hardware accelerators. Finally, software frameworks allow workloads to be mapped to such hardware fabrics and evaluated considering the impact of non-idealities.

that constitute the crossbar, applying a vector of input voltages to the rows (columns) of the crossbar, and reading out the currents from the columns (rows) of the crossbar. This mode of computation uses Ohm's law and Kirchoff's laws of current summation to natively realize massively parallel analog MVM operations inside memory, and can achieve significantly higher energy efficiency over digital accelerators by eliminating sequential memory accesses. Further, the high density of resistive crossbars can be leveraged to achieve large on-chip

storage, thus avoiding frequent data transfer between off-chip memory and the processor. This promise has fueled great interest over the past decade in designing systems that can leverage the high compute efficiency as well as the high density storage of crossbar-based MVM units [15], [16], [68].

Despite the promise of resistive crossbars, their analog nature of computing poses two primary challenges to large-scale system design: i) Functional inaccuracies due to device and circuit non-idealities, and ii) Overheads due to peripheral circuitry.

Non-idealities signify effects that cause a deviation from ideal MVM computations in resistive crossbars. Undesirable device characteristics and circuit behavior can manifest themselves in three kinds of non-idealities: a) Linear non-idealities, b) Non-linear non-idealities, and c) Random device and transistor variations. Linear non-idealities arise from parasitic resistances in metal lines, source resistances in input drivers as well as sink resistances in sensing circuits. On the other hand, non-linear non-idealities result from read and write non-linearities in the NVM devices and the access device characteristics. Moreover, as the NVM process is in its nascent stage of development; device variations are a major challenge. Such variations can appear in the form of intra-crossbar and inter-crossbar variations. The cumulative effect of the aforementioned non-idealities can cause functional errors at the MVM unit level. In the context of large-scale systems, these errors can accumulate and degrade the application accuracy significantly [22], [69].

The outputs produced by analog MVM units need to be communicated to other such units for the evaluation of large-scale neural networks. As analog signals are less tolerant to noise, global communication in any large-scale system needs to be digital. As a result, MVM units need peripheral circuits such as Digital to Analog Converters (DAC) at their inputs and Analog to Digital Converters (ADCs) at their outputs. These peripheral circuits within MVM units, particularly the ADCs, have been shown to account for a large fraction (up to $\sim 80\%$) of the total energy consumption of the MVM unit [70]. In addition, ADCs consume close to 70% of the area of the MVM unit. Thus, the power and area of peripheral circuits significantly reduces the benefits of the crossbars in terms of compute efficiency and high density.

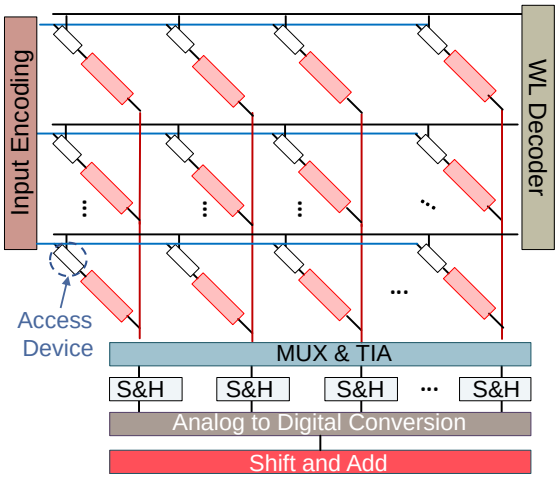


Figure 3.2. resistive crossbar and its peripherals for large-scale integration. Input Encoding circuits apply analog voltages to the rows and Word-line (WL) decode activates the access devices. At the output of the crossbar, muxing network and transimpedance amplifiers (TIA) converts analog current to voltage, which is then passed through Sample & Hold (S&H) circuits. The Analog to Digital Converter (ADC) is shared across the columns. The ADC outputs go through shift and add circuits to account for bit-slicing and bit-streaming.

In summary, in-memory computing using resistive crossbars present great promise but also poses several challenges that need to be analyzed and addressed.

3.2 The Anatomy of Resistive Crossbars

Let us first introduce the concept of crossbar organization. Figure 3.3 shows the conceptual NVM-based crossbar structure, where NVM devices are connected as a matrix similar to a standard memory array. In such resistive crossbars, each memory cell is connected to its corresponding word-line (WL) and bit-line (BL). In principle, MVM operations can be efficiently achieved in an resistive crossbar using Ohm's law for multiplication and Kirchhoff 's current law (KCL) for summation. The MVM vector operands are applied as analog voltages to WLs of the crossbar. Thus, the resulting current passing through a memory cell is the multiplication of the applied voltage (V) at the WL connected to that cell and the NVM device conductance (G). Further, KCL sums up the currents from all cells connected to the same BL resulting in the multiply and accumulate (MAC) operation of V and G.

Such inherent MAC computations in crossbars enable parallel MVM operations where the inputs are V and G while the MAC ouput is represented as the current passing at BL (I_{BL}). In the following subsections, we discuss in details the basic building blocks of an resistive crossbar including the NVM cell structure based on various device technologies, selectors, and, peripherals required for analog computing.

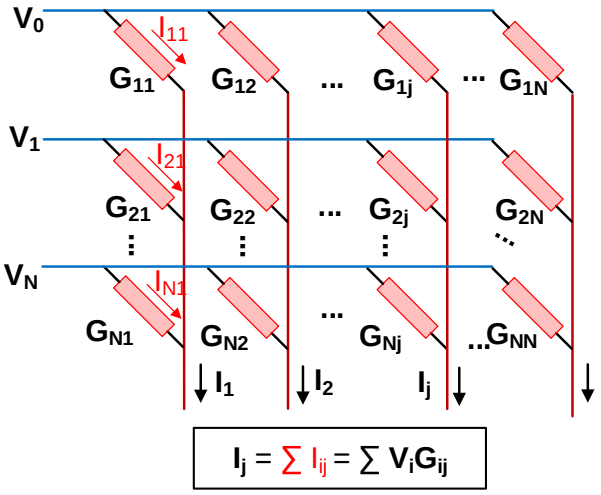


Figure 3.3. Crossbar arrangement of NVM devices to enable matrix-vector multiplication operations. A voltage vector, $V_0, V_1, ..., V_N$ is applied to each row of the resistive crossbar where the conductance of the devices, $G_{11}, ..., G_{NN}$, form the 2-D matrix. The current output vector $I_1, I_2, ..., I_N$ in the column represents the MVM result.

3.2.1 Device Technologies

The basic building block of the MVM compute macro based on crossbars are the NVM devices. However, CMOS memories have also been explored to build this macro which we will discuss later. NVM devices are mostly two-terminal devices based on Phase Change Materials (e.g. PCRAM) [71], Oxide-based materials (e.g. RRAMs) [72] or magnetic materials (e.g. Spin Transfer Torque (STT) - MRAM) [73]. On the other hand, some spintronic devices can also be three-terminal devices such as Spin Orbit Torque (SOT) based devices [74] which have decoupled read and write paths. In this section, we will discuss how the inherent physics of these devices enable multi-bit storage and in-situ multiplication operations.

Two Terminal Devices

Two terminal devices usually consist of a layer sandwiched between two electrodes. This layer is a phase change material in case of a PCM device, an oxide-based material in case of a RRAM device, and stack of ferromagnetic layers in case of STT-MRAM.

a. Phase-Change Materials: PCMs are chalcogenides which have large contrast in electrical and optical properties resulting in a significantly high ON/OFF resistance or reflectance ratio. The device structure follows a mushroom cell structure representing the shape of the switching volume above the heater, shown in Figure. 3.4. These materials can exist in multiple intermediate states between two extreme resistance states, amorphous (high-resistance) and crystalline (low-resistance) states. Progressive crystallization reduces amorphous thickness gradually and can result in multiple programmable states in PCM devices, which makes it suitable for MVM operations.

The most widely used phase change material is $Ge_2Sb_2Te_5$, which provides optimal trade-off between switching speeds and write power [75]. The advantages of PCMs as the building block of MVM units, are their scalability and high density [76]. Despite the promising avenues of PCM crossbars, there are several challenges of the technology that still remains to be addressed. First, the resistance of PCM devices in the amorphous state drifts over time due to structural relaxations after the melt-quench amporphization process. This can adversely affect the MVM functionality of PCM crossbars. Second, PCM devices have high switching times ($\sim 100ns$) [71] and suffer from low endurance ($\sim 10^7 cycles$) [77] which makes write operations in PCM crossbars costly.

b. Metal-Oxide Materials and CBRAMs: Perovskite oxides such as $SrTiO_3$ and binary metal oxides such as HfOx, TiOx, TaOx, *etc* exhibit resistive switching properties [78]–[81]. Two-terminal devices based on these materials form the basic building blocks of RRAM based crossbars. Another class of devices formed using the same structure by simply replacing the oxides by conductive elements, constitute Conductive Bridge RAMs or CBRAMs [82]. RRAM and CBRAM devices can behave as multi-level memory with two extreme resistance states. The multi-level behavior is owing to soft dielectric breakdown in metal-oxide RRAMs and dissolution of metal ions in CBRAMs. Although switching mechanisms in CBRAMs are

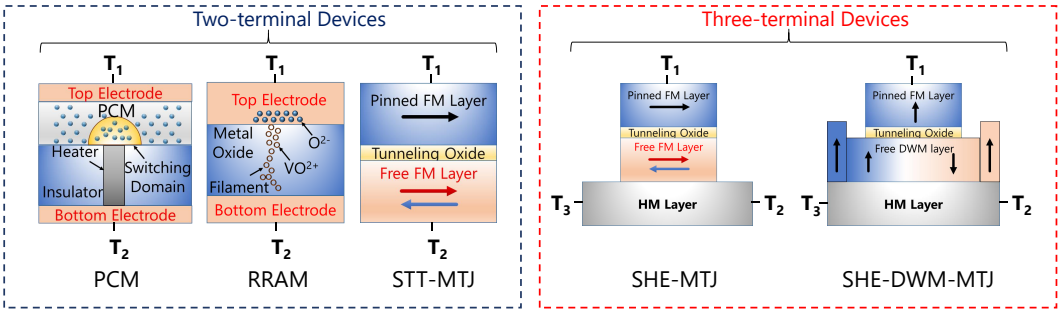


Figure 3.4. Different NVM device technologies can be broadly categorized into two types: Two terminal Devices and Three terminal Devices. Two terminal devices include Phase Change Material (PCM), metal oxide Resistive Random access memory (RRAM) and Spin Transfer Torque Magnetic Tunnel Junction (STT-MTJ). Three terminal devices include Spin Hall Effect MTJ (SHE-MTJ) and SHE domain wall magnet MTJ (SHE-DWM-MTJ).

primarily filamentary, metal-oxide RRAMs can exhibit both filamentary and non-filamentary switching. Thermal redox reactions between metal electrodes and oxide material causes formation and rupture of filamentary conductive paths, resulting in filamentary switching. Through this switching, RRAM devices can be programmed to multiple states either by modulating the number of conductive filaments using current compliance or by controlling the degree of oxidation [83]. On the other hand, non-filamentary switching occur in oxides of some transition metals such as $PrCa_xPCMO$, through modulation of Schottky barrier by charge-trapping or defect migration at the metal-oxide interface [84]. Switching in CBRAMs is analogous to RRAM filamentary switching except that the filament is a metallic conductive path instead of oxygen vacancies. Typical CBRAMs are based on Ag electrodes where the multi-level behavior is achieved by tuning the volume of Ag-rich and Ag-poor regions [85]. The multi-level memory behavior of RRAMs and CBRAMs is suitable for the aforementioned MVM macro using the crossbar arrangement of these devices.

c. Spintronic Devices: Unlike PCMs and RRAMs, spintronic devices use electron spin as the storage variable. Spin devices are particularly promising with respect to other NVM technologies due to the extremely high endurance ($\sim 10^{15} cycles$) [86] and fast switching ($\sim 10ns$) [87]. Two terminal spin devices constitute of a structure known as Magnetic Tunnel Junctions (MTJs) which is basically two ferromagnetic (FM) layers sandwiching a spacer layer (commonly MgO). These ferromagnets encode information in form of the

direction of magnetization and can achieve two stable opposing directions. The direction of the two FMs can either be parallel (P) or anti-parallel (AP), which result in two extreme resistance states in these devices. The MTJ can be switched from AP to P or vice-versa by passing a spin polarized current, which initiates a phenomenon called Spin Transfer Torque (STT) [88]. STT-MTJs can typically attain only two stable states, unlike PCMs or RRAMs, which limit the density of the crossbars based on them. Moreover, writing into STT-MTJs can require high amounts of current compared to RRAM or PCM devices. Although STT-MTJs have been experimentally demonstrated at a large-scale for storage class memories [89], their applicability as MVM macros still is at a nascent stage of development. This is primarily due to the low resistance ratio between the P and AP states, commonly quantified as the Tunneling Magnetoresistance (TMR). Although TMR of 600% have been achieved experimentally for MTJs [90] and theoretical models predicting that 1000% can be achievable [91], [92], large-scale demonstrations are still lacking.

Two terminal devices achieve high density and possess simpler fabrication process. However, due to merged read and write paths, writing into these devices may cause sneak path issues [93]. To address such issue, researchers have proposed using access devices such as transistors or selectors. An alternative way to circumventing this problem is using three-terminal devices with decoupled read and write paths.

Three Terminal Devices

Typically, three terminal devices are primarily based on spin devices. By stacking an MTJ on top of a heavy metal (HM) layer, one can write into the FM layer by passing current through the HM using a phenomenon called the Spin Hall Effect (SHE) [94]. SHE-MTJs can achieve higher spin injection efficency than STT-MTJs, thus achieving lower write currents. Moreover, the decoupled read and write paths can address sneak path issues prevalent in crossbars with two terminal devices. SHE-MTJs have also been explored for achieving multiple states using bigger domain-wall magnets (DWM) [74]. Here, the position of the domain wall in the DWM can be controlled by passing appropriate current through the HM layer. The position of the domain wall governs the conductance of the SHE-MTJ.

Much like STT-MTJs, SHE-MTJs also suffer from significantly low TMR. Thus, although three terminal SHE-MTJs offer immense promise in terms of energy efficiency and speed, studies have mostly been limited to simulations [62]. To realize the potential of spin-based crossbars, material exploration to achieve higher TMR is important.

Having elucidated on different NVM device technologies that have been explored for crossbars, it is important to understand the key device parameters that play a key role in large-scale crossbar designs. The scalability of resistive crossbars as a MVM macro is primarily dependent on:

- Resistance ON/OFF ratio: The ratio between the resistances of the two extreme states in the device.
- ON Resistance: The low-resistance state of the device.
- Bit precision: Number of states that can be reliably programmed and read from the device.

These parameters affect the resistive crossbar size, thus playing a role in both the MVM functionality as well as energy and latency.

3.2.2 Circuits: Peripherals and Access

The aforementioned MVM unit based on resistive crossbars needs to be supplemented by peripheral circuits and access devices for suitable interfacing between the analog compute unit and digital communication and data flow. Let us first consider various device options that provide efficient data access in a memristive cell including two-terminal selectors and three-terminal selectors (transistors). The access devices are used to avoid sneak path issues in resistive crossbars during write operation. Next, we consider several peripheral circuitry needed for MVM operations based on the type of input encoding.

Selectors/Access transistors

As mentioned earlier, all cells in a crossbar are activated to perform the dot product operation. Therefore, sneak-path currents induced from partial select operation are avoided

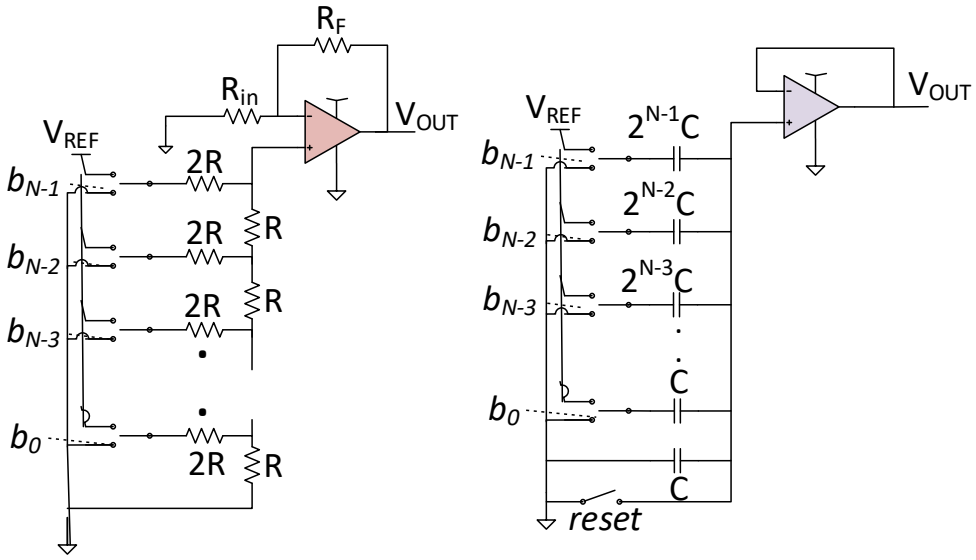


Figure 3.5. (a) R-2R ladder based DAC. (b) Capacitance-based DAC [102]

during dot-product computing. However, such sneak-path current issue becomes significant during normal read/write operations where only one row is selected and the unselected rows should be simultaneously biased to GND. To that end, selector devices and access transistors have been adopted in practical implementations of memristive crossbars to eliminate the effects of the sneak path problem [95]–[101].

Two dimensional selector devices are more dense and suitable for 3D integration than a transistor, thus one-selector–one-memristive-device (1S-1R) arrays were reported with different types of selector devices such as [96]–[98], [100], [101]. On the other hand, one-transistor–one-memristive-device (1T-1R) crossbars have exhibited MVM computations with fewer errors[103], [104], in comparison to 1S-1R crossbars. Typically, in an 1T-1R crossbar WL controls the gate of the transistor, while the source-line (SL) is connected to the transistor source. The NVM device top electrode is connected to BL, while its bottom electrode is connected to the access transistor drain. One disadvantage in 1T-1R structure is that the cell area is determined by the transistor area, depending on the maximum programming current required to pass across the NVM device. Nevertheless, 1T-1R structure is a standard practice for constructing resistive crossbars as MVM units. Next, let us discuss in details

various peripheral circuit design choices needed for efficient MVM computing in memristive crossbars.

Input Encoding

In MVM operations of ML workloads, the vector represents the inputs of the neural networks. These input vectors can be encoded in voltage or time in resistive crossbars. WL DACs can be used to convert the digital inputs to analog voltages to be applied at every row. Different variants of DAC utilize charge, current, and voltage multiplication or division using switched-capacitor circuits, current steering circuits, and resistor ladders, respectively, to generate the analog output [102], [105]. Figure 3.5 shows schematic of the aforementioned DAC structures.

Output Sensing

The MVM output current (I_{BL}) at each crossbar column needs to be converted to the digital domain for transferring data to the next compute unit in the system. To that end, I_{BL} at each column is converted to analog voltage, then to digital through current to voltage converter and analog-to-digital converter (ADC), respectively. Figure 3.2 shows the crossbar structure with peripheral circuitry including input encoding and sensing.

Current-to-Voltage Conversion: After performing the MVM operations in memristive crossbars, the resulting output current at BLs (I_{BL}) is converted to analog voltage through a current-to-voltage converter. A common circuit technique to convert analog current to voltage is the trans-impedance amplifier (TIA) [106]. In its simplest form, a TIA can be just a single grounded resistance as shown in Figure 3.6 (a). Further, a more sophisticated TIA can be an amplifier having an open-loop gain of $-A_0$ with a negative feedback resistance R_F placed around the amplifier as shown in Figure 3.6 (b). The fundamental difference between single-resistance and amplifier-based TIA is that I_{BL} sees a high impedance in Figure 3.6 (a) and a low impedance in Figure 3.6 (b).

Analog-to-Digital Conversion: Analog to Digital conversion is a critical step in the MVM operation using resistive crossbars, as the analog voltages need to be converted to

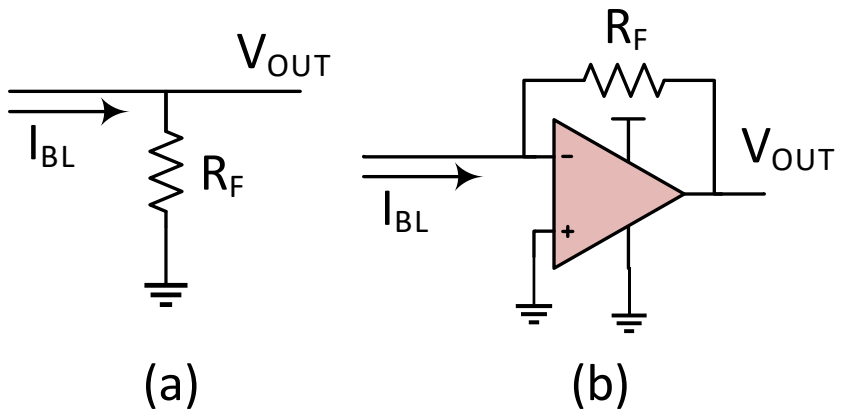


Figure 3.6. (a) The resistor-based current-to-voltage converter. (b) The op-amp based TIA [106].

digital output for further processing. To that effect, various ADC architectures have been adopted in memristive crossbars including Successive Approximation Register (SAR) based ADC and Flash ADC [107], [108]. Typically, Flash ADCs employ parallel sampling to achieve fast conversion, an n-bit Flash ADC consists of 2^n comparators, a resistor ladder, and a decoder. Due to the large number of comparators, the energy consumption of a Flash ADC is much larger than same-precision SAR ADC [109].

SAR ADC is the most common ADC architecture adopted in analog in-memory computing memristive crossbars due to its simplicity and low area/energy overhead compared to other ADC architectures [109]. Various ADC architectures and precisions have been adopted in memristive crossbar MVM units [15], [16], [103]. For instance, ISAAC [16], and PUMA [15] adopted an 8-bit SAR ADC for higher precision computations, while XNOR-RRAM implemented a 3-bit Flash ADC for binary input/weight computation [110]. Note, ADC area and energy costs are huge and dominate the whole crossbar area and energy [15]. Typically, as reported in [70] ADC consumes 80.61% and 68.13% of the crossbar power and area, respectively. Therefore, ADCs are shared across the columns, and analog multiplexers (MUX) and sample and hold (SnH) blocks are adopted to allow the time-multiplexed ADC usage for further energy efficiency. On the other hand, reducing the ADC precision further reduces its area and energy.

It is worth mentioning that the ADC precision required to compute exact MVM operation in an NxN crossbar having K input precision and M weight precision should be $ceil(\log_2((2^K - 1) * (2^M - 1) * N))$. To conclude, peripheral design choices like input encoding, input/weight precision, ADC architecture, and precision are key factors to improve energy consumption, area, and performance at the crossbar and system levels.

3.2.3 Crossbar Write Operations

The operation of resistive crossbars for memory applications uses the more conventional read-verify-write mechanism for reliable programming of devices. However, due to the serial nature of this update mechanism and high write energy of NVM technologies, the cost of *in-situ* training of large-scale DNNs using resistive crossbars become significantly high. As a solution, researchers have proposed parallel updating scheme [116], particularly for ML workloads, which leverage the analog mechanism of programming to mimic the outer-product operation commonly used in DNN training algorithms. Figure 3.8 shows such a parallel writing scheme, where the row inputs, x_i, are encoded by the pulse-width and the column inputs, y_j, encoded as pulse amplitude. Simultaneous application of pulses at the

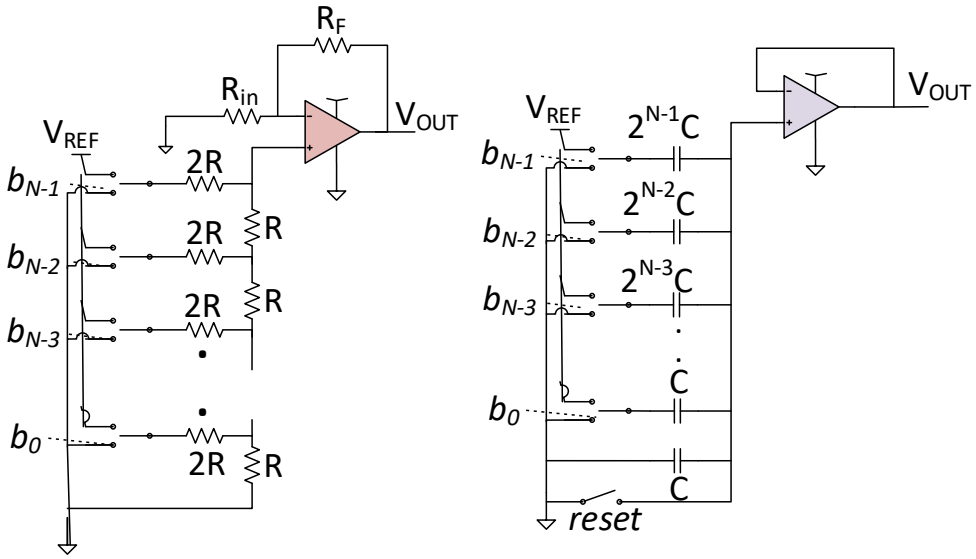


Figure 3.7. (a) Flash ADC architecture. (b) SAR ADC architecture [107], [108].

Table 3.1. Silicon Demonstrations

Category	[60]	[59]	[111]	[112]	[113]	[110]	[114]	[103]	[115]
Technology Node (nm)	2 μ m	180 nm	180 nm	130 nm	130 nm	65 nm	65 nm	55 nm	22 nm
Device Type	RRAM	PCM	RRAM	RRAM	RRAM	RRAM	RRAM	RRAM	RRAM
Crossbar Size	128x64	512x1024	64x64	128x16	784x100	128x128	256x64	256x512	512x512
Input Precision	4b	NA	1b	1b	1b	1b	1b	1b, 2b	1b, 2b, 4b
Weight Precision	2b	NA	8b	5b	3b	1b	4b	3b	2b, 4b, 4b
MAC output	4b	NA	1b	8b	8b	3b	8b	4b	6b, 10b, 11b
Accuracy (MNIST)	99%	97.95%	90.8%	NA	94.4%	98.43%	NA	98.8%	NA
Accuracy (CIFAR-10)	NA	88.29%	NA	94%	NA	86.08%	NA	88.52%	90.18%
Accuracy (CIFAR-100)	NA	67.96%	NA	NA	NA	NA	NA	NA	66.46%
Accuracy (ImageNet)	NA	NA	NA	NA	NA	NA	69%	NA	NA
Energy Efficiency[1] (TOP/s/W)	115	28.065 [2]	20.7	11	78.4	141.18	5.9	53.17	121.38 (6b) 45.52 (10b) 28.93 (11b)

[1] TOP/s/W values are reported with respect to the corresponding technology node and the output precision.
[2] scaled to 14 nm.

rows and column produces a multiplicative effect such that a higher voltage at the column and longer pulse at the row results in a bigger change in conductance of the NVM devices at the crosspoint. The result, of course, is the update of the weights, expressed as:

$$W_{ij} = W_{ij} + x_i \times y_j \tag{3.1}$$

Weights in crossbar are stored in unsigned form. To account for both positive and negative weight updates, the inputs are represented in sign magnitude form. The polarity of weights can be represented using two separate devices for positive and negative weights. To represent both positive and negative weights in the same device, the range of weight, R_{ON} to R_{OFF}, is symmetrically centered around $(R_{ON}+R_{OFF})/2$ [117]. This means a zero weight is represented by $(R_{ON}+R_{OFF})/2$. Such sign-magnitude and biased representation of weights can enable both positive and negative updates to the same device.The polarity of the update is controlled by the polarity of the simultaneous pulses [118]. Such parallel writing schemes can be adopted entirely in the time domain as well [119]. Such a scheme has also been experimentally demonstrated on floating gate memory arrays [120].

We have discussed the basic anatomy of resistive crossbars, starting from the building blocks (NVM devices at the crosspoint) to detailed description of peripheral circuits required for proper operation of the crossbar as compute cores in DNN accelerators. The devices and circuits, however, can introduce undesirable behavior that affects the desired MVM

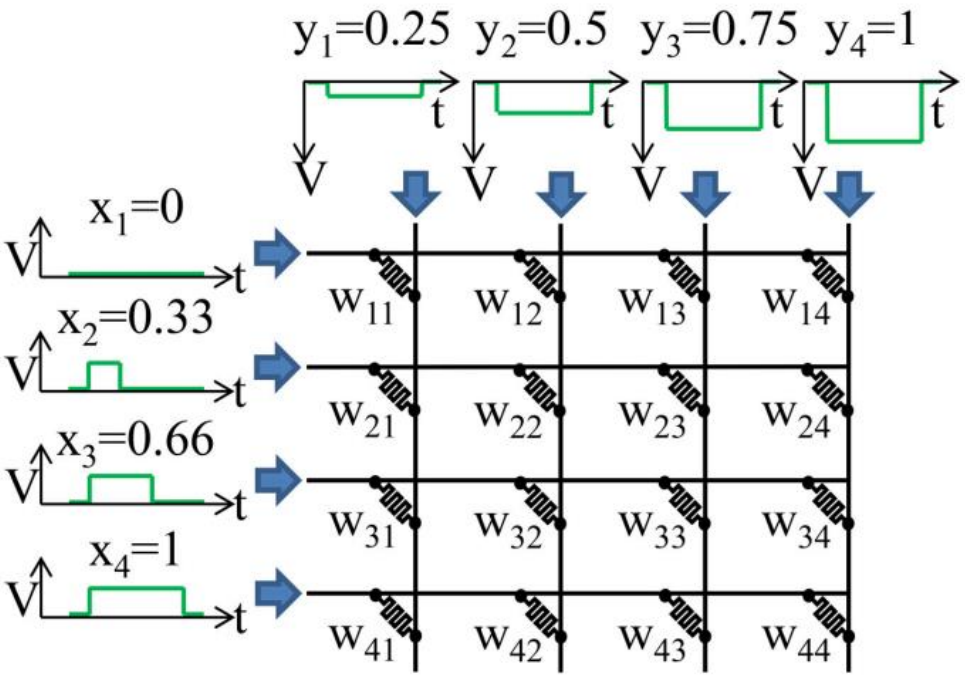


Figure 3.8. Row inputs, x_i, are encoded by the pulse-width and the column inputs, y_j, encoded as pulse amplitude. Simultaneous application of pulses at the rows and column produces a multiplicative effect such that a higher voltage at the column and longer pulse at the row results in a bigger change in conductance of the NVM devices at the crosspoint [116]

functionality of the crossbars. In the next section, we will study the origin of such non-idealities and their impact on the MVM functionality.

3.2.4 Silicon Demonstrations

NVM MVM macros:

Although NVM technologies are in their nascent stage of developments, there are several experimental demonstrations of MVM macros based on these technologies. There has been extensive work on experimental demonstration of PCM [59], [121] and RRAM crossbars [104], [122], [123]. At the time of writing this thesis, experimental demonstrations of Spintronic and FeFET crossbars are yet to be explored in a large-scale. Table 3.1 summarizes the recently-demonstrated memristive crossbars configurations and results.

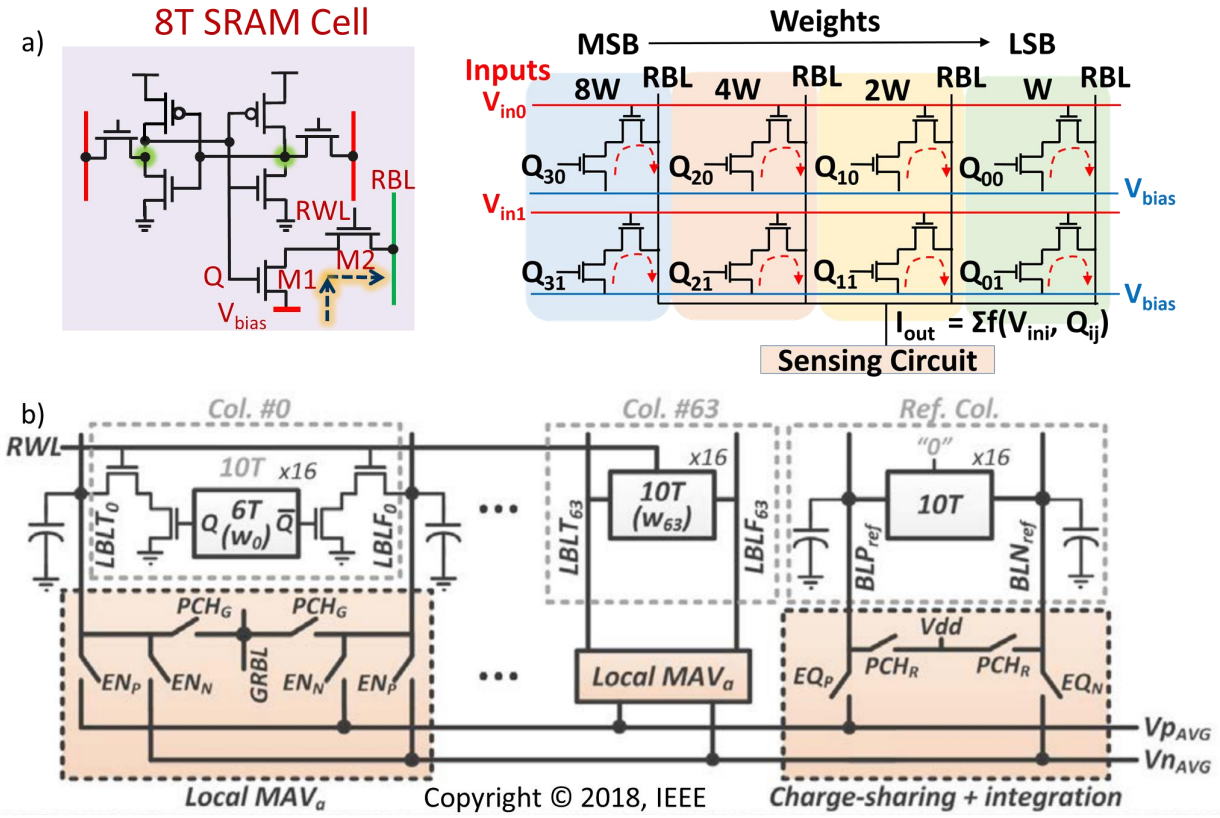


Figure 3.9. (a) MVM macro using current based computations in 8T SRAM cells. The weight bit is represented by node 'Q'. By applying a voltage on RWL, the current in the RBL is a function of the weight bit and the applied voltage. The transistors are sized appropriately to represent multi-bit weights. (b) MVM macro using voltage based computation in 10T SRAM cells. The local bit-lines (LBLT and LBLF) are discharged according to the weight bit stored in the 6T SRAM. Through charge sharing, average MVM output is calculated in the horizontal lines [11].

CMOS MVM macros:

It is worth mentioning that MVM units can also be constructed using CMOS-based SRAM cells **akyel2016drc**, [11], [14], [24]. Figure 3.9 (a) shows an example of such a macro based on an eight-transistor (8T) SRAM array [24]. The bitcell consists of six-transistor (6T) SRAM cell along with a read port constituted by two additional transistors. The node 'Q' stores the weight bit. By applying appropriate voltage on the read word-line (RWL), the current sourced from the source-line (SL) onto the bit-line (BL) becomes a function

of the stored weight bit and the applied voltage. The currents from all the bit-cells in a single column accumulate in the BL, which represent the MVM output, similar to resistive crossbars. To enable multi-bit MVMs, the transistors in the read port can be appropriately in the ratio 8:4:2:1 as shown in Figure 3.9 (a). If the weight $W_{\mathrm{i}}^{\mathrm{j}} = w_3 w_2 w_1 w_0$, where w_{i} are the bits corresponding to the 4-bit weight, the vector matrix dot product becomes:

$$\Sigma(v_{\mathrm{i}} \cdot W_{\mathrm{i}}^{\mathrm{j}}) = \Sigma[v_{\mathrm{i}} \cdot (2^3 w_3 + 2^2 w_2 + 2^1 w_1 + w_0)] \tag{3.2}$$

Alternatively, voltage accumulation can be also used to perform MVM operations through charge-sharing in the BL. Figure 3.9 (b) shows a voltage-based MVM operation in a ten transistor SRAM cell [11]. This bitcell consists of two local BLs, which are discharged according to the weight bit stored in the bit-cell. The local BLs are appropriately shorted to evaluate the average MVM output which is then passed to a charge sharing ADC to obtain the digital output. Due to the inherent non-linearity in transistor characteristics, the linearity of MVM operations in CMOS memories is difficult to realize. Moreover, as SRAM cells can only store single weight bits, CMOS based analog computing MVM macros lose in area efficiency to resistive crossbars for multi-bit MVM operations. As a result, researchers have explored CMOS macros primarily in the context of binary neural networks [18], [39] where the matrix vector multiplications are represented using XNOR operations [110], [124] between binary inputs and weights. Despite several efforts investigating possibilities of MVM computations using CMOS memories, their significantly higher area footprint with respect to resistive crossbars make the latter more promising as basic compute primitives for ML workloads.

4. TECHNOLOGY AWARE TRAINING IN MEMRISTIVE NEUROMORPHIC SYSTEMS FOR NONIDEAL SYNAPTIC CROSSBARS

Recent developments in computational neuroscience have resulted in a paradigm shift away from Boolean computing in sequential von-Neumann architectures as the research community strives to emulate the functionality of the human brain on neurocomputers. Although extensive research has been done to accelerate computational functions such as matrix operations on general-purpose computers, the parallelism of the human brain has remained elusive to von-Neumann architecture, thus engendering high hardware cost and energy consumption[125]. This has resulted in the exploration of non-von Neumann architectures with 'massively parallel operations in-memory', thus avoiding the overhead cost of exchanging data between memory and processor. Especially with the recent advances in machine learning in various cognitive tasks such as image recognition, natural language processing etc, the search for such energy-efficient 'in-memory computing' platforms has become quintessential. Although standardized hardware implementations of neuromorphic systems like $CAVIAR$[126], $IBM\ TrueNorth$[127], $SpiNNaker$[128] have primarily been dominated by CMOS technology, the memristor-based non-volatile memory (NVM) technology[129]–[133] has naturally evolved into an exciting prospect. To that end, various technologies such as spintronics[134], oxide-based memristors [135], [136], phase change materials (PCM) [137], etc., have shown promising progress in mimicking the functionality of the core computational units of a neural network, i.e., neurons and synapses.

The core functionality of a neuromorphic system is a parallelized dot product between the inputs and the synaptic weights[138]. This has been demonstrated to be efficiently realized by a dense resistive crossbar array [139], [140]. The ability to naturally compute matrix multiplications makes crossbar arrays the most convenient way of implementing neuromorphic systems. However, real crossbars could suffer from various non-idealities including device variations[141], [142], parasitic resistances, non-ideal sources, and neuron resistances. Although neural networks are generally robust against small variations in the crossbar, the aforementioned technological constraints can severely impact accuracy of recognition tasks

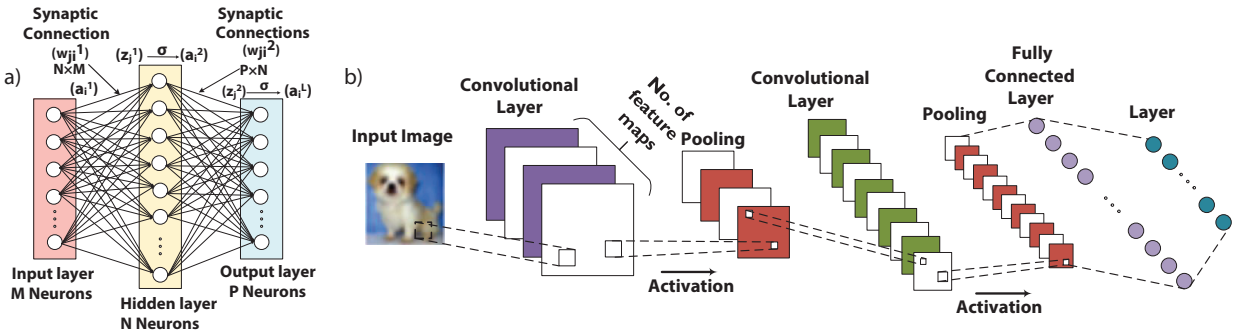


Figure 4.1. (a) Fully connected 3-layered neural network showing the input layer, hidden layer, and an output layer. Each neuron in a particular layer is fed by weighted sum of all inputs of the previous layer and it performs a sigmoid operation on the sum to provide the inputs for the next layer. (b) CNN Architecture with different convolutional and pooling layers terminated by a fully connected layer.

as well as restrict the crossbar size. Several techniques such as redundancy schemes[143], technology optimization[144] and modified training algorithms[145]–[147] have been explored for both on-chip and ex-situ learning to mitigate specific non-ideal effects such as IR drops, synaptic device variations. However, mathematical modeling of non-idealities and its incorporation in standard training algorithm needs further exploration.

In this chapter, we analyze the impact of non-idealities such as source resistance, neuron resistances, and synaptic weight variations in hardware implementations of neuromorphic crossbars. We show how such non-idealities can significantly degrade the accuracy when traditional training methodologies are employed. The presence of these parasitic elements also severely limits the crossbar sizes. As a solution, we propose an ex-situ technology aware training algorithm that mathematically models the aforementioned non-idealities and accounts for the same in the traditional backpropagation algorithm. Such a technique not only preserves the accuracy of an ideal network appreciably but also allows us to use larger crossbar sizes without significant accuracy degradation. The key highlights of our work are as follows:

1. We mathematically model the effect of source resistance, neuron resistance, and variations in synaptic conductance on the output currents of a neuromorphic crossbar. We

establish the validity of our model by comparing against SPICE-like simulations of resistive networks.

2. We analyze the impact of these non-idealities on the accuracy of two types of image recognition tasks with varying amounts of non-ideality within relevant technological limits.

3. We propose a training algorithm which incorporates the mathematical models of the crossbar non-idealities and modifies the standard training algorithm in an effort to restore the ideal accuracy.

4.1 Crossbar Implementation of Neural Networks

4.1.1 Types of network topologies

Fully Connected Networks

Traditionally, deep neural networks such as deep belief nets (DBNs) comprise of multiple layers of interconnected units. Fully connected networks (FCN) involve a series of neuron layers between the input and the output layers. The output of each neuron in a layer is connected to the inputs of all the neurons in the subsequent layer. Fig. 6.5(a) shows a 3-layered fully connected network consisting of a single hidden layer between the input and output layers.

Convolutional Networks

Complex image recognition datasets comprise of objectively different classes where global weight mapping like FCNs prove to be less efficient. As an alternative, convolutional neural networks (CNN) have been recognized as a more powerful tool for complex image recognition problems using locally shared weights to learn common spatially local features. As shown in Fig. 6.5(b), CNNs consist of several layers performing operations like convolution, activation, and pooling, finally terminating with a fully connected layer. The convolution function can be mathematically represented by a 4 dimension tensor. Intuitively, a convolution layer is composed of a number of filter banks. The number of filter banks is equal to the number of

output maps. Each output map represents a feature. A filter bank is made up of multiple kernels, one for each input map. Hence each filter bank operates on all the input maps to extract one output feature map. A kernel is mathematically represented as a $n \times n$ weight matrix. During convolution, the kernels of a filter bank are convolved with their respective input maps. The outputs of these convolutions are then summed together to form the corresponding output map of that filter bank. Thus a convolution operation captures the spatially local features of an input image. Convolution of a $m \times m$ input map with a kernel of size $n \times n$ yields an output map of size $((m-n+2p)/s+1) \times ((m-n+2p)/s+1)$, where s is the stride of the filter and p is the padding. In practice, s and p are chosen such that the original input size is preserved. The activation layer which can be 'RELU' [148], 'sigmoid'[149], or other non-linear functions, introduces a non-linearity in the network[150]. The pooling layer reduces the dimensionality of the output map. Most commonly used pooling techniques are average and max-pooling[151]. Finally, the fully connected layer uses the learned features to classify the images. In essence, a fully connected layer could also be represented by a convolutional layer where the kernel size is equal to the input size.

4.1.2 Hardware representations of Neural networks

In hardware realizations of neural networks based on the non-Von Neumann architecture framework, the synaptic connections between the neurons of two adjacent layers are represented using a resistive crossbar. The weights are represented in terms of conductance and the inputs are encoded as voltages. Convolutional layers have locally concentrated connections, hence each filter bank is represented by a crossbar of equivalent size. The input to the crossbar is a subset of the image being sampled by the kernel. Each element of the output map is calculated through time multiplexing of the outputs from a particular crossbar for different subsets of the image. This is repeated for each filter bank to obtain different output maps. In contrast, fully connected layers have all possible connections between input and the output and the entire connection matrix can be represented by a crossbar. The basic computational function of any layer is a dot product and can be seamlessly performed by

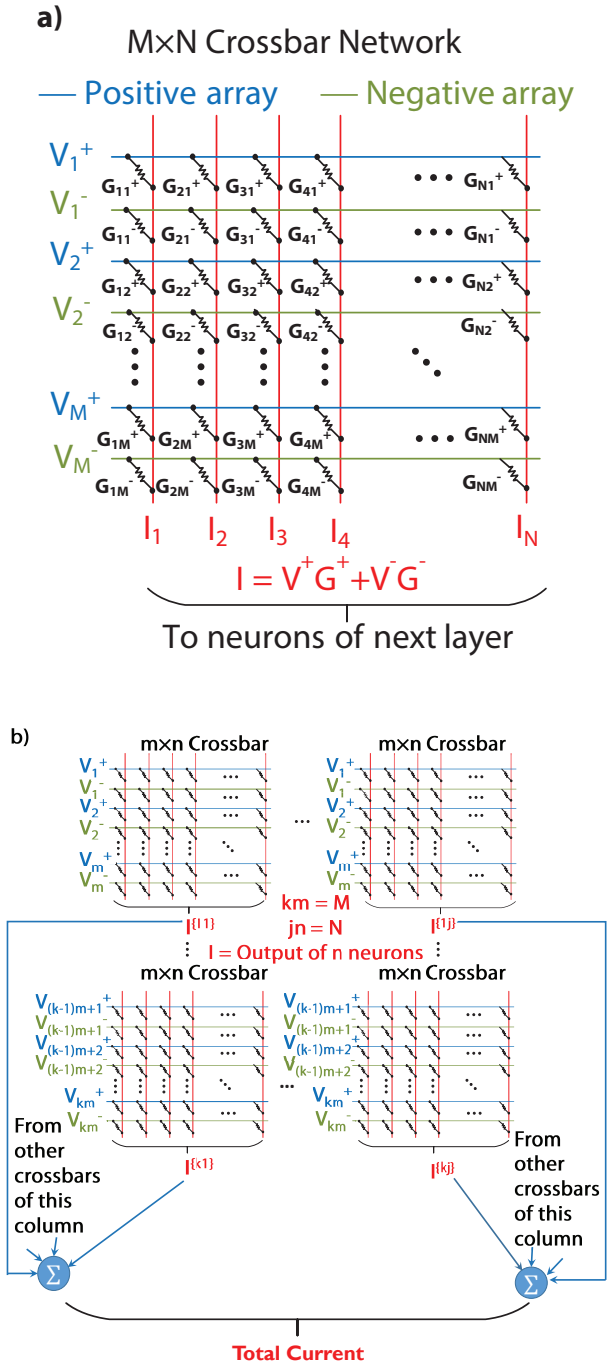


Figure 4.2. (a) Hardware implementation of a single fully connected network layer represented by two resistive crossbar arrays. The output of the crossbar will be fed to another crossbar representing the next layer. (b) An arrangement of multiple sub-crossbars to realize the functionality of a large crossbar.

representing the weights as the resistances in a crossbar fashion. The output current of j^{th} neuron of each crossbar is computed as

$$I_j = \sum V_i^+ G_{ji}^+ + V_i^- G_{ji}^- \tag{4.1}$$

where V_i is the input voltage corresponding to i^{th} input neuron and G_{ji} represents the conductance corresponding to the synaptic weights between the neurons. Two resistive arrays are deployed to account for bipolar weights. The input to the positive array is $+V_i$ whereas the input to the negative array is $-V_i$. The weight matrix $[w_{ji}]$ is mapped to a corresponding conductance range $[G_{low}, G_{high}] \subset [G_{on}, G_{off}]$. To represent bipolar weights, the conductance of the synapse connecting the j^{th} neuron in the next layer to the i^{th} input is denoted by a positive (G_{ji}^+) component and a negative (G_{ji}^-) component. For positive (negative) weights, the programming is done such that $G_{ji}^+(G_{ji}^-) = |w_{ji}|G_{high}$ and $G_{ji}^-(G_{ji}^+) = 0$ (no connection). Fig. 4.2(a) shows a crossbar implementation of a fully connected neural network.

As mentioned earlier, crossbar arrays could suffer from non-ideal effects and incur limitations on their sizes. As a result, larger crossbars are divided into smaller crossbars and the output of each crossbar is time-multiplexed to obtain the desired functionality of the entire crossbar. Fig. 4.2(b) shows how multiple small crossbars can be efficiently mapped to realize the functionality of a large crossbar in a particular layer. The small size of the crossbar reduces fan-out and fan-in, thus minimizing the impact of non-idealities. FCNs, being densely connected, are severely affected by hardware imperfections, especially when implemented on large crossbars. Convolutional layers in CNNs are usually implemented on very small crossbars and are thus insensitive to non-ideal effects. However, the final fully connected layers which acts as a classifier can be significantly affected by these non-idealities due to their large sizes. In this chapter, we are thus considering the impact of non-idealities on FCNs, and fully connected layers of CNNs.

4.1.3 Training

The training of Artificial Neural Networks (ANN) are traditionally done off-chip through the standard backpropagation algorithm which updates weight matrices using gradient de-

scent technique[152]. It is important to note down the vital aspects of the algorithm here in relevance to the later sections. The basic algorithm updates weights based on the gradients of a cost function. The cost function depends on the error computed from the feed-forward network which assumes a form : $C = \frac{1}{2}\sum(y_{\mathrm{j}} - a_{\mathrm{j}})^2$, where y_{j} is the expected output and a_{j} is actual output from the j^{th} neuron in the output layer. The sensitivity of the errors for each layer are calculated from the derivatives of the cost function with respect to the outputs and weights and after each iteration, the weights are updated based on those of the corresponding layer. The detailed description of the algorithm is well documented[152]. In this chapter, we focus on the aspects of the algorithm pertinent to fully connected layers and we build mathematical models to account for the non-idealities experienced by the hardware implementation of neuromorphic crossbars.

4.1.4 Technologies

Various technologies have been explored for crossbar implementations of neural networks. Memristive crossbars based on different material systems (like TaO_x[153], TiO_2[154], Ag/Si[155] etc) have been proposed to realize neuromorphic functionality in an energy efficient manner. Phase change materials (PCM)[137] have also been investigated as potential candidates for neuromorphic computing due to their high scalability. More recently, neurons and synapses implemented with spintronic devices[134], [140] have shown great promise in performing ultra-low power neuromorphic computing. However, each technology suffers from specific drawbacks. An important metric in regard of resistive crossbars for neuromorphic systems is the ratio of the high resistance state (R_{off}) and the low resistance state (R_{on}) of the synaptic device. Usually, a high R_{off}/R_{on} ratio is desired for a near-ideal implementation of the weights in a neuromorphic crossbar. Moreover, in the light of non-ideal systems, higher values of R_{on}, R_{off} may be less significantly impacted by parasitic resistances. In this chapter, we have chosen a maximum to minimum conductance ($G_{low} = \alpha/R_{off}$, $G_{high} = 15G_{low}$, α is a parameter of choice) ratio of 15 which is a potentially realizable predictive measure for all memory technologies[137], [156], [157].

Table 4.1. Resistance Ranges for Various technologies

Technology	$[R_{on}, R_{off}]$	Considered Range $[R_{low}, R_{high}]$	R_s/R_{high} %	R_{neu}/R_{high} %
TiO_2[159]	15k, 2M	40k,600k	0.033 - 0.13	0 - 0.033
Ag/Si [142]	25k,10M	100k,1.5M	0.013 - 0.053	0 - 0.013
TaOx [160]	1k,1M	20k,300k	0.067 - 0.27	0 - 0.067
Spintronics*	Function of MTJ oxide thickness [161]	40k, 400k	0.05 - 0.2	0-0.05
PCM [137]	10k, 3M	60k, 900k	0.022 - 0.08	0 - 0.022

*The spintronic analysis is done based on predictive measures of R_{off}/R_{on}[156]
R_s range - 200 to 800 Ω, R_{neu} range - 0 to 200 Ω

Memristor based neuromorphic crossbar designs leverages its inherent capability of matrix multiplication to provide high accuracy at a relatively modest computational cost[158]. However, the memristor technology is still in its nascent stage. Thus, the hardware implementation of such crossbars may suffer various kinds of non-ideal effects arising from memristor device variations, parasitic resistances as well as non-idealities in sources, and sensing neurons.

4.2 Modeling the non-idealities

In this chapter, we have considered three kinds of non-idealities that arise in crossbar implementations, namely,

1. Neuron Resistance (R_{neu})
2. Source Resistance (R_s)
3. Memristive resistance variations

To perform an analysis of the impact non-idealities might have on accuracy of recognition task, it is important to note the ratio of non-ideal resistances to the synaptic resistances for a particular technology. Table. 4.1 shows the range of considered resistance ratios to

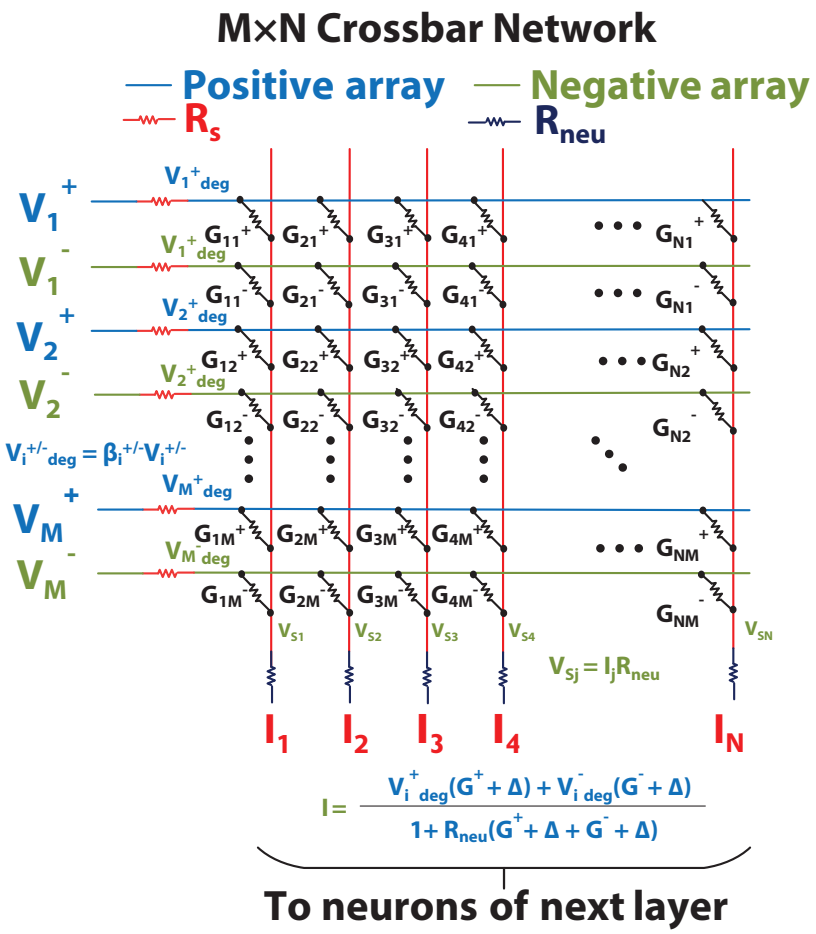


Figure 4.3. Crossbar Architecture showing non-ideal elements like source and neuron resistances. The final output current equation is modified by the impact of these non-ideal elements.

synaptic resistances R_s/R_{high} and R_{neu}/R_{high} for various technologies, considering relevant values of source (R_s) and neuron (R_{neu}) resistances.

4.2.1 Neuron Resistance

The resistance offered by the neuron in a neuromorphic crossbar varies from technology to technology. In many cases, such as, PCM technology, the resistance of the neuron is not a hardware issue as the crossbar outputs are sensed through a sense amplifier, where virtual ground at the input eliminates the voltage drop across the neuron. However, in spintronic crossbars[134], crossbar outputs are fed to the neuron as a current stimulus and thus, the

resistance of the neuronal device becomes relevant. Fig. 4.3 shows the effect of neuron resistance on the crossbar output. This can be mathematically modeled to modify Eqn (1) as:

$$I_{\mathrm{j}} = \frac{\sum V_{\mathrm{i}}^{+} G_{\mathrm{ji}}^{+} + V_{\mathrm{i}}^{-} G_{\mathrm{ji}}^{-}}{1 + R_{neu} \sum G_{\mathrm{ji}}^{+} + G_{\mathrm{ji}}^{-}} \tag{4.2}$$

Here, I_{j}, $V_{\mathrm{i}}^{+/-}$, $G_{\mathrm{ji}}^{+/-}$ and R_{neu} carry the same meaning as described in earlier sections. Eqn (1) can be derived by applying Kirchoff's law at the output nodes of the crossbar and considering the voltage drop across the neuron to be $V_{\mathrm{j},neu} = I_{\mathrm{j}} \times R_{neu}$. It is evident that the denominator is close to 1 for smaller arrays as G_{ji}s are much smaller than neuron conductances (resistances of the order of a few hundred ohms [134]). However, larger arrays could lead to $G_{neu} = 1/R_{neu}$ being comparable to sum of the conductances in a particular column. More specifically, a higher number of rows in the crossbar lead to enhanced impact of neuron resistance.

4.2.2 Source Resistance

The source resistance (R_s) in a neuromorphic crossbar could arise due to non-ideal voltage sources and input access selectors lumped together. The input voltages to crossbar gets degraded due to R_s and the degradation can be mathematically modeled as:

$$V_{\mathrm{i},deg}^{+} = V_{\mathrm{i}}^{+} \frac{1/R_s}{1/R_s + \sum \frac{1}{R_{\mathrm{ij}}^{+} + R_{neu}}} \tag{4.3}$$

$$V_{\mathrm{i},deg}^{-} = V_{\mathrm{i}}^{-} \frac{1/R_s}{1/R_s + \sum \frac{1}{R_{\mathrm{ij}}^{-} + R_{neu}}} \tag{4.4}$$

Here $R_{\mathrm{ij}}^{+/-}$ is the resistance of the synaptic element between the i^{th} row and j^{th} column in the positive or negative array. The model ignores the effect of sneak paths. In neuromorphic crossbars, all the inputs are simultaneously active. As the IR drops in the metal lines are negligible, all the nodes in a particular row are supplied by the degraded source voltage of that row. As all the rows are supplied by voltages of same polarity, even the shortest possible current sneak path will experience a low potential difference. Thus, the current through the series connection of the synaptic memristor and neuron would be primarily dependent on

the degraded supply voltage and effective series resistance. We have verified the validity of the model by comparing against SPICE-like simulations, which is described in more detail in Section IV B.

4.2.3 Memristive Conductance Variations

The weights obtained from the training algorithm are usually discretized in order to be represented as memristive synapses. In this chapter, we have used a 4-bit discretization technique where we have used a R_{high}/R_{low} ratio of 15, relevant to the technologies considered. We have mapped the weights such that the maximum weight always maintains the R_{high}/R_{low} ratio to the minimum weight. We have chosen the maximum and minimum weight limits so as to minimize the accuracy degradation due to discretization. To analyze the impact of chip-to-chip variation of weights, we have introduced weight variations in terms of standard deviation (σ) errors, ranging from -2σ to +2σ after discretization. This implies that all the memristive devices on a neuromorphic chip suffer the same variation at a particular process corner. The weight variations are incorporated in the mathematical model as a Δ variation to the conductances.

4.2.4 Proposed Training Algorithm

The mathematical representations of the non-idealities are finally collated and incorporated in the feed-forward path and the backpropagation algorithm for training the ANN. Weights w_{ji} and inputs a_{i} replaces the conductances G_{ji} and voltages V_{i} respectively in Eqn (1) and Eqn (2). The symbol z_{j} is used to represent the current output of the crossbars I_{j} corresponding to jth neuron of the next layer. We assume that the neuronal function receives a current input and provides a voltage output. For the sake of simplicity, we assume ideal mathematical representations of activation functions like 'RELU' [148] and 'sigmoid'[149].

As described in Section. II A, the ideal crossbar output of the jth column in any layer is given by $z_{\mathrm{j}} = \sum_{\mathrm{i}} a_{\mathrm{i}} \times w_{\mathrm{ji}}$. The modified crossbar output can be computed as follows:

$$z_{\mathrm{j}}^{l} = \frac{\sum a_{\mathrm{i},deg}^{+} w_{\mathrm{ji},vary}^{+} + a_{\mathrm{i},deg}^{-} w_{\mathrm{ji},vary}^{-}}{\gamma_{\mathrm{j}}} \tag{4.5}$$

$$\gamma_{\mathrm{j}} = 1 + R_{neu} \sum_{\mathrm{i}} w_{\mathrm{ji},vary}^{+} + w_{\mathrm{ji},vary}^{-}$$

where,

$$a_{\mathrm{i},deg}^{+} = a_{\mathrm{i}} \frac{1/R_s}{\beta_{\mathrm{i}}^{+}}$$

$$a_{\mathrm{i},deg}^{-} = -a_{\mathrm{i}} \frac{1/R_s}{\beta_{\mathrm{i}}^{-}}$$

$$w_{\mathrm{ij},vary}^{+/-} = w_{\mathrm{ij}}^{+/-} + \Delta$$

$$\beta_{\mathrm{i}}^{+/-} = 1/R_s + \sum \frac{1}{R_{\mathrm{ij}}^{+/-} + R_{neu}}$$

$$R_{\mathrm{ij}}^{+/-} = 1/w_{\mathrm{ij},vary}^{+/-}$$

As described earlier, two weight matrices are deployed to account for bipolar weights in the original weight matrix $W = [w_{\mathrm{ji}}]$. Positive (Negative) inputs are fed to the positive (negative) weight array. The weight matrices are created such that $w_{\mathrm{ji}}^{+}(w_{\mathrm{ji}}^{-}) = 0$ for all i,j for which $W_{\mathrm{ji}} < 0 (> 0)$ and $w_{\mathrm{ji}}^{+}(w_{\mathrm{ji}}^{-}) = W_{\mathrm{ji}}$ for all i,j for which $W_{\mathrm{ji}} > 0 (< 0)$. Note that mapping the weights to a particular conductance range is equivalent to multiplication by a scaling factor as we have already discretized the weights based on a maximum to minimum weight ratio equal to $G_{high}/G_{low} = 15$. Thus an equivalent representation in terms of conductance would be $G_{\mathrm{ji}}^{+/-} = W_{\mathrm{ji}} G_{high}$.

The output of each crossbar is passed as inputs to the next crossbar through a sigmoid function such that $a_{\mathrm{i}}^{L+1} = \sigma(z_{\mathrm{i}}^{L})$ (where L is the layer index). The backpropagation algorithm is modified to account for the modified crossbar functionality. As described earlier, learning in neural networks relies on computation of gradients of a cost function. Here, it is calculated from the error between the expected and the actual output of the output layer neurons in the form of $C = \frac{1}{2} \sum (y_{\mathrm{j}} - a_{\mathrm{j}}^{L})^2$. The delta-rule in the backpropagation algorithm [162] involves

calculation of δ for each layer accounting for the change in the cost function for unit change in inputs to that particular layer. Thus, δ for layer l can be written as:

For output layer,

$$\delta_{\mathrm{j}}^{L} = \frac{\partial C}{\partial z_{\mathrm{j}}^{L}} = \sum \frac{\partial C}{\partial a_{\mathrm{j}}^{L}} \frac{\partial a_{\mathrm{j}}^{L}}{\partial z_{\mathrm{j}}^{L}} = (a_{\mathrm{j}}^{L} - y_{\mathrm{j}})\sigma(a_{\mathrm{j}}^{L}) \tag{4.6}$$

For other layers,

$$\delta_{\mathrm{j}}^{l} = \frac{\partial C}{\partial z_{\mathrm{j}}^{l}} = \sum_{k} \frac{\partial C}{\partial z_{k}^{l+1}} \frac{\partial z_{k}^{l+1}}{\partial z_{\mathrm{j}}^{l}} = \sum_{k} \delta_{k}^{l+1} \frac{\partial z_{k}^{l+1}}{\partial z_{\mathrm{j}}^{l}} \tag{4.7}$$

$$\frac{\partial z_{k}^{l+1}}{\partial z_{\mathrm{j}}^{l}} = \frac{\partial z_{k}^{l+1}}{\partial a_{\mathrm{j}}^{l}} \frac{\partial a_{\mathrm{j}}^{l}}{\partial z_{\mathrm{j}}^{l}} = \frac{\partial z_{k}^{l+1}}{\partial a_{\mathrm{j}}^{l}} \sigma(a_{\mathrm{j}}^{l})$$

$$\frac{\partial z_{k}^{l+1}}{\partial a_{\mathrm{j}}^{l}} = \frac{\frac{a_{\mathrm{j},deg}^{+,l}}{a_{\mathrm{j}}^{l}} w_{\mathrm{j}k,vary}^{+} - \frac{a_{\mathrm{j},deg}^{-,l}}{a_{\mathrm{j}}^{l}} w_{\mathrm{j}k,vary}^{-}}{\gamma_{\mathrm{j}}} \tag{4.8}$$

Finally, the δs of each layer are used to compute the weight updates as:

$$dw_{\mathrm{j}k}^{l} = \frac{\partial C}{\partial w_{\mathrm{j}k}^{l}} = \frac{\partial C}{\partial z_{\mathrm{j}}^{l}} \frac{\partial z_{\mathrm{j}}^{l}}{\partial w_{\mathrm{j}k}^{l}} = \delta_{\mathrm{j}}^{l} \frac{\partial z_{\mathrm{j}}^{l}}{\partial w_{\mathrm{j}k}^{l}} \tag{4.9}$$

$$\frac{\partial z_{\mathrm{j}}^{l}}{\partial w_{\mathrm{j}k}^{l}} = \frac{\gamma_{\mathrm{j}}(a_{k,deg}^{+}(1 - \frac{w_{k\mathrm{j}}^{+}}{\beta_{k}^{+}}) + a_{k,deg}^{-}(1 - \frac{w_{k\mathrm{j}}^{-}}{\beta_{k}^{-}})) - R_{neu} z_{\mathrm{j}}^{l} \gamma_{\mathrm{j}}}{\gamma_{\mathrm{j}}^{2}} \tag{4.10}$$

To simulate the impact of non-idealities on varying crossbar size, we divide the large crossbars of size $M \times N$ into several smaller crossbars of size $m \times n$. Fig. 6.5(c) shows the network architecture of combining smaller crossbars to realize the neuromorphic functionality of larger crossbars. The source degradation factor β_{i} is more prominent for larger number of columns as it depends on the term $\sum_{\mathrm{j}} 1/(R_{\mathrm{ij}} + R_{neu})$ summed over the columns. The neuron resistance degradation factor γ_{j}, on the other hand, increases with the number of rows due to its dependence on the term $\sum_{\mathrm{i}} w_{\mathrm{ji}}$, summed over the rows. Thus, the combined effect of these two non-idealities is expected to have a higher impact on the network for larger crossbars.

Table 4.2. CNN Architecture

Input 32 × 32 RGB image
5 × 5 conv. 64 RELU 2 × 2 max-pooling stride 2
5 × 5 conv. 128 RELU 2 × 2 max-pooling stride2
3 × 3 conv. 256 RELU 2 × 2 avg-pooling stride 2
4 × 4 conv. 512 Sigmoid (fully connected) 0.5 Dropout
1 × 1 conv. 10 (fully connected)
10-way softmax

4.3 Simulation Framework

4.3.1 Model simulations

The model described in the previous section was implemented on FCNs using the MATLAB® Deep Learning Toolbox[163] and CNNs using MatConvNet [164].

FCN

A 3-layered neural network was employed to recognize digits from the MNIST Dataset. The training set consists of 60000 images, while the testing set consists of 10000 images. The input layer consists of 784 neurons designated to carry the information of each pixel of each 28×28 image. The hidden layer consists of 500 neurons and the output layer has 10 neurons to recognize 10 digits. The neuron transfer function was chosen to be the sigmoid function which can be written as $\sigma(x) = \frac{1}{1+e^{-}x}$.

CNN

For the classification of more complex dataset CIFAR-10, we have used a network with RELU-activated convolutional layers and a sigmoid-activated fully connected layer. The architecture is represented as 32×32×3-64c5-2s-128c5-2s-256c3-2s-512o-10o.The details of the layers are provided in Table. II. Essentially, the different layers represent subsequent

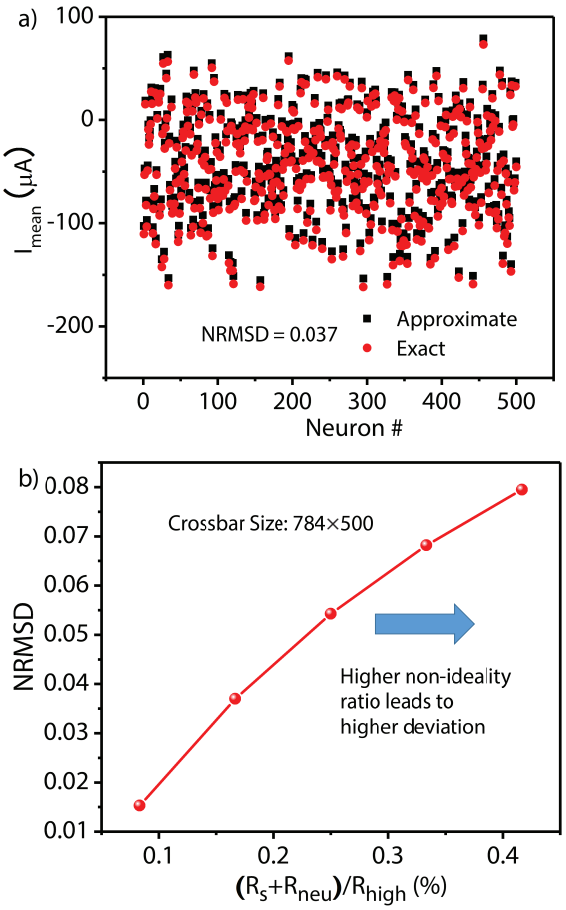


Figure 4.4. (a) Distribution of output currents (I_{mean}), averaged over 100 images, across 500 neurons in the hidden layer comparing the approximate model to SPICE-like simulation framework. (b) Variation of Normalized Root Mean Square Deviation (NRMSD) with non-ideality ratio. NRMSD is close to zero for the relevant range of non-idealities.

operations such as convolution, max-pooling or average-pooling and activation. The operations are described in detail in Secton II.A. Each convolutional layer is followed by a batch-normalization layer for better performance. We concentrate our analysis on the fully connected layers of the network as the initial convolutional layers possess local connections implemented on small crossbars equal to the kernel sizes.

4.3.2 SPICE-like Simulations for validation

Each fully connected layer for both FCNs and CNNs can be implemented in a crossbar architecture comprising of all possible connections. A SPICE-like framework was implemented in MATLAB® by creating a netlist of all connections, voltage source, source and neuron resistances in such resistive crossbars and evaluating the voltages at each node by solving the conductance matrix: $[V] = [G]^{-1}[I]$. The framework was benchmarked with HSPICE®. This framework was used to calculate the output of non-ideal crossbars on application of the inputs from the MNIST dataset as voltages. The resistances of the crossbar elements R_{ji} were determined such that $R_{\mathrm{ji}} = 1/w_{\mathrm{ji}}$, where w_{ji} are the weights determined by the ideal training scheme described in the previous subsection. The output obtained by showing 100 images of the testing set was averaged and the distribution was compared with the mathematical model simulations. Fig. 6.8(a) shows the comparison in the distribution of output currents of a crossbar where the approximate model shows good agreement with the exact SPICE-like simulations. Fig. 6.8(b) shows that the normalized root mean square deviation (NRMSD) between the two techniques for various $(R_s + R_{neu})/R_{high}$ combinations remains very close to zero for relevant values. As SPICE simulations automatically takes account of possible sneak paths, the agreement of our model to SPICE simulations means that the effect of sneak paths, even if not absolutely zero, is insignificant. Thus, the dominant issues in the crossbar to be considered are source and neuron resistances. It is to be noted that the validation of our approximate model was important in the context of reducing the time required for simulating the training and inferencing of each network for the entire dataset as the matrix operations could be more efficiently performed using the mathematical model. This eliminated simulating the network for each input image in HSPICE®and the subsequent iterative steps involving MATLAB-SPICE interfacing.

4.4 Results and Discussion

We analyzed the impact of technological constraints in crossbar implementations on both FCNs and CNNs. As fully connected layers form the crux of classification in both network topologies, it is expected that such non-ideal conditions will have similar detrimental effects

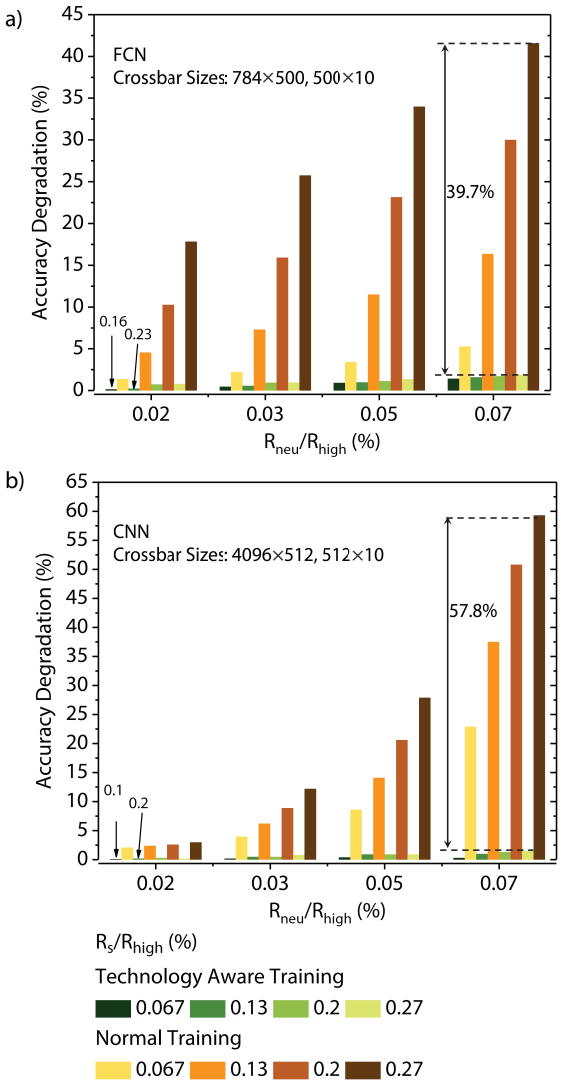


Figure 4.5. Accuracy degradation v/s varying R_{neu}/R_{high} ratio for different R_s/R_{high} combinations comparing technology aware training scheme with normal training for (a) FCN and (b) CNN.

on both. We present the detailed impact of each non-ideality on FCNs and CNNs for better understanding.

We consider a 3-layered FCN and a CNN architecture described in Table. 4.2 to analyze the impact of the non-idealities on the accuracy of recognition task on MNIST and CIFAR-10 datasets respectively. The other convolutional layers in the CNN are usually implemented using small crossbars and hence do not suffer significant effects of non-ideal resistances.

First, the neural networks were trained under ideal conditions using the training set. Then, the non-ideal model was included in the feed-forward path and the ideally trained network was tested using the testing set to determine the performance degradation due to the non-idealities. Next, the technology aware training algorithm was implemented by incorporating the mathematical formulation of the non-idealities in the standard training iterations of feed-forward and backpropagation as described in the Section III D. For each iteration, the weights were discretized as described in Section III C. The testing accuracy of an ideally trained FCN with a sigmoid neuronal function was 98.12% on MNIST and that of an ideally trained CNN was 85.6% on CIFAR-10 datasets. The accuracy degradations discussed in this section has been calculated with respect to these ideal testing accuracies such that Accuracy Degradation (%) = Ideal Accuracy (%) - Accuracy Obtained (%). We use the parameters R_s/R_{high} and R_{neu}/R_{high} to denote the ratios of the non-ideal resistances and the maximum synaptic resistance.

Source and Neuron Resistance

Fig. 4.5(a) and 4.5(b) shows the accuracy degradation for different R_{neu}/R_{high} and R_s/R_{high} combinations in FCN and CNN, respectively. The effect of the non-ideal resistances on the performance of the network predictably worsens monotonically with higher R_s/R_{high} and R_{neu}/R_{high} ratios. It can be observed that with normal training methods, the non-ideal resistances result in accuracy degradation for FCN: up to 41.58% for $R_{neu}/R_{high} = 0.07\%$ and $R_s/R_{high} = 0.27\%$. Our proposed training scheme incorporates the impact of non-idealities and achieves significant restoration of accuracy, within 1.9% of the ideal accuracy, for the worst case combination of resistances considered, shown in Fig. 4.5(a).

In case of CNNs, we show that due to the large crossbar sizes of the fully connected layers in the CNN, it can suffer up to 59.3% degradation in accuracy for the worst case non-ideal resistances considered. Our proposed algorithm, on the other hand, achieves an accuracy within 1.5% of the ideal accuracy (Fig. 4.5(b)), considering the largest crossbar sizes for the architecture.

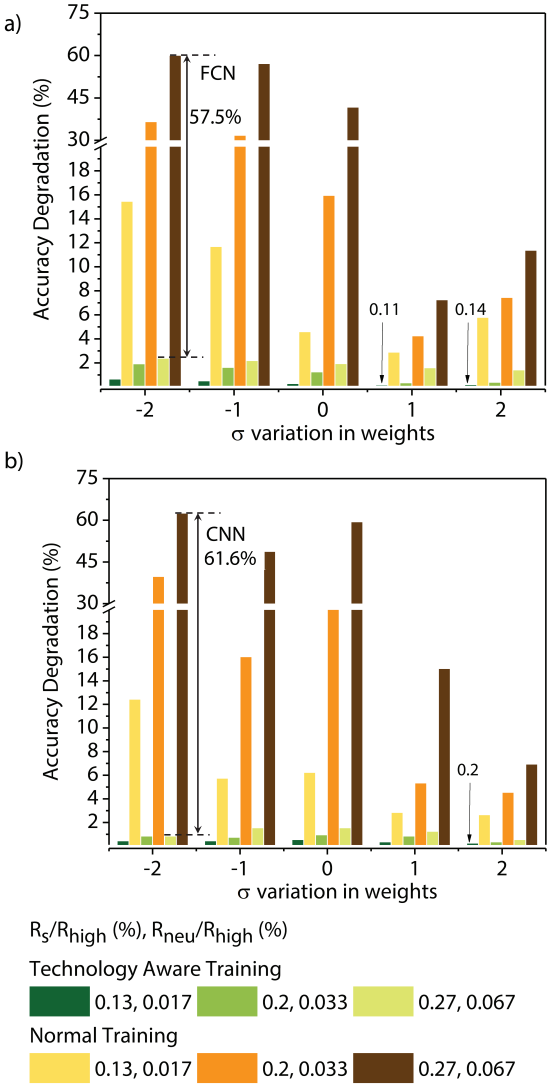


Figure 4.6. Accuracy degradation v/s σ variations in weights for various R_s/R_{high} and R_{neu}/R_{high} combinations comparing the technology aware training scheme with normal training for (a) FCN and (b) CNN.

Weight variations

On-chip crossbar implementations suffer from chip-to-chip device variations. To account for such variations, we form a defect weight matrix, and include it in the feed-forward network, as described in detail in Section III C. We have considered up to $\pm 2\sigma$ variation in the synaptic weights. Fig. 4.6 shows the impact of such device variations on the accuracy of FCN and CNN for different combinations of R_s/R_{high} and R_{neu}/R_{high}. Predictably, changes in the

positive direction reduces the accuracy degradation from the nominal (no variation) case as it enhances the significance of the neurons. However, changes in the negative direction slightly degrades the accuracy from the nominal case. It is observed that a -2σ variation can result in an accuracy degradation of up to 59.9% for $R_{neu}/R_{high} = 0.067\%$ and $R_s/R_{high} = 0.27\%$ in FCN. By accounting for these variations in the backpropagation algorithm, our proposed training methodology successfully restores the accuracy within 2.34% of the ideal accuracy for worst case of non-idealities considered, as shown in Fig. 4.6(a).

Weight variations in the negative direction also adversely affect CNNs where -2σ variation can result in an accuracy degradation of 62.4% considering the non-ideal resistances mentioned above. Our proposed algorithm achieves an accuracy within 0.8% of the ideal testing accuracy as shown in Fig. 4.6(b).

Crossbar Size

Non-idealities in crossbars usually establish restrictions on the allowable crossbar sizes due to the dependence of their performance on fan-in and fan-out. For example, the impact of R_s on the crossbar depends on the parallel combination of column resistances and a higher number of columns (and hence, higher fan-out) result in severe performance degradation. Also, the impact of R_{neu} intensifies with increasing number of rows in the crossbar as it leads to more fan-in. As observed in Fig. 4.7(a), the combined effect of these resistances and variations can result in significant accuracy degradation (41.58%) when the network is implemented on crossbars of sizes 784×500 and 500×10 for the respective layers in the FCN. Under the same non-ideal conditions, accuracy degradation drops to 1.2% when smaller crossbars of sizes 112×100 and 100×10 are used to represent the functionality of the network. In contrast, considering the same R_s and R_{neu}, our proposed training algorithm achieves an accuracy degradation within $\sim 1.89\%$ for sizes $784 \times 500, 500 \times 10$ and $\sim 0.3\%$ for sizes $112 \times 100, 100 \times 10$. Thus, the proposed algorithm ensures that a network implemented on larger crossbars can parallel the performance of ideally trained networks implemented on smaller crossbars with minimal degradation.

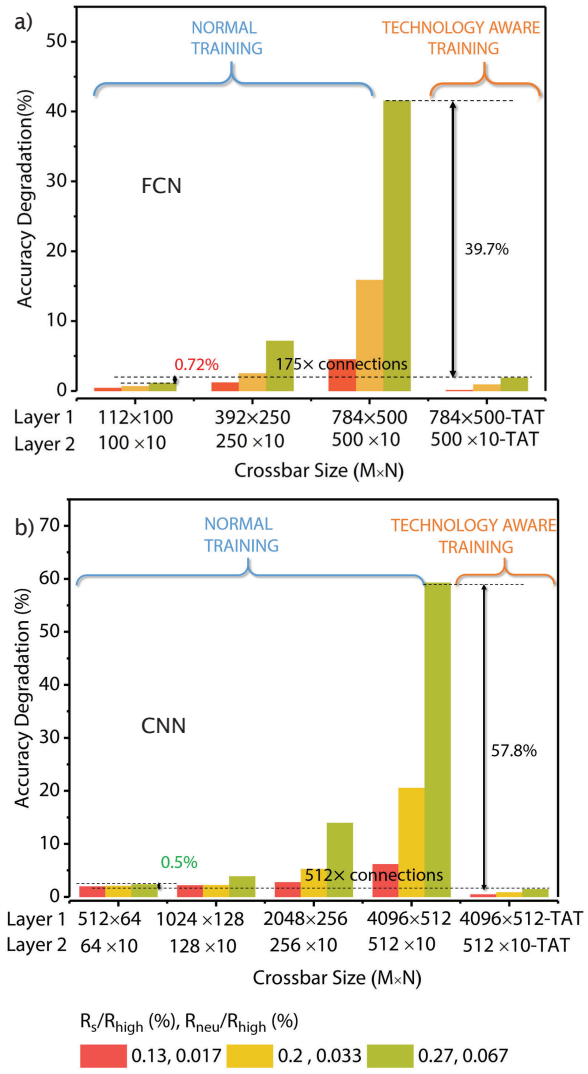


Figure 4.7. Accuracy degradation v/s crossbar size for various R_s/R_{high} and R_{neu}/R_{high} combinations comparing the technology aware training scheme with normal training for (a) FCN and (b) CNN. Larger crossbars show higher accuracy degradation.

The convolutional layers in CNNs are implemented on smaller crossbars. For the fully connected layers in the CNN architecture, we have considered significantly larger crossbars of sizes 4096×512 and 512×10. Due to large sizes of the last 2 layers of the considered architecture, we show in Fig. 4.7(b), that the network, when trained under ideal conditions, can suffer as large as 59.3% degradation in accuracy for the worst case resistance constraints considered. On the other hand, using smaller crossbars of sizes $512 \times 64, 64 \times 10$ reduces the accuracy degradation to 2.4% for the same conditions. In comparison, a network trained with

the proposed technology aware training algorithm restores the accuracy to within $\sim 1.5\%$ of the ideal accuracy even for the highest crossbar sizes ($4096 \times 512, 512 \times 10$). Thus, the proposed algorithm ensures that a CNN with fully connected layers implemented on crossbars of size in the order of 4096×512 can achieve better performance than for crossbars of size 512×64 with standard training algorithms. Such a provision of using large crossbars for implementing neuromorphic systems could potentially reduce overheads of repeating inputs, time multiplexing outputs, thus ensuring faster operations.

4.5 Conclusion

Hardware implementations of neuromorphic systems in crossbar architecture could suffer from various non-idealities resulting in severe performance degradation when employed in machine learning applications such as recognition tasks, natural language processing, etc. In this chapter, we analyzed, by means of mathematical modeling, the impact of non-idealities such as source resistance, neuron resistance and chip-to-chip device variations on performance of a 3-layered FCN on MNIST and a state-of-the-art CNN architecture on CIFAR-10. Severe degradation in recognition accuracy, up to 59.84%, was observed in FCNs. Although convolution layers in CNN can be implemented on smaller crossbars, the large fully connected layers at the end made them prone to performance degradation (up to 62.4% for our example). As a solution, we proposed a technology aware training algorithm which incorporates the mathematical models of the non-idealities in the training algorithm. Considering relevant ranges of non-idealities, our proposed methodology recovered the performance of the network implemented on non-ideal crossbars to within $\sim 2.34\%$ of the ideal accuracy for FCNs and $\sim 1.5\%$ for CNNs. We further show that the proposed technology aware training algorithm enables the use of larger crossbars of sizes in the order of 4096×512 for CNNs and 784×500 for FCNs without significant performance degradation. Thus, we believe that the proposed work potentially paves the way for implementation of neuromorphic systems on large crossbars which otherwise is rendered unfeasible using standard training algorithms.

5. GENIEX: A GENERALIZED APPROACH TO EMULATING NON-IDEALITY IN MEMRISTIVE XBARS USING NEURAL NETWORKS

5.1 Introduction

The pervasiveness of deep learning in a wide-variety of applications such as object detection, language processing etc. has been a major force behind the recent success of Artificial Intelligence (AI). Consequently, there has been a growing interest in developing specialized accelerators to improve the efficiency of deep learning. Such accelerators include Google TPU [3], Microsoft BrainWave [10], and Nvidia V100. One key aspect driving these accelerators is moving computations closer to the memory, which has brought forth the paradigm of in-memory computing. Despite the breakthroughs in custom hardware, the storage and computation requirements of Deep Neural Networks (DNNs) have been increasing at a much faster rate than the efficiency improvements in digital CMOS hardware [57]. To this effect, researchers have explored Non Volatile Memory (NVM) [59], [165] based crossbar architectures to achieve higher on-chip storage density and efficient MVMs in the analog domain [15], [16].

NVM devices can store multiple states per device, and crossbars built with these devices can be integrated on chip leading to high storage density [165]. Second, the voltage-driven nature of these two-terminal devices enables crossbar-like arrangement to perform MVMs, at significantly higher efficiency compared to digital CMOS [104]. Despite the multifold promises of NVM technologies, the analog nature of computing in crossbars poses several challenges due to the device and circuit non-idealities such as: parasitic resistance, non-linearity from access transistors, and I-V characteristics of the NVM device. Parasitic resistances lead to undesirable IR-drops in the metal lines of the crossbar. On the other hand, the non-linearity leads to inaccurate multiplications at the cross-points. As a result, non-idealities can have an adverse effect on the MVM arithmetic. This gets exacerbated further due to the device variations. Eventually, the inaccuracies in the MVM arithmetic

can accumulate over the multiple layers of a neural network, causing significant accuracy degradation [69].

To address this accuracy degradation, there have been efforts towards exploring techniques to model non-idealities and subsequently mitigating them [22], [69], [166]. The efficacy of these mitigation techniques strongly depend upon the modelling [22], [69], [167] approach to exhaustively capture the sources of the non-idealities and retraining of the neural network weights. The non-idealities in crossbars can be broadly categorized into non-data dependent or linear (for eg. parasitic resistances), and data-dependent or non-linear types (for eg. access transistors and device I-V characteristics). While the current analytical techniques can model the non-data dependent aspects [22], [69], [167], they fail to capture the data dependent non-idealities. Data-dependent non-idealities can have a pronounced effect on the crossbar outputs, particularly at higher operating voltages (discussed in Section 5.3). Thus, it is important to move away from approximate analytical models to data-based models in order to truly capture all the non-idealities. In this chapter, we present GENIEx, a neural network based modelling approach that provides an accurate as well as generalized representation of the non-ideal behavior of crossbars. The key contributions of this work are:

- Analyze the sources of non-ideality in crossbars through extensive SPICE simulations (Section 5.3).
- Propose GENIEx, a generalized approach for modelling non-idealities in crossbars using neural networks (Section 5.4).
- Develop a PyTorch-based functional simulator which models the key architectural aspects namely tiling, and bit-slicing to evaluate large-scale DNNs using GENIEx (Section 5.5).
- Perform detailed analysis of different non-idealities on the classification accuracy of DNNs (Section 5.7).

To the best of our knowledge, this is the first work proposing an end-to-end framework for data-dependent crossbar modeling along with a functional simulator considering tiling, and

bit-slicing. This enables studying the accuracy impacts of device and circuit properties at the application level. It is worth noting that due to the ability to capture data dependency of crossbar behavior (transfer characteristics), GENIEx can be used to model crossbars from both simulations as well as experimental measurements. *We believe that our proposed approach paves the way for universal modeling of practical crossbars with the scope of seamless functional evaluation and mitigation.* We plan to open-source the framework for further research on crossbar hardware.

5.2 Related Work

Past research have explored modeling crossbar non-idealities and subsequently mitigating them [22], [69], [166], [168]. Jain et al [69] used matrix inversion techniques to model the effects of parasitic resistances due to input driver, metal lines *etc.* Liu et al [168] proposed an approximation technique based on sample input/output behavior. An alternative way of capturing effects such as stuck-at-faults [169] or device variations [170] is to map the distribution of the variations or defects. While the above modelling approaches [69], [168]–[170] consider linear (non-data dependent) non-idealities, GENIEx also captures the non-linear (data dependent) non-idealities. Note, there could be non-linearity during programming of NVM devices. Sun et al [171] propose analytical models to study such non-linearity during programming. However, analyzing the impact of non-linearity on the subsequent MVM computations (after programming) requires a data-dependent model like GENIEx.

Researchers have also proposed evaluation frameworks such as [69] and NeuroSim [172] to study the impact of these non-idealities using analytical models. Other works have explored the impact of quantization noise of ADCs for analog computing [173]. However, these frameworks do not consider the architectural aspects of MVM computations such as tiling and bit-slicing, which have a significant implication on classification accuracy (shown in Section 5.7.2). Our work explores a neural network based technique to model the crossbar non-idealities using a functional simulator with detailed MVM architecture. Table 5.1 summarizes our contribution with respect to the related work.

Table 5.1. Related work comparison

Related Work	Linear + Non-linear non-idealities	Large scale DNNs	Architecture model of MVM
GENIEx	✔	✔	✔
CxDNN [69]	✗	✔	✗
CrossSim [174]	✔	✗	✗
NeuroSim [172]	✔	✗	✗
AMS [173]	✗	✔	✗

5.3 Analysis of NVM Non-Idealities

Background: A typical memristive crossbar consists of NVM devices arranged in a crossbar fashion as shown in Figure 5.1. The two terminals of each NVM device connect to a horizontal word-line (WL) and a vertical bit-line (BL). These devices are accompanied by access transistors or selectors to avoid the sneak path issues during writing [175]. This primitive can be used to compute Matrix Vector Multiplications (MVMs) in the *analog domain* by activating all the WLs and sensing all the BLs simultaneously. For example, to perform a multiplication between a $1 \times N$ vector and a $N \times M$ matrix, the vector is encoded as input voltages (V_i) while the matrix is encoded as conductances (G_{ij}). Consequently, the output current in the j^{th} BL (*for ideal crossbar*) is the sum of currents through each NVM device in the corresponding column: $I_j = \sum_i V_i G_{ij}$. Thus, the currents from the M columns constitute the output vector of the MVM operation. Typically, a crossbar requires peripheral circuits such as Digital-to-Analog Converters (DACs) and Analog-to-Digital Converters (ADCs) for system-level integration. The DACs convert the digital inputs into analog voltages while the ADCs convert the analog currents in the BLs to digital outputs. Due to the analog nature of computing, several non-idealities can lead to errors in the MVM computations. These non-idealities can be classified into two kinds - linear and non-linear, as shown in Table 5.2.

Analysis: Under the influence of non-idealities, the crossbar design parameters such as size, ON resistance, conductance ON/OFF ratio etc. can have a considerable effect on the magnitude of errors in computations. To analyze this effect, we perform SPICE analysis of a 64×64 crossbar. Herein, the linear non-idealities are modeled using parasitic resistances

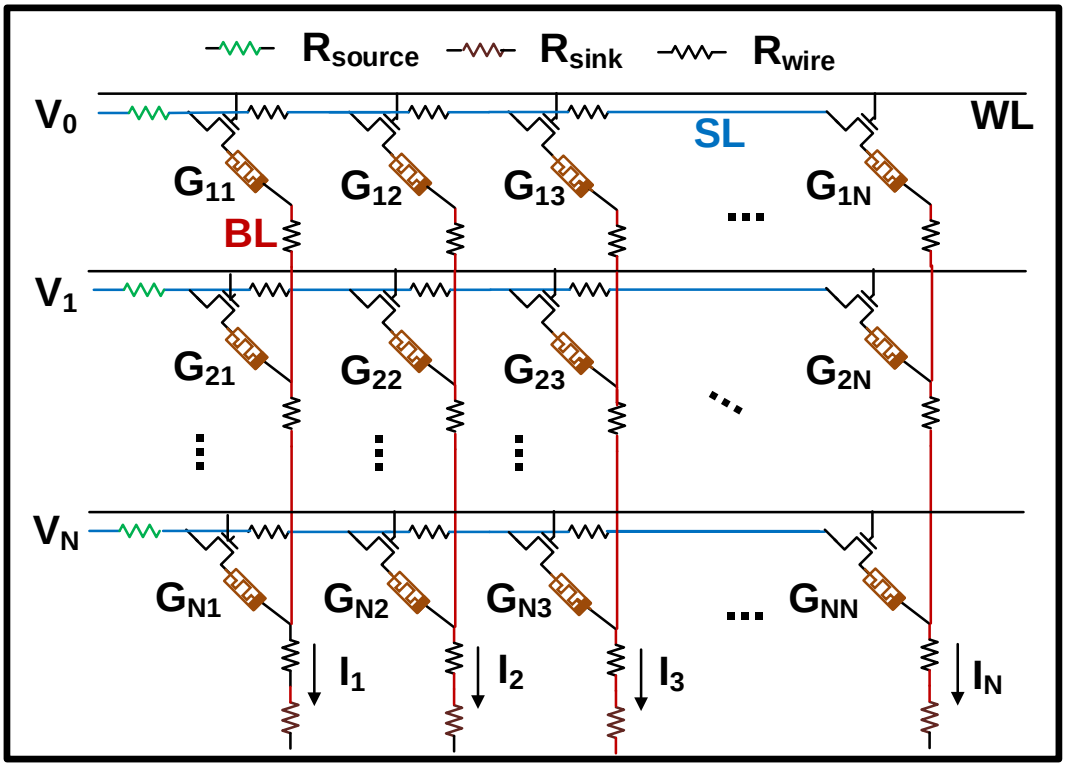


Figure 5.1. A typical non-ideal crosspoint structure with NVM devices accompanied by a transistor at every junction of the word-lines (WL) and bit-lines (BL).

Table 5.2. Non-idealities in crossbar

Linear Non-idealities	**Non-linear Non-idealities**
Source Resistance (R_{source})	Access devices or selectors
Sink Resistance (R_{sink})	Device non-linearity
Wire Resistance (R_{wire})	

as shown in Figure 5.1. The access devices are based on transistor models from TSMC 65nm technology. The device models are adopted from a compact model of a filamentary RRAM [176], where the current flowing through the device can be expressed as: $I(d, V) = I_0 exp(\frac{d}{d_0}) sinh(\frac{V}{V_0})$. Here, d is the gap-size between the tip of the filament and electrode, I_0, d_0 and V_0 are fitting parameters.

Figure 5.2 (a) shows a typical plot of ideal current (I_{ideal}) v/s non-ideal current ($I_{non-ideal}$) of a crossbar. Here, we observe that different voltage (V) and conductance (G) conditions which lead to similar I_{ideal} can result in a varying range of $I_{non-ideal}$ outputs, causing errors

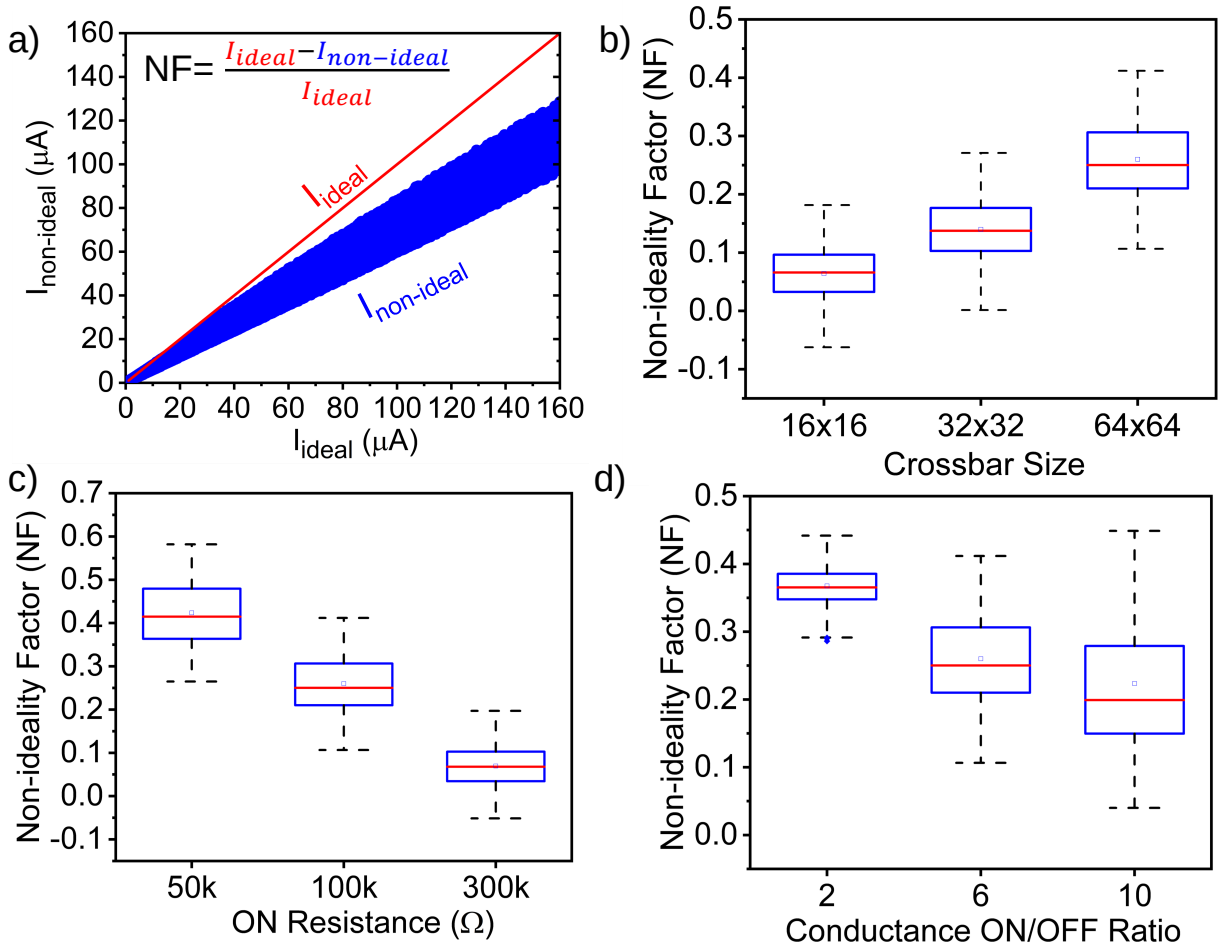


Figure 5.2. (a) Output currents from a 64x64 crossbar showing the deviation of ($I_{non-ideal}$) from (I_{ideal}). (b), (c) and (d) shows the box-plot variation of the NF with varying crossbar design parameters.

in computations. To quantify the error, we define a non-ideality factor (NF) as the relative error between the I_{ideal} and $I_{non-ideal}$. NF is calculated as: $\frac{I_{ideal}-I_{non-ideal}}{I_{ideal}}$. We observe in Figures 5.2 (b) and (c) that lower ON resistances and higher crossbar sizes lead to higher NF. This is due to the fact that bigger crossbars have longer metal lines leading to higher R_{wire}. Moreover, the parallel combination of resistances along the columns and rows results in a reduced effective resistance of the crossbar in case of bigger crossbars as well as low ON resistances. In addition, Figure 5.2 (d) shows that lower ON/OFF conductance ratio leads to high NFs. This is due to the fact that for a given ON resistance, the average resistance in the crossbar is low for lower ON/OFF ratio.

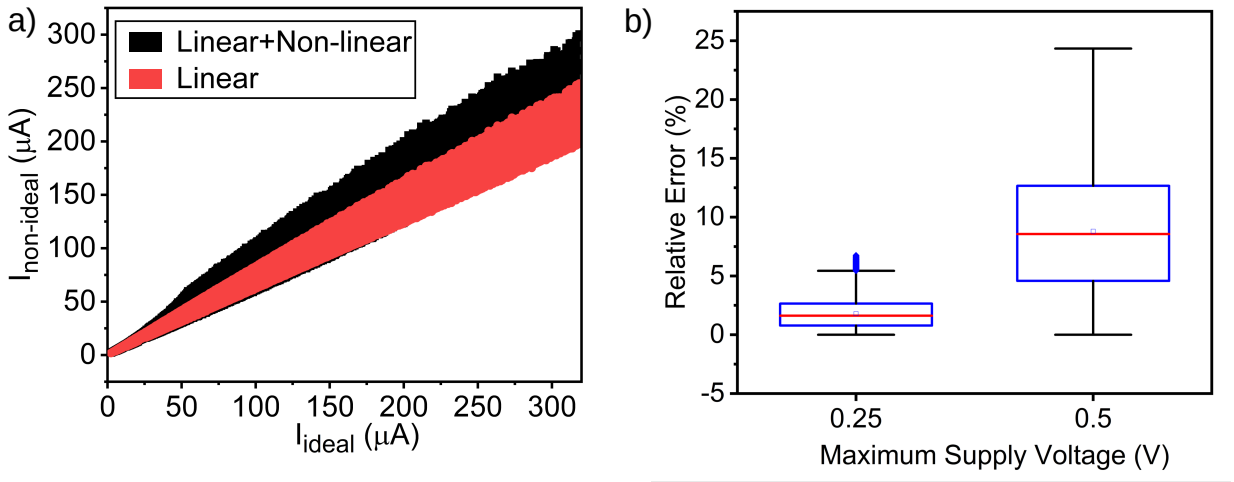


Figure 5.3. (a) Output current distribution showing impact of non-linearity. (b) Relative error between the cases with and without nonlinearity increases with increase in maximum supply voltage.

Next, we analyze the impact of non-linear non-idealities. We consider two cases i) only linear non-idealities, ii) both linear and non-linear non-idealities. Figures 5.3 (a) and (b) show the relative difference in output currents between the two cases. We observe that the output currents in case (i) vary noticeably from case (ii). This effect becomes even more prominent for higher supply voltage of $V_{supply} = 0.5V$, thereby implying an inherent data dependence of $I_{non-ideal}$ on the V and G. *This result underlines the drawbacks of analytical models which fail to capture the data-dependent non-idealities.* We propose a neural network based modeling technique that captures the data-dependent errors in crossbar computations.

5.4 GENIEx - A Neural Network Based Crossbar Model

Neural networks project data to a high dimensional space which enables them to distinguish between different input patterns. We leverage this property of neural networks to propose GENIEx, which models the non-ideal behavior of memristive crossbars for different input voltage and conductance combinations. As discussed in Section 5.3, non-idealities in crossbars can lead to a varying range of NF for similar I_{ideal}. Using a neural network can help us capture the data-dependent nature of such non-ideal behavior.

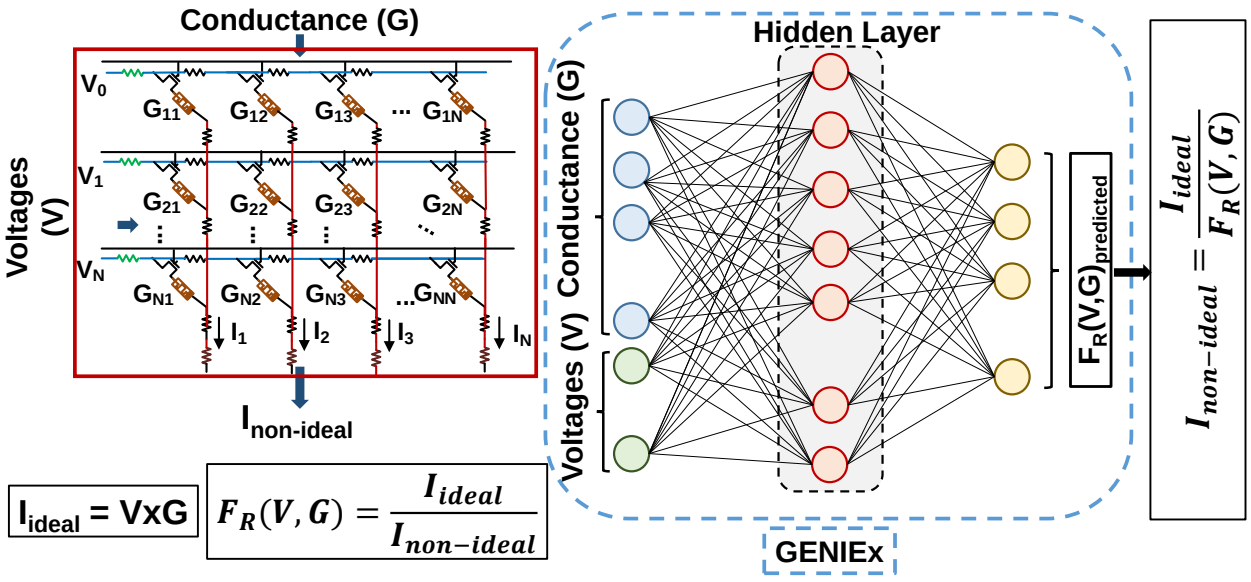


Figure 5.4. Crossbar computation mapped to GENIEx. V and G are concatenated to form the input vector for neural network, with output being the ratio $f_R = I_{ideal}/I_{non-ideal}$.

NN Formulation: The output current vector of an ideal crossbar (I_{ideal}) represents an MVM operation between V and G. Meanwhile, the output current vector from a real crossbar is non-ideal and can be expressed as a distorted MVM function: $I_{non-ideal} = f_D(V, G)$. Therefore, it represents multiplicative behavior between the input variables, V and G. The objective here is to model such non-ideality function $f_D(V, G)$ being input-dependent and having multiplicative behavior. The intuitive way of modeling $f_D(V, G)$ using neural networks is to provide V and G as inputs and obtain $I_{non-ideal}$ as output. However, as neural networks perform linear transformations, it is difficult for them to model multiplicative interactions between its inputs. To avoid such input multiplications, we propose extracting only the distortion information of the real output current from $f_D(V, G)$. We define a function which represents the ratio of I_{ideal} to $I_{non-ideal}$: $f_R(V, G) = \frac{I_{ideal}}{I_{non-ideal}}$. $f_R(V, G)$ represents the deviation of the $I_{non-ideal}$ from I_{ideal}, thus eliminating the need to capture multiplicative relationships. For an $N \times N$ crossbar, the input vector to the neural network is a concatenation of $(N \times 1)$ voltage vector and $(N^2 \times 1)$ flattened conductance vector. The output vector obtained is $f_R(V, G)$ which is of size $N \times 1$. Subsequently, the $I_{non-ideal}$ is obtained using $I_{ideal}/f_R(V, G)$. **Dataset:** To train GENIEx for predicting the ratio $f_R(V, G)$ for a set of V and G vectors, we create a dataset covering the exhaustive space of V and G combinations.

Crossbar-based accelerators commonly use bit-slicing to perform high precision MVM operations [15], [16]. We observed that this leads to high sparsity in V and G vectors across the popular deep learning tasks. To exhaustively capture the resulting sparse data distributions, we consider various degrees of sparsity while generating the training set of V and G. We apply the V and G vectors to various crossbars and perform SPICE simulations to obtain the corresponding $I_{non-ideal}$. The obtained $I_{non-ideal}$ is used to calculate $f_R(V, G)$, the prediction labels for the dataset. To evaluate the accuracy of GENIEx, we create a separate validation set of V, G and expected $f_R(V, G)$.

NN Topology: GENIEx considers a two layer fully-connected neural network consisting of an input layer, a hidden layer and an output layer. For a $N \times N$ crossbar, the size of the neural network is given as: $(N^2 + N) \times P \times N$, where P is the number of neurons in the hidden layer. The training set mentioned above is used to train the neural network by feeding V, G combinations as inputs and $f_R(V, G)$ as the output.

Benchmarking: We compare the accuracy of GENIEx against HSPICE results and a baseline linear analytical model for the same test voltage and conductance combinations. We use the metric NF, defined in Section 5.3, to compare the models with HSPICE results. The baseline analytical model considers only linear non-idealities. We observe, in Figure 5.5 that even at a low supply voltage of $V_{supply} = 0.25V$, GENIEx achieves a Root Mean Square Error (RMSE) of 0.25 while estimating the NF with respect to HSPICE. This is 7× lower than the baseline analytical model. For higher supply voltage of $V_{supply} = 0.5V$, GENIEx achieves a RMSE of 0.7, which is 12.7× lower than the analytical model. To evaluate the impact of non-idealities on large scale DNNs, we develop a functional simulator that incorporates GENIEx with detailed MVM architecture model.

5.5 Functional Simulator

Several frameworks such as Ares [177], Distiller [178] etc. have been developed using TensorFlow and PyTorch to enable hardware-software codesign studies. However, such frameworks cannot emulate the implications of crossbar-based hardware, because of the intrinsic differences in the CMOS-based and Crossbar-based computation models. For CMOS, ma-

RMSE (wrt SPICE)

	V=0.25V	V=0.5		V=0.25V	V=0.5
Analytical	**1.73**	**8.99**	**GENIEx**	**0.25**	**0.7**

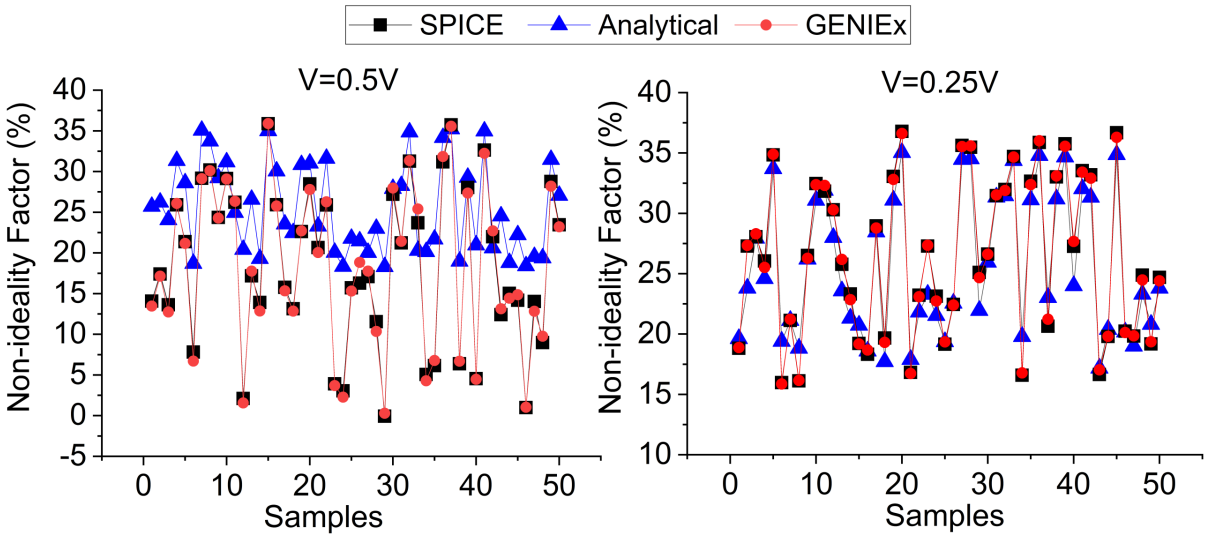


Figure 5.5. Comparison of NF for a typical 64x64 crossbar between HSPICE outputs, analytical model and GENIEx.

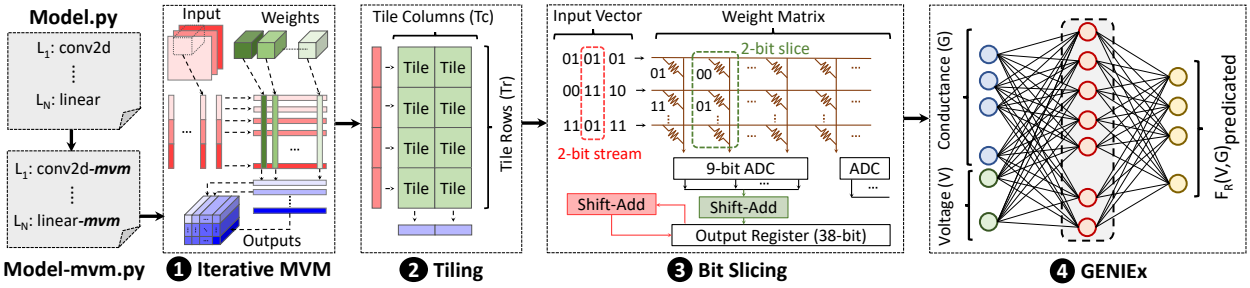


Figure 5.6. Logical organization of functional simulator

trix operations (ops) in a ML model are expressed as General Matrix-Matrix Multiplications (GEMMs) that use floating/fixed point compute units, whereas Crossbar requires matrix ops expressed as Matrix-Vector Multiplications (MVM) that use bit-serial compute units [15], [16]. To address this, we design a functional simulator using PyTorch that implements the *conv2d* (convolution) and *linear* (fully connected) layers based on the crossbar-based computation (*conv2d-mvm, linear-mvm*).

Table 5.3. Functional simulator parameters

Component	Parameters (architecture parameters in *italics*)
Iterative-mvm	Input feature size, Kernel size, Input channels, Output channels, Padding, Stride
Tiling	*Crossbar size*
Bit-slicing	Input bits, Weight bits, *Accumulator width* *ADC bits*, *Stream width*, *Slice Width*
GENIEx	*Crossbar size*, R_{on}, R_{off}, R_{source}, R_{sink}, R_{wire}

Functional Simulator. As shown in Figure 5.6, the execution of a convolution layer is divided into three phases within the functional simulator: *Iterative-mvm*, *Tiling*, and *Bit-slicing*. Each phase depends on parameters that either capture the layer or architecture details pertinent to MVMs. Consequently, we extract the analog computing aspect of crossbar hardware and ignore any impact of memory and communication. First, Iterative-mvm expresses a convolution as repeated MVMs, where the weights forms the matrix and a block of pixels across all input channels form the vector (for an iteration). Each iteration produces an output vector which is comprised of one pixel from all output channels. Second, Tiling expresses the weight-matrix as a combination of several sub-matrices (or tiles) where, each sub-matrix's size equals the crossbar size. A slice of input vector is shared by tiles in a row. Tiles in a column produce partial sums, which are added together to produce a slice of the convolution output. Third, Bit-slicing (both input and weight bits) expresses the bit-serial nature of crossbar computations [15], [16]. We will refer to a bit-slice ($\geq$ 1 bits) of inputs and weights as stream and slice, respectively. Within each step, an input stream is applied to a crossbar's rows to produce ADC outputs. Next, the shift-and-add units merge the ADC outputs of different weight slices. Eventually, the outputs of successive input streams go through shift-and-add units to produce the partial sums for a tile. Depending on the simulation mode (ideal or non-ideal), the ADC outputs are generated either by actual dot-product computation or a forward pass of GENIEx discussed in Section 5.4. *In summary, the three phases together provide the projection of a layer's execution on actual crossbar hardware.*

PyTorch Modelling. The weight-matrix and input-vectors are modelled as multi-dimensional tensors of shape - $(Slices, Tr, Tc, Xr, Xc)$, and $(Batch\ Size, Tr, Xr, Streams)$, respectively. Here, the symbols - T, X, r, and c refer to a tile, crossbar, row and column. Accordingly, Tr refers to a "tile row" and so on. The tensor operations *torch.mul* and *torch.sum* execute the individual crossbar operations. Subsequently, reduction across the weight slices ($Slices$) and input streams ($Streams$) with scalar factors for shift-and-add generates the partial products. Subsequently, reduction across Tr dimension produces the convolution output. Multiple input vectors corresponding to different iterations of MVM are implemented as a batch of vectors ($Batch\ Size$). Table 5.3 lists the layer and architecture parameters supported by the functional simulator.

5.6 Experimental Methodology

Crossbar: We simulate memristive crossbars using HSPICE. The test vectors for V and G are collected from the dataset (CIFAR-100 and ImageNet) and the pretrained neural network models (ResNet) respectively. $I_{non-ideal}$ obtained from SPICE simulations is used to calculate the non-ideality ratio, f_R, described in Section 5.4. Finally, V, G and $f_R(V, G)$ are normalized to the range [0,1] to form the training set for GENIEx. To verify the generalization and applicability of GENIEx, we generated datasets for crossbar configurations with different design parameters such as crossbar size (16, 32, 64), ON resistance (50kΩ, 100kΩ, 300kΩ), and conductance ON/OFF ratio (2, 6, 10). The non-ideality parameters are $R_{source} = 500\Omega/1000\Omega$, $R_{sink} = 100\Omega/500\Omega$, $R_{wire} = 2.5\Omega$ per cell. The device parameters are $d_0 = 0.25nm$, $V_0 = 0.25V$, $I_0 = 0.1mA$ [179], [180].

Functional Simulator: The precisions of different components of the functional simulators are as follows: accumulator = 32-bit (24 fractional), ADC = 14-bit, inputs and weights = 16-bit (13 fractional), input Streams = 4-bit, weight Slices = 4-bit, unless otherwise specified. All networks use fixed-point (FxP) representations.

DNN: GENIEx has 500 hidden layer neurons and ReLU non-linearity [148]. We use PyTorch to evaluate large-scale neural networks on the functional simulator using GENIEx. For the CIFAR-100 dataset, we use the network architecture ResNet-20. For the ImageNet

dataset, we considered ResNet-18 on a subset of 7680 test images of the dataset. We report the top-1 accuracies for both datasets. The ideal floating point 32-bit (FP) accuracies for CIFAR-100 and subset of ImageNet are 69.6% and 76.01% respectively.

5.7 Results

5.7.1 Impact on Design Parameters

First, we study the impact of non-idealities on the classification accuracy of DNNs under different design considerations of crossbar sizes, ON resistances, and conductance ON/OFF ratio. The studies are performed on ResNet-20 for CIFAR-100 dataset with the features of bit-slicing and bit-streaming using 4-bit Streams and Slices. The weights and activations for these networks have been considered as 16-bit fixed point representations.

We observe in Figure 5.7 (a) that the classification accuracy degrades by 12% for a 64×64 crossbar compared to an ideal 16 bit fixed-point (Ideal FxP) implementation. However, for lower crossbar sizes like 16×16, the degradation is $\leq 1\%$. This is due to reduced effective resistance for higher crossbar sizes, as discussed in Section 5.3. For higher ON resistances such as 300kΩ, we observe, in Figure 5.7 (b) that the accuracy degradation is 7.6% lower than the case with $100k\Omega$. This is because the parasitic resistances have more pronounced effect on crossbars with lower ON resistances, resulting in higher accuracy degradation. In 5.7 (c), we observe that for a given ON resistance ($100k\Omega$ in this case), lower ON/OFF ratio like 2 results in upto 46% degradation in accuracy due to the average resistances in the crossbar being low. With higher ON/OFF ratio such as 10, the accuracy degradation reduces to 8.6%.

Figure 5.7 (d) shows that there is a significant difference between the accuracies predicted by an analytical model and GENIEx. Further, it also illustrates that an analytical model overestimates the accuracy degradation by 12.34% for supply voltage, $V_{supply} = 0.25V$ and 11.6% for $V_{supply} = 0.5V$ compared to GENIEx. Note that the analytical model considers only linear non-idealities (parasitic resistances). *It underscores that the device non-linearity which is captured by our model can push the behavior of the crossbar towards ideality, thus resulting in a lower accuracy degradation.*

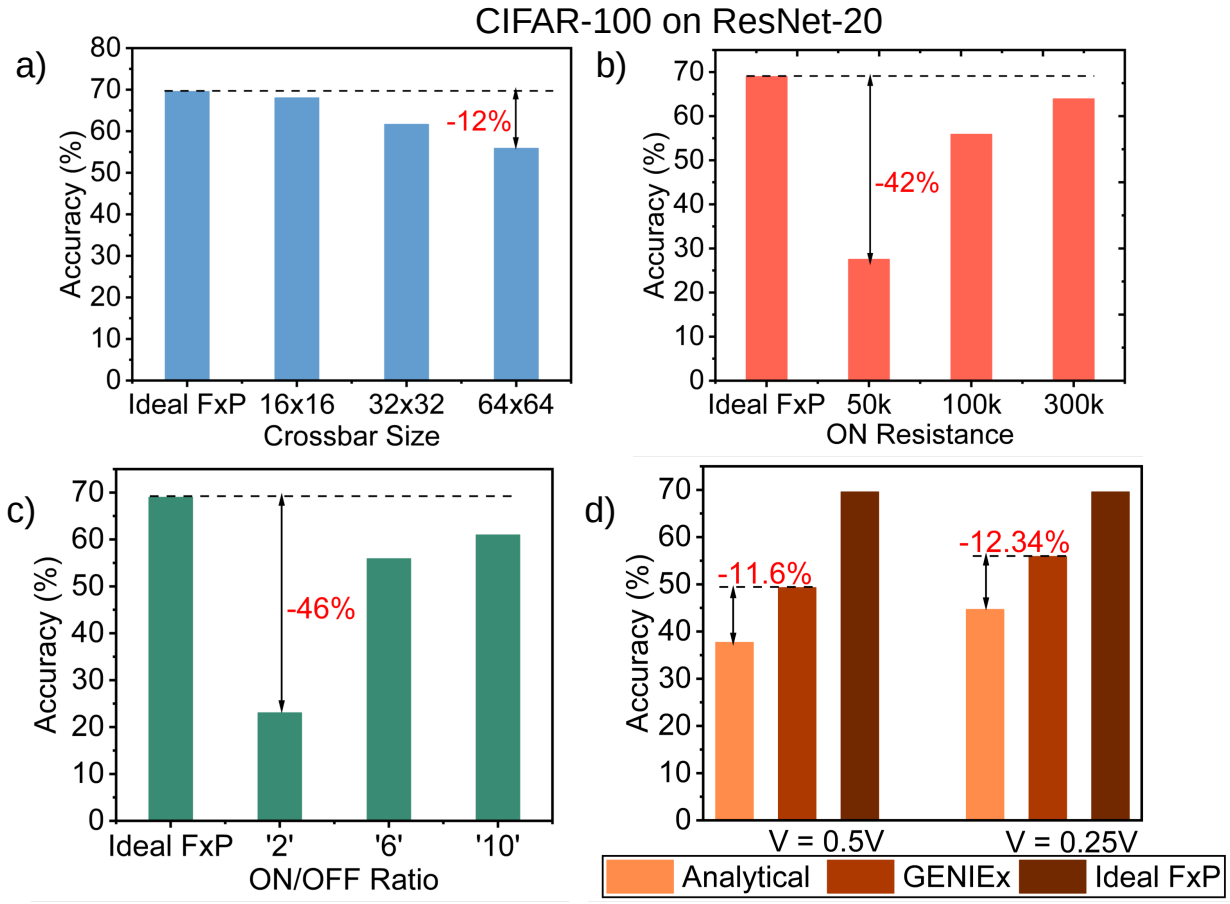


Figure 5.7. Impact of non-idealities with crossbar design parameters (a) Crossbar Size, (b) ON resistance, (c) ON/OFF ratio. (d) Comparison between analytical model and GENIEx.

5.7.2 Impact of Quantization

We study the effect of non-idealities on DNNs with different bit-precision for weights and activations. We consider 3 cases for networks where the weights and activations are 16-bit, 8-bit and 4-bit fixed point representations: i) Ideal, ii) Non-idealities estimated by analytical model, and iii) Non-idealities estimated by GENIEx. Figure 5.8 shows that when the weights and activations are represented as 16-bit fixed point, the classification accuracy is close to ideal 32-bit floating point accuracy. When the precision of the weights and activations is reduced to 8-bit, the accuracy degrades by 30.03% for CIFAR-100, and 35.29%

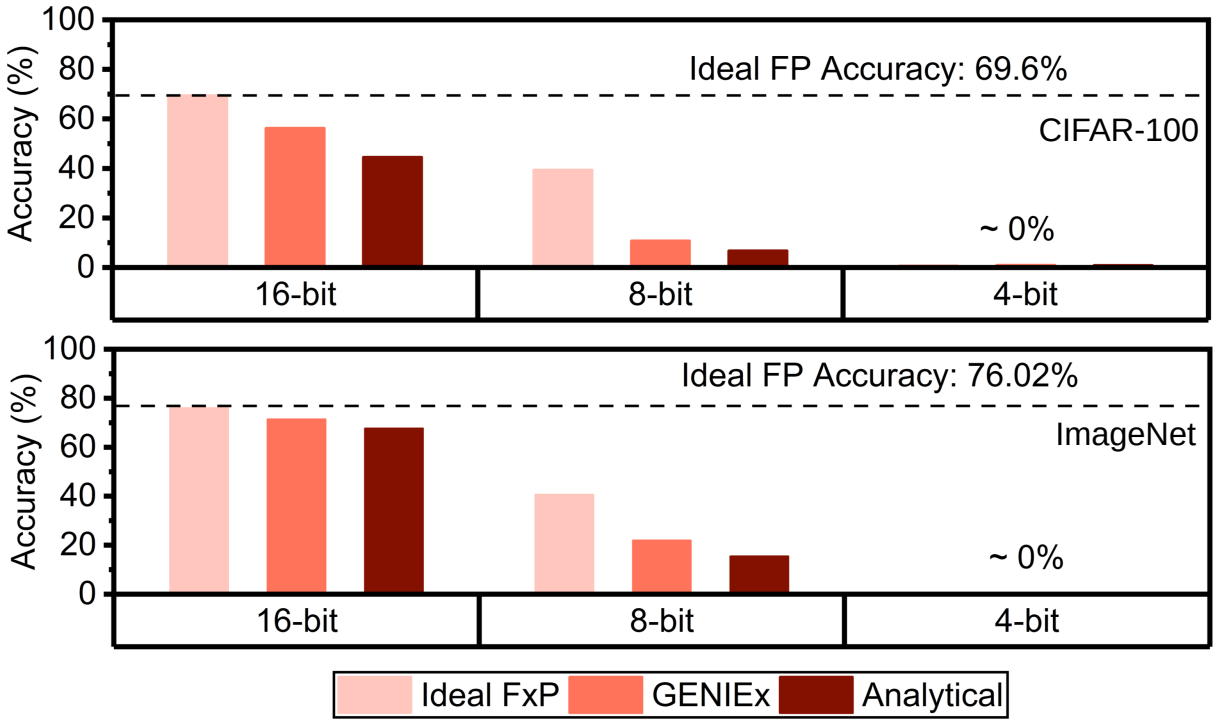


Figure 5.8. Impact of precision of weights and activations on classification accuracy under the influence of non-idealities.

for ImageNet. For 4-bit case, the accuracy is $\simeq$ 0%. Further, the accuracy degradation increases from 12.5% to 29.6% for CIFAR-100 and 4.54% to 17.67% for ImageNet when the bit-precision is reduced from 16-bit to 8-bit. Thus, non-idealities have an increased detrimental effect at lower bit-precisions. Note that the analytical models overestimate the degradation in accuracy by 12.34% and 3.99% for CIFAR-100, and 3.70% and 6.49% for ImageNet compared to GENIEx for 16-bit and 8-bit cases respectively. *This result shows that due to the constraints on a) the number of bits NVM devices can store reliably [165], and b) the DAC precisions for efficient MVM [16], bit slicing of weights and inputs are essential features to achieve close to full-precision accuracies.*

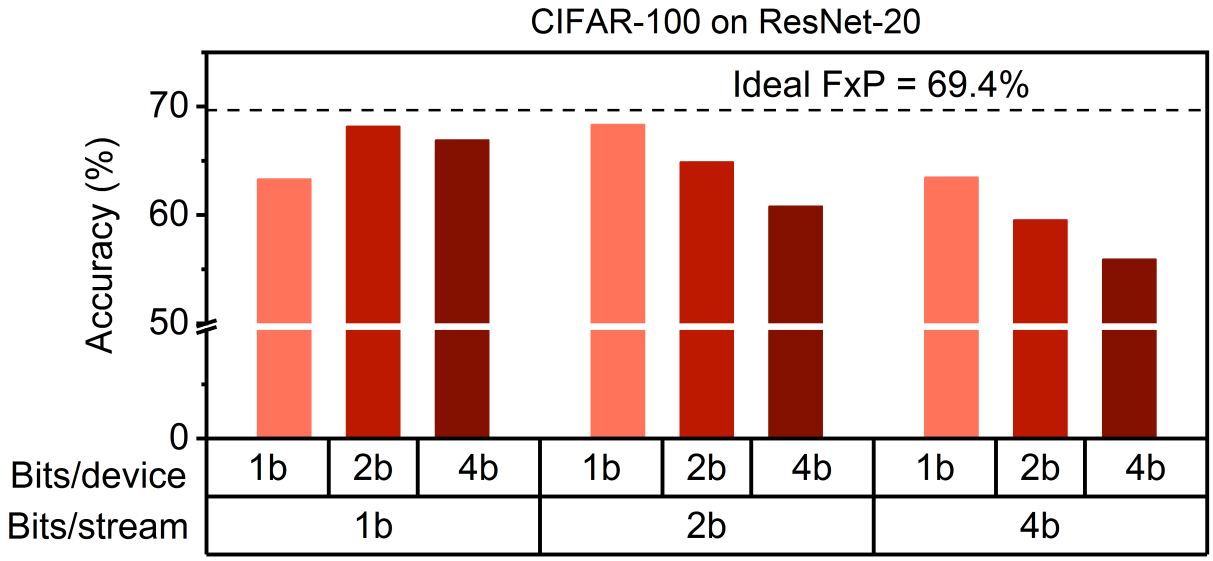


Figure 5.9. Impact of number of bits/device and bits/stream.

5.7.3 Impact of Bit Slicing

Finally, we study the impact of different bit-slicing configurations for inputs (Streams) and weights (Slices) for 16-bit FxP network on the classification accuracy of DNNs in presence of non-idealities. Figure 5.9 shows that using 2-bit or 1-bit Streams and Slices, achieves close to ideal FxP accuracy. Increasing the Stream and Slice widths to 4-bit results in a 12.48% degradation in accuracy. Note that 1-bit Streams and 1-bit Slices result in a slightly lower accuracy. This is because the combination of 1-bit Streams and Slices results in very high sparsity that makes the crossbar resilient to parasitic resistances. In such a case, device non-linearity can lead to non-ideality factor, NF to be lower than 0, resulting in lower accuracy. Nonetheless, using lower number of bits per slice and stream can help achieve close to ideal FxP accuracies. *This result provides a perspective on architectural design parameters such as the Slice and Stream widths in presence of crossbar non-idealities.*

5.8 Conclusion

We present GENIEx, a generalized approach to emulating non-ideality in memristive crossbars using neural networks. We perform extensive SPICE simulations and subsequently train a neural network to learn a generalized behavior of the non-ideal crossbar. Finally, we use GENIEx in a functional simulator for evaluating the impact of these non-idealities on the image classification performance of large-scale DNNs. We show that GENIEx achieves a *low* RMSE of 0.25 for $V_{supply} = 0.25V$ and 0.7 for $V_{supply} = 0.5V$ with respect to HSPICE, which is 7× and 12.8× lower, respectively, than an analytical model. This is due to the ability of GENIEx to model both linear and non-linear non-idealities. We further show that an analytical model overestimates the degradation in classification accuracy by 12.3% on CIFAR-100, and 4% on ImageNet compared to GENIEx. We analyze the impact of non-idealities on the crossbar design parameters such as crossbar-size, ON resistance, conductance ON/OFF ratio, Stream width, and Slice width. We observe that packing lower bits per device as well as using low crossbar sizes with higher ON resistances is necessary to minimize the impact of non-idealities. The proposed end to end framework for evaluating crossbar based architectures on realistic crossbars can pave the way for efficient crossbar designs for future machine learning systems.

6. 8T SRAM CELL AS A MULTIBIT DOT-PRODUCT ENGINE FOR BEYOND VON NEUMANN COMPUTING

6.1 Introduction

State-of-the-art computing platforms are widely based on the von-Neumann architecture [181]. The von-Nuemann architecture is characterized by distinct spatial units for *computing* and *storage*. Such physically separated memory and compute units result in huge energy consumption due to frequent data transfer between the two entities. Moreover, the transfer of data through a dedicated limited-bandwidth bus limits the overall compute throughput. The resulting memory bottleneck is *the major throughput concern* for hardware implementations of data intensive applications like machine learning, artificial intelligence *etc.*

A possible approach geared towards high throughput beyond von-Neumann machines is to enable distributed computing characterized by tightly intertwined storage and compute capabilities. If computing can be performed inside the memory array, rather than in a spatially separated computing core, the compute throughput can be considerably increased. As such, one could think of *ubiquitous* computing on the silicon chip, wherein both the logic cores and the memory unit partake in compute operations. Various proposals for *'in-memory'* computing with respect to emerging non-volatile technologies have been presented for both dot product computations [182], [183] as well as vector Boolean operations [184]. Prototypes based on emerging technologies can be found in [85], [183] .

With respect to the CMOS technology, Boolean in-memory operations have been presented in [185] and [186]. In [185] authors have presented vector Boolean operations using 6T SRAM cells. Additionally, authors in [186] have demonstrated that the 8 transistor (8T) SRAM cells lend themselves easily as vector compute primitives due to their decoupled read and write ports. Both the works [185] and [186] are based on vector Boolean operations. Interestingly, by adding additional peripheral circuits more complex functions like add and multiplication can be implemented using bulk-bit-wise computations as shown in [187]. However, perhaps the most frequent and compute intensive function required for numerous applications like machine learning is the *dot product* operation. Memristors based on resistive-RAMs (Re-RAMs) have been reported in many works as an analog dot product

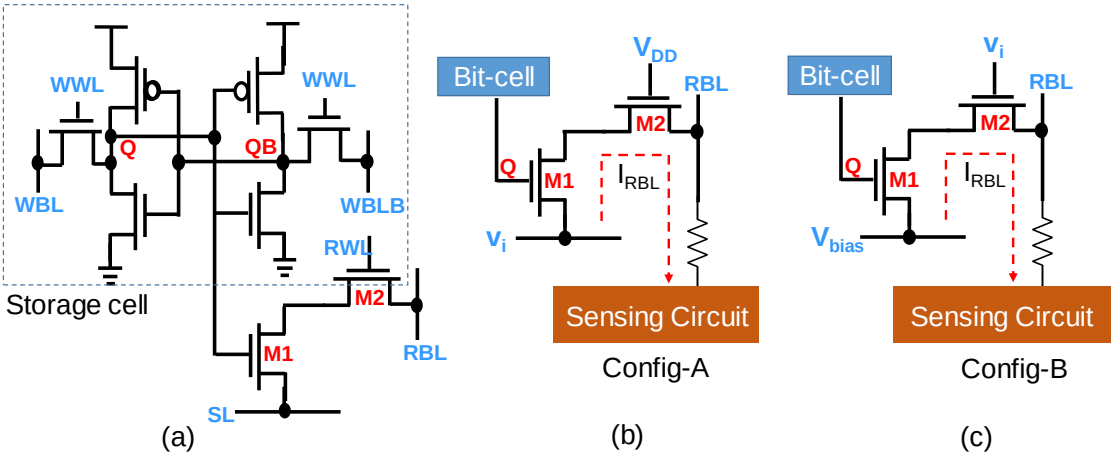


Figure 6.1. (a) Schematic of a standard 8T-SRAM bit-cell. It consists of two decoupled ports for reading and writing respectively. (b) First proposed configuration (Config-A) for implementing the dot product engine using the 8T-SRAM bit-cell. The SL is connected to the input analog voltage v_{i}, and the RWL is turned ON. The current I_{RBL} through the RBL is sensed and is proportional to the dot product $v_{\mathrm{i}} \cdot g_{\mathrm{i}}$, where g_{i} is the ON/OFF conductance of the transistors $M1$ and $M2$. (c) Second proposed configuration (Config-B). The input analog voltages are applied to the RWL, while the SL is supplied with a constant voltage V_{bias}. The current through the RBL is sensed in the same way as in Config-A.

compute engine [122], [184]. Few works based on analog computations in SRAM cells can be found in [188]–[191]. These works use 6T SRAM cells and rely on the resultant accumulated voltage on the bit-lines (BLs). Not only 6T SRAMs are prone to read-disturb failures, the failures are also a function of the voltage on the BLs. This leads to a tightly constrained design space for the proposed 6T SRAM based analog computing. More recently, there have been works on modified SRAM cells such as 8T or 10T cells for more robust in-memory computing [11], [192]–[194]. These works use a modified SRAM cell enabling parallel computations of binary dot products. In general, the existing analog computing works in SRAM cells have implemented either multi-bit multiplication or have been limited to binary dot products. Interestingly, using current based computations allow highly parallel multi-bit dot product operation. 8T SRAM cells as current based dot product accelerators were first proposed in [195]. Subsequently, the work in [14] used a twin 8T cell to enable multi-bit dot products using current based computations. Based on [195], in this thesis, we employ

8T cells that are much more robust as compared to the 6T cells due to isolated read port. We show that without modifying the basic bit-cell for the 8T SRAM cell, it is possible to configure the 8T cell for in-memory multi-bit dot product computations. Note, in sharp contrast to the previous works on in-memory computing with the CMOS technology, we enable *current based, analog-like* dot product computations using robust digital 8T bit-cells.

The key highlights of the present work are as follows:

1. We show that the conventional 8T SRAM cell can be used as a primitive for analog-like dot product computations, without modifying the bit-cell circuitry. In addition, we present two different configurations for enabling dot product computation using the 8T cell.

2. Apart for the sizing of the individual transistors consisting the read port of the 8T cell, the basic bit-cell structure remains unaltered. Thereby, the 8T SRAM array can also be used for usual digital memory read and write operations. As such, the presented 8T cell array can act as a dedicated dot product engine or as an *on-demand* dot product accelerator.

3. A detailed simulation analysis using 45nm predictive technology models including layout analysis and effect of non-idealities like the existence of line-resistances and variation in transistor threshold voltages has been reported highlighting the various trade-offs presented by each of the two proposed configurations.

6.2 8T-SRAM as a Dot Product Engine

A conventional 8T bit-cell is schematically shown in Fig. 6.1(a). It consists of the well-known 6T-SRAM bit-cell with two additional transistors that constitute a decoupled read port. To write into the cell, the write word-line (WWL) is enabled, and write bit-lines (WBL/WBLB) are driven to V_{DD} or ground depending on the bit to be stored. To read a value from the cell, the read bit-line (RBL) is pre-charged to V_{DD} and the read word-line (RWL) is enabled. Note, that the source-line (SL) is connected to the ground. Depending on whether the bit-cell stores a logic '1' or '0', the RBL discharges to 0V or stays at V_{DD},

respectively. The resulting voltage at the RBL is read out by the sense amplifiers. Although 8T-cells incur a ∼30% increase in bit-cell area compared to the 6T design, they are read-disturb free and more robust due to separate read and write path optimizations [196].

We now show how such 8T-SRAMs, with no modification to the basic bit-cell circuit (except for the sizing of the read transistors), can behave as a dot product engine, without affecting the stability of the bits stored in the SRAM cells. We propose two configurations - *Config-A* and *Config-B*, for enabling dot-product operations in the 8T-SRAMs. Config-A is shown in Fig. 6.1(b). The inputs v_i (encoded as analog voltages) are applied to the SLs of the SRAM array, and the RWL is also enabled. The RBL is connected to a sensing circuitry, which we will describe later. Thus, there is a static current flow from the SL to the RBL, which is proportional to the input v_i and the conductance of the two transistors $M1$ and $M2$. For simplicity, assume that the weights (stored in the SRAM) have a single-bit precision. If the bit-cell stores '0', the transistor $M1$ is OFF, and the output current through the RBL is close to 0. Whereas if the bit-cell stores a '1', the current is proportional to $v_i \cdot g_{ON}$, where g_{ON} is the series 'ON' conductance of the transistors. Assume similar inputs v_i are applied on the SLs for each row of the memory array. Since the RBL is common throughout the column, the currents from all the inputs v_i are summed into the RBL. Moreover, since the SL is common throughout each row, the same inputs v_i are supplied to multiple columns. Thus, the final output current through RBL of each column is proportional to $I^j_{RBL} = \Sigma(v_i \cdot g^j_i)$, where g^j_i is the 'ON' or 'OFF' conductance of the transistors, depending on whether the bit-cell in the i-th row and j-th column stores a '1' or '0', respectively. The output current vector thus resembles the vector-matrix dot product, where the vector is v_i in the form of input analog voltages, and the matrix is g^j_i stored as digital data in the SRAM.

Let us now consider a 4-bit precision for the weights. If the weight $W^j_i = w_3 w_2 w_1 w_0$, where w_i are the bits corresponding to the 4-bit weight, the vector matrix dot product becomes:

$$\Sigma(v_i \cdot W^j_i) = \Sigma[v_i \cdot (2^3 w_3 + 2^2 w_2 + 2^1 w_1 + w_0)]$$
$$= \Sigma(v_i \cdot 2^3 w_3) + \Sigma(v_i \cdot 2^2 w_2) + \Sigma(v_i \cdot 2^1 w_1) + \Sigma(v_i \cdot w_0)$$

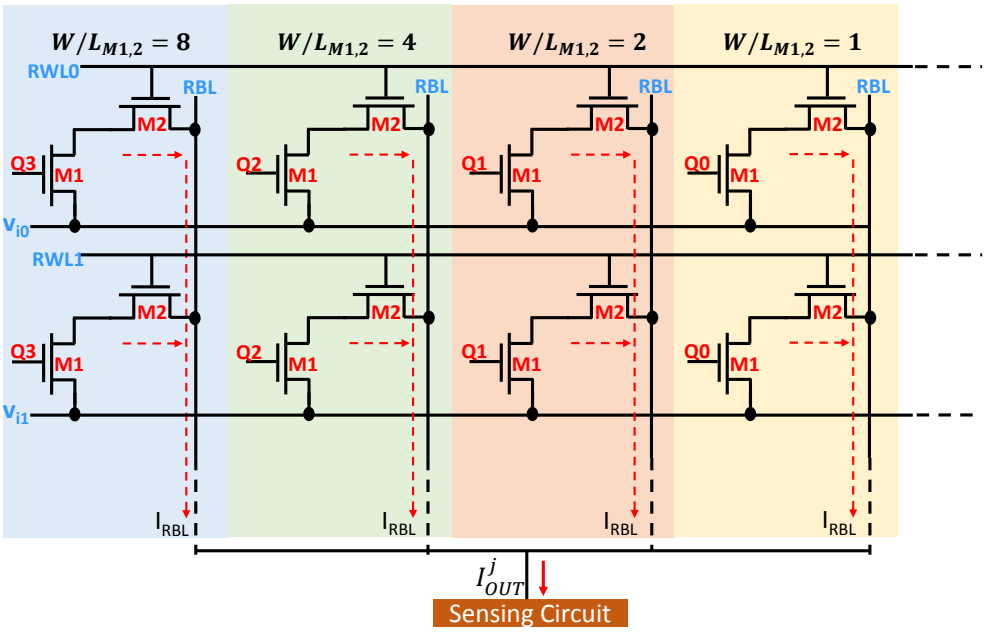


Figure 6.2. 8T-SRAM memory array for computing dot-products with 4-bit weight precision. Only the read port is shown, the 6T storage cell and the write port are not shown. The array columns are grouped in four, and the transistors $M1$ and $M2$ are sized in the ratio $8:4:2:1$ for the four columns. The output current I^{j}_{OUT} represents the weighted sum of the I_{RBL} of the four columns, which is approximately equal to the desired dot-product.

Now, if we size the read transistors $M1$ and $M2$ of the SRAM bit-cells in column 1 through 4 in the ratio $2^3 : 2^2 : 2^1 : 1$, as shown in Fig. 6.2, the transistor conductances in the 'ON' state would also be in the ratio $2^3 : 2^2 : 2^1 : 1$. Thus, summing the currents through the RBLs of the four columns yields the required dot product in accordance to the equation shown above. This sizing pattern can be repeated throughout the array. In addition, one could also use transistors having different threshold voltages to mimic the required ratio of conductances as $2^3 : 2^2 : 2^1 : 1$. Note that, the currents through the RBLs of the four consecutive columns are summed together, thus we obtain one analog output current value for every group of four columns. In other words, the digital 4-bit word stored in the SRAM array is multiplied by the input voltage v_{i} and summed up by analog addition of the currents on the RBLs. This *one-go* computation of vector multiplication and summation in a digital memory array would result in high throughput computations of the dot products.

It is worth mentioning, that the way input v_{i} are multiplied by the stored weights and summed up is reminiscent of memristive dot product computations [122]. However, a concern with the presented SRAM based computation is the fact that the ON resistance of the transistors (few kilo ohms) are much lower as compared to a typical memristor ON resistance which is in the range of few tens of kilo ohms [22]. As such the static current flowing through the ON transistors $M1$ and $M2$ would typically be much higher in the presented proposal. In order to reduce the static current flow, we propose scaling down the supply voltage of the SRAM cell. Note, interestingly, 8T cells are known to retain their robust operation even at highly scaled supply voltages [197]. In the next section we have used a V_{DD} lower than the nominal V_{DD} of 1V. We would now describe another way of reducing the current, although with trade-offs, as detailed below.

Config-B is shown in Fig. 6.1(c). Here, the SLs are connected to a constant voltage V_{bias}. The input vector v_{i} is connected to RWLs, i.e., the gate of $M2$. Similar to Config-A, the output current I_{RBL} is proportional to v_{i}. We will later show from our simulations that for a certain range of input voltage values, we get a linear relationship between I_{RBL} and v_{i}, which can be exploited to calculate the approximate dot product. To implement multi-bit precision, the transistor sizing is done in the same way as Config-A as represented in Fig. 6.2, so that the I_{RBL} is directly proportional to the transistor conductances. Key features of the proposed Config-B are as follows. The input voltages v_{i} have a capacitive load, as opposed to a resistive load in Config-A. This relaxes the constraints on the input voltage generator circuitry, and is useful while cascading two or more stages of the dot product engine. However, as presented in the next section, Config-B has a small non-zero current corresponding to zero input as opposed to Config. A that has zero current for zero input.

In order to sense the output current at the RBLs, we use a current to voltage converter. This can most simply be a resistor, as shown in Fig. 6.1. However, there are a few constraints. As the output current increases, the voltage drop across the output resistor increases, which in turn changes the desired current output. A change in the voltage on the RBL would also change the voltage across the transistors $M1$ and $M2$, thereby making their conductance a function of the voltage on the RBL. Thus, at higher currents corresponding to multiple rows of the memory array, the I_{RBL} does not approximate the vector-matrix dot product, but

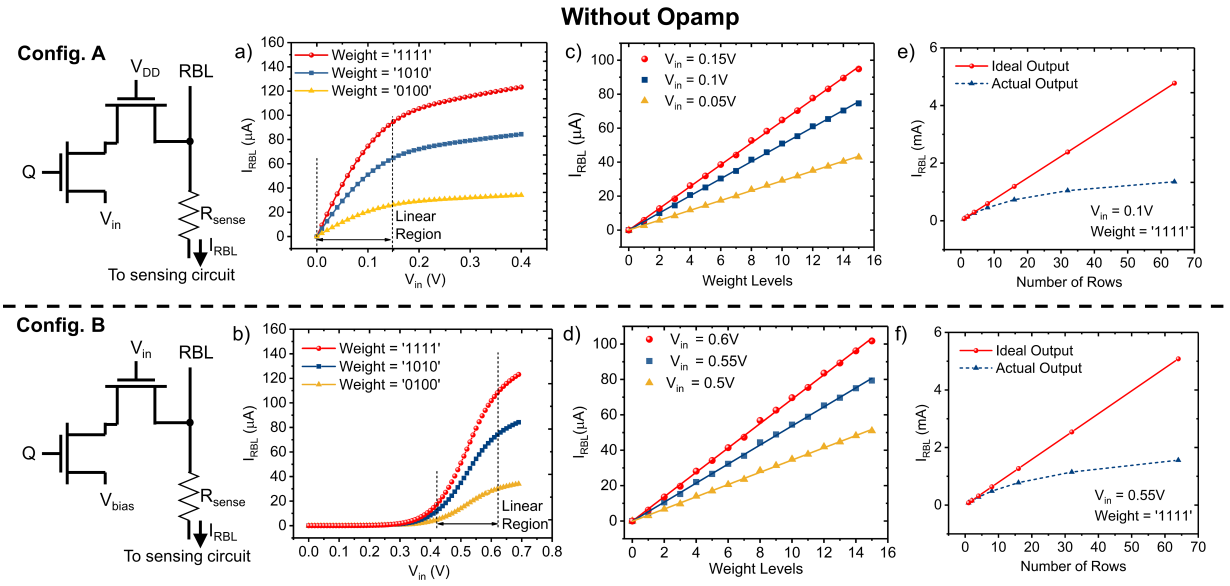


Figure 6.3. I_{RBL} versus V_{in} characteristics for (a) Config. A and (b) Config. B shows the linear region of operation for different weights. I_{RBL} versus Weight levels for (c) Config. A and (d) Config. B shows desirable linear relationship at various voltages V_{in}. I_{RBL} shows significant deviation from ideal output ($I_N = N \times I_1$ with increasing number of rows for both (e) Config. A and (f) Config. B, where I_1 is the current corresponding to one row and N is the number of rows. The analyses were done for $V_{DD} = 0.65V$

deviates from the ideal output. This dependence of the RBL voltage on the current I_{RBL} will be discussed in detail in the next section with possible solutions.

6.3 Results

The operation of the proposed configurations (Config-A and Config-B) for implementing a multi-bit dot product engine was simulated using HSPICE on the 45nm PTM technology [198]. For the entire analysis, we have used a scaled down V_{DD} of 0.65V for the SRAM cells. The main components of the dot-product engine implementation are the input voltages and conductances of the transistors for different states of the cells. A summary of the analysis for the two configurations is presented in Fig. 6.3. In Fig. 6.3, we have assumed a sensing resistance of 50-ohms connected to the RBL. Note, a small sense resistance is required to ensure that the voltage across the sensing resistance is not high enough to drastically alter the conductances of the connected transistors $M1$ and $M2$.

In Fig. 6.3(a)-(b) we plot the output current in RBL (I_{RBL}) as a function of the input voltage for three 4-bit weight combinations '1111', '1010' and '0100' for the two different configurations described in the previous section. The results presented are for a single 4-bit cell. To preserve the accuracy of a dot-product operation, it is necessary to operate the cell in the voltage ranges such that the current is a linear function of the applied voltage v_{i}. These voltage ranges are marked as linear region in Fig. 6.3(a)-(b). The slope of the linear section I_{RBL} versus V_{in} plot varies with weight, thus signifying a dot product operation. Further, at the left voltage extremity of the linear region, I_{RBL} tends to zero irrespective of the weight, thus satisfying the constraint that the output current is zero for zero V_{in}. It is to be noted that the two configurations show significantly different characteristics due to the different point-of-application of input voltages.

Fig. 6.3(c)-(d) presents the dependence of the current I_{RBL} on the 4-bit weight levels for Config-A at constant voltages V_{in} = 0.05V, 0.1V, 0.15V and configuration B at V_{in} = 0.5V, 0.55V, 0.6V, respectively. Different voltages were chosen so as to ensure the circuit operates in the linear region as depicted by Fig. 6.3(a)-(b). Desirably, I_{RBL} shows a linear dependence on weight levels and tends to zero for weight = '0000'. The choice of any voltage in the linear regions of Fig 6.3(a)-(b) does not alter the linear dependence of the I_{RBL} on weight levels.

To expand the dot-product functionality to multiple rows, we performed an analysis for upto 64 rows in the SRAM array, driven by 64 input voltages. In the worst case condition, when the 4-bit weight stores '1111', maximum current flows through the RBLs, thereby increasing the voltage drop across the output resistance. Fig. 6.3(e)-(f) indicates that the total current I_{RBL} deviates from its ideal value with increasing number of rows, in the worst case condition. The deviation in Fig. 6.3(e)-(f) is because we sense the output current with an equivalent sensing resistance (R_{sense}) and hence the final voltage on the bit-line (V_{BL}) is dependent on the current I_{RBL}. At the same time, I_{RBL} is also dependent on V_{BL} and as a result the effective conductance of the cell varies as V_{BL} changes as a function of the number of rows. It was also observed that the deviation reduces with decreasing sensing resistance as expected. Another concern with respect to Fig. 6.3 is the fact that the total summed

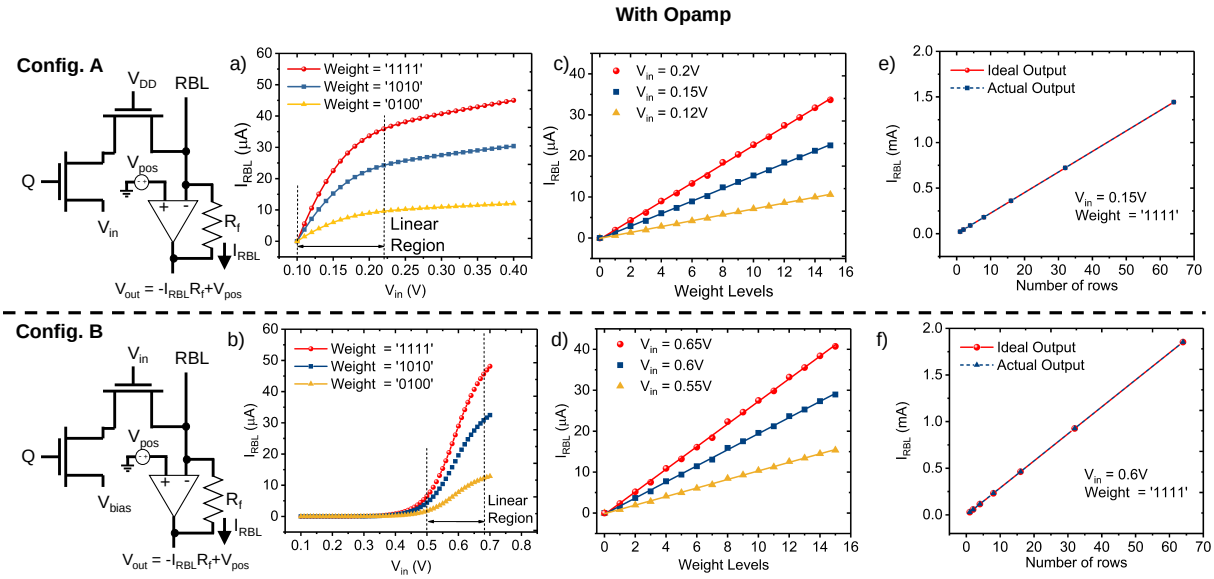


Figure 6.4. I_{RBL} versus V_{in} characteristics for (a) Config. A and (b) Config. B shows the linear region of operation for different weights. I_{RBL} versus weight levels for (c) Config. A and (d) Config. B shows desirable linear relationship at various voltages V_{in}. I_{RBL} shows almost zero deviation from ideal output ($I_N = N \times I_1$ with increasing number of rows for both (e) Config. A and (f) Config. B, where I_1 is the current corresponding to one row and N is the number of rows. These analyses were done for $V_{DD} = 0.65V$

up current reaches almost 6mA for 64 rows for the worst case condition (all the weights are '1111').

There are several ways to circumvent the deviation from ideal behavior with increasing number of simultaneous row accesses and also reduce the maximum current flowing through the RBLs. One possibility is to use an operational amplifier (Opamp) at the end of each 4-bit column, where the negative differential input of the Opamp is fed by the bit-line corresponding to a particular column. Whereas, the positive input is supplemented by a combination of the Opamp offset voltage and any desired voltage required for suitable operation of the dot-product as shown in left hand side of Fig. 6.4. Opamp provides a means of sensing the summed up current at the RBL while maintaining a constant voltage at the RBL. Opams in the configuration as shown in Fig. 6.4 have been traditionally used for sensing in memristive crossbars as in [183].

We performed the same analysis as previously described in Fig. 6.3 for the two proposed configurations with the bit-line terminated by an Opamp. For our analysis, we have set $V_{pos} = 0.1V$ for the positive input of the Opamp and thus analysis is limited to input voltages above V_{pos} to maintain the unidirectional current. Note, we have used an ideal Opamp for our simulations, where the voltage V_{pos} can be accounted for both the non-ideal offset voltage of the Opamp and a combination of an externally supplied voltage. Fig. 6.4(a)-(b) shows the plot of I_{RBL} versus input voltage V_{in} for the two configurations. Similar behavior as in the case of Fig. 6.3(a)-(b) is observed even in the presence of the Opamp. However, note that the current ranges have decreased since RBL is now clamped at V_{pos}. Further, the dot-product operation is only valid for $V_{in} > V_{pos}$ and thus the acceptable input range is shifted in the presence of an Opamp. Fig. 6.4(c)-(d) shows the behavior of I_{RBL} versus weight levels for the two configurations and desirably, linearity is preserved.

Fig. 6.4(e)-(f) presents the current through the RBL as a function of the number of rows. As expected, due to the high input impedance of the Opamp, and the clamping of V_{BL} at a voltage V_{pos} the deviation of the summed up current from the ideal value have been mitigated to a huge extent. Although, the current levels have reduced significantly as compared to the Fig. 6.3, the resultant current for 64 rows would still be higher than the electro-migration limit for the metal lines constituting the RBL [199]. One possible solution is to sequentially access a smaller section of the crossbar (say 16 or 8 rows at a time), convert the analog current into its digital counterpart each time and finally add all accumulated digital results. In addition use of high threshold transistors for the read port of the SRAM would also help to reduce the maximum current values. Further, the maximum current is obtained only when all the weights are '1111', which is usually not true due to the sparsity of matrices involved in various applications as in [36], [200].

We also performed functional simulations using the proposed dot-product engine based on Config. A in a fully connected artificial neural network consisting of 3 layers as shown in Fig. 6.5. The main motivation behind this analysis is to evaluate the impact of the non-linearity in the I-V characteristics on the inference accuracy of the neural network. We chose an input voltage range of 0.1-0.22V. As can be observed in Fig. 6.4(a), the I-V characteristics are not exactly linear within this range, as such a network level functional simulation is required

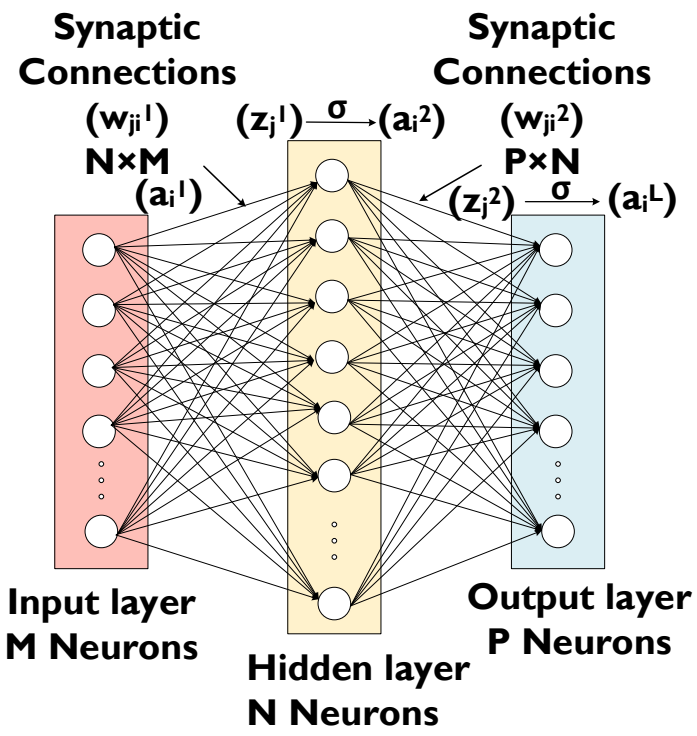


Figure 6.5. Fully connected network topology consisting of 3 layers, the input layer, the hidden layer and the output layer [22]. We have used M=784, N = 500 and P = 10.

to ascertain the impact of the non-linearity on classification accuracy. The network details are as follows. The hidden layer consisted of 500 neurons. The network was trained using the Backpropagation algorithm [201] on the MNIST digit recognition dataset under ideal conditions using MATLAB ®Deep Learning Toolbox[163].

During inferencing, we incorporated the proposed 8T-SRAM based dot-product engine in the evaluation framework by discretizing and mapping the trained weights proportionally to the conductances of the 4-bit synaptic cell. The linear range of the voltage was chosen to be [0.1-0.22V] and normalized to a range of [0 1]. The dot-product operation was ensured by normalizing the I-V characteristics for all the weight levels such that current corresponding to the highest input voltage and highest weight level is $I_{max} = V_{max} \times G_{max}$. The activation function of the neuron was considered to be a behavioral *satlin* function scaled according to the scaling factor of the weights to preserve the mathematical integrity of the network. To be noted, the normalization of current and input voltage simplifies the scaling of the neuron

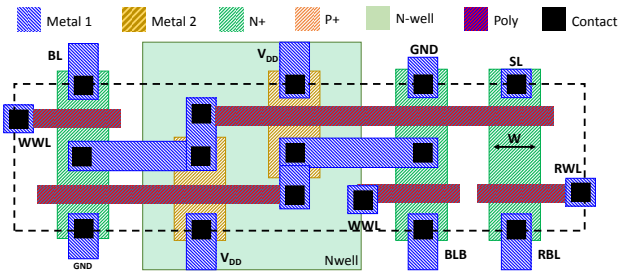


Figure 6.6. Thin-cell layout for a standard 8T-SRAM bit-cell [196].

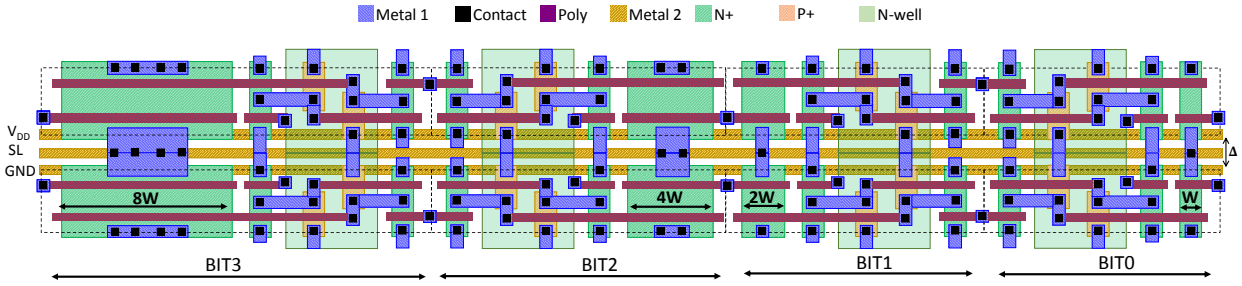


Figure 6.7. Thin-cell layout for the proposed 8T-SRAM array with 4-bit precision weights. The width of read transistors of different bit positions are sized in the ratio 8:4:2:1. An additional metal line for SL is also required, which runs parallel to the power-lines. This incurs an area overhead of ∼29.4% compared to the standard 8T-SRAM bit-cell.

activation function. The accuracy of digit recognition task was calculated to be merely 0.11% lower than the ideal case (98.27%) thus indicating that the proposed dot-product engine can be seamlessly integrated into the neural network framework without significant loss in performance.

Further, it is to be noted that in many cases the inherent resilience of the applications that require dot product computations can be leveraged to circumvent some of the circuit level non-idealities. For example, for cases like training and inference of an artificial or a spiking neural network, various algorithmic resilience techniques can be applied where modeling circuit non-idealities and modifying the standard training algorithms[22], [147] can help preserve the ideal accuracy of the classification task concerned. Additionally, the proposed technique can either be used as a dedicated CMOS based dot product compute engine or as an on-demand dot product accelerator, wherein the 8T array acts as usual

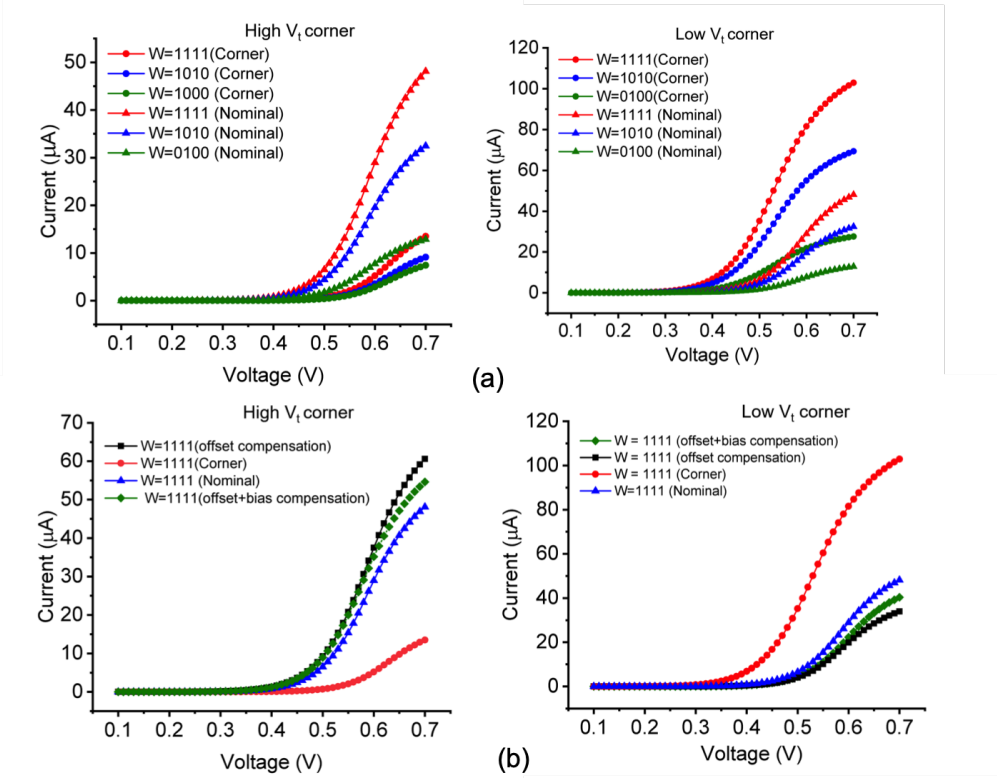


Figure 6.8. (a) Shows the effect of global change in VT on the output current for +/- 90mV change in the nominal VT. (b) Shows that by adjusting the Vpos (Offset compensation) or Vpos in addition to Vbias (Offset + bias compensation), the resultant currents in presence of global variations can be easily compensated for.

digital storage and can also be configured as a compute engine as and when required. It is also worth mentioning that the 8T cell has also been demonstrated in [186] as a primitive for vector Boolean operations. This work, significantly augments the possible use cases for the 8T cells by adding analog-like dot product acceleration.

Due to different sizing of the read transistors and an additional metal line routing for SL, there is an area penalty of using the proposed configurations, compared to the standard 8T-SRAM bit-cell used for storage. Fig. 6.6 shows the thin-cell layout for a standard 8T-SRAM bit-cell [196]. Note that the rightmost diffusion with width (W) constitute the read transistors ($M1$ and $M2$). To implement the 4-bit precision dot-product, we size the width of read transistors in the ratio $8:4:2:1$, as described earlier. Thus, the width of the rightmost diffusion is increased to 8W, 4W, and 2W, increasing the bitcell length (horizontal dimension)

by ∼39.6%, 17.1%, and 5.7% for bits 3, 2 and 1, respectively, compared to the standard minimum sized 8T bit-cell with diffusion width W. Moreover, to incorporate an extra metal line (SL), that runs parallel to V_{DD} and ground lines, the cell width (vertical dimension) increases by ∼12.5%. The resulting layout of first four columns for two consecutive rows in the proposed array is shown in Fig. 6.7. The overall area overhead for the whole SRAM array with 4-bit weight precision, amounts to ∼29.4% compared to the standard 8T SRAM array. Note, this low area overhead results from the fact that both the read transistors $M1$ and $M2$ share a common diffusion layer and hence an increase in transistor width can be easily accomplished by having a longer diffusion, without worrying about spacing between metal or poly layers. Additionally, instead of progressively sizing the read transistors one could also use multi-V_T design wherein the LSBs consist of hight V_T read transistors and the MSBs consist of nominal (or low V_T read transistors). The use of multi-V_T design can significantly reduce the reported area overhead. As such, the reported area overhead is close to the worst case impact on the bit-cell area without resorting to additional circuit tricks like multi-V_T design.

6.4 Variation Analysis

To ascertain the robustness of the presented dot product computations, in this section, we analyze the effects of non-idealities on the output current. The non-idealities considered are SL and BL line-resistances and transistor threshold voltage variations.

6.4.1 Corner Analysis

Since the currents through the read transistors depend on the effective conductance of the read port, one would expect that the output current on the BL would be a strong function of global variations in transistor threshold (VT) due to process corners or temperature changes. We reproduced (Fig. 6.8(a)) the leftmost figure of Fig. 6.4(b) assuming global changes in the threshold voltage of +/- 90mV% as compared to the nominal threshold voltage. As can be seen from the figure, the output currents indeed are a strong function of global variations in VT. However, interestingly, such global effects can be easily taken care of by

compensating/calibrating circuits. An initial calibration based on on-chip tracking of process corner or temperature can be used to either calibrate the input voltage ranges, or the offset voltage (Vpos) applied to the OPAMP. As an illustration, in Fig. 6.8(b), we show that by adding or subtracting voltages from Vpos and Vbias one can easily compensate the deviation in current due to global changes in VT. Note, the plots for Fig. 6.8(b) were generated in a similar manner to Fig. 6.4(b) except that Vpos and Vbias were added or subtracted with a constant compensating voltage. One can also compensate digitally, by adding or subtracting a constant number (a bias) from the resultant dot product (after converting it to a digital value). It is worth mentioning, that since process corners and temperatures have global effects, a single bias estimator circuit can be shared among multiple sub-banks of the SRAM array, amortizing its energy and area overhead. For the rest of the analysis, we have assumed the nominal corner case.

6.4.2 Effect of Line-Resistances

Both the SL and BL line-resistances add parasitic voltage drops along the rows and the columns. Moreover, to complicate the analysis, the error in the output current would be a function of both the spatial dependence due to distributed line-resistances and data-dependence as a function of the stored weights in the memory array. We, therefore, resort to worst case analysis. The worst case arises when all the weights and all the inputs are at the highest value. This scenario results in maximum current flow through the BLs and SLs and hence has maximum impact of parasitic line-resistances. To analyze the impact, we consider a line resistance of 1.3 ohms/μm [202]. Based on the layout, the average line resistance between each bit-cell was found to be 1.25 ohms in the bit-line direction and 2.5 ohms in the SL direction. We explore both the configurations (Config. A and Config. B) to analyze the impact of the line-resistances and ways to compensate for the voltage degradation along the metal lines. In addition, for Config. B, we explore two variants to minimize the effect of line resistances. Note, in Config. B the inputs are connected to the word-lines i.e. to the gate of the transistors. As such, the inputs drive capacitive load and there is no voltage degradation due to line-resistances. On the other hand, the bias voltage is connected to the SL, which

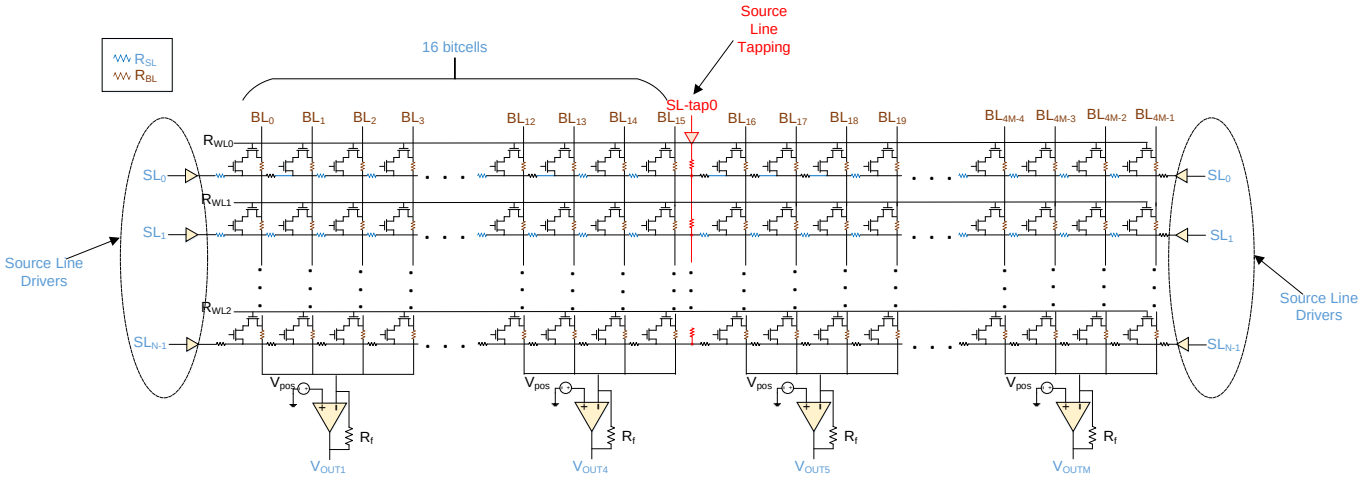


Figure 6.9. Bitcell organization of Config. B variants showing SL driven from both ends and tapping of SL every 16 bitcells. The line resistances in the source line (SL) and the bit-line (BL) are shown.

would degrade due to line-resistances and induce error in the final output current flowing through each column. To minimize this error, the two variants of Config. B presented in the chapter are:

- Config. B with the bias voltage driving the SL from both the ends (i.e. from the extreme right and extreme left ends, as shown in Fig. 6.9).
- Config. B with the SL tapped every 16 bits with regenerated values of the bias voltage in the horizontal direction, as depicted in Fig. 6.9.

Fig. 6.10 shows the worst case impact (when all inputs are at the highest value and all the weights are '1111') of the line-resistances in terms of percentage error in the output current (Note, this is error with respect to the current values, it should not be confused with the error corresponding to the classification accuracy) for the various configurations for simultaneous activation of 16 and 8 rows, respectively. As observed, Config. A has a higher error than all the variants of Config. B. Note, tapping is infeasible in case of Config. A because in Config A, the input voltages are connected to the SL. Tapping in Config. A would therefore require regeneration of input voltage along the horizontal direction, making it infeasible. In contrast, in case of Config. B, SL is supplied by a global bias voltage and

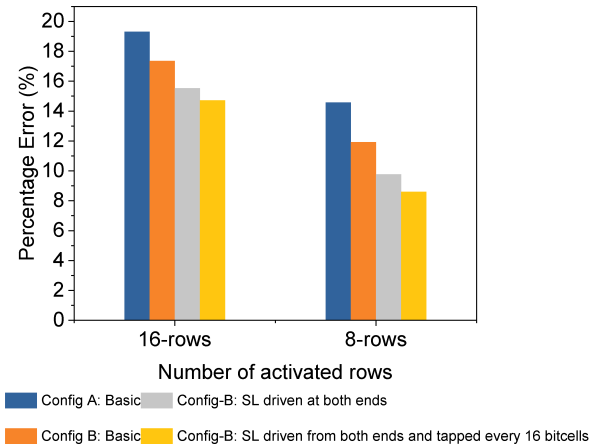


Figure 6.10. Percentage error in output current for worst case combination (highest input values and all weights = '1111'). The left set of bar graphs represent the error for various combinations assuming 16 rows are activated simultaneous for the dot product computation, while the right set of bar graphs correspond to simultaneous activation of 8 rows.

hence, is easy to regenerate. We have assumed an array size of 64x128 (64 rows and 128 columns). Further, for our analysis we assume that the 'farthest' 16 rows are simultaneously activated. SL and BL distributed resistances were included for all the activated rows, while the unactivated rows were modeled by an equivalent lumped resistance.

For rest of the analysis, we choose Config. B with tapping every 16 bits. We now analyze the error percentage across all voltages and weight combinations to understand the impact of the degradation in light of applications discussed in the chapter. Fig. 6.11 (a) and (b) shows the 2D error-map due to line resistances for various combinations of input voltages and weights for 16 and 8 activated rows, respectively. Note, for each input voltage − weight combination all rows are supplied with the same input voltage and all weights in the array are same. In addition, Fig. 6.11 (c) shows the weight level distribution of a neural network layer trained on the MNIST dataset. As observed from Fig. 6.11 (a) and (b), the error above 6% for 16 rows and above 4% for 8 rows is concentrated to the top 25% of the map corresponding to the highest weights and inputs. However, from Fig. 6.11 (c), we observe that for relevant applications such as neural network the trained weights are mostly concentrated to the weight level 1-6 where the error is close to 0-5% for 16 rows and 0-3.4%

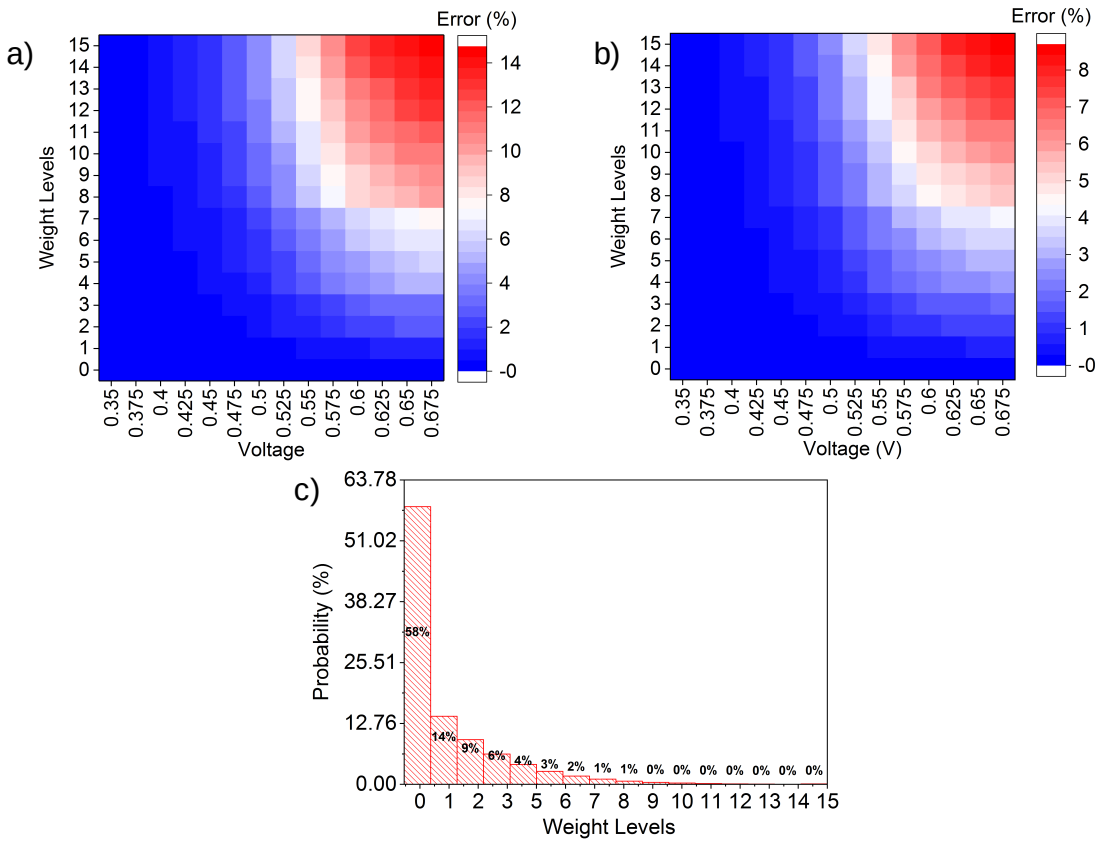


Figure 6.11. (a) and (b) shows the percentage error map arising due to line resistance for different weight levels ranging from '0000' to '1111' and input voltages ranging from 0.35V to 0.675V for 16 and 8 activated rows. For e.g., the data point corresponding to V = 0.35V and weight level = '0000' means the test case where all the 4-bit weight elements in the memory array are considered to be at weight '0000' and the input voltages to all rows are 0.35V. The percentage error decreases with decreasing weight and input value. (c) Probability of occurrence of weight levels in a trained neural network on MNIST dataset shows lowest weight levels have the highest frequency, thus indicating low impact due to line resistance.

for 8 rows. From this analysis, we can conclude that using the circuit techniques presented *i.e.* driving SL from both the sides and tapping every 16 columns and also leveraging the weight distribution for a trained neural network, the effect of line resistances for simultaneous activation of 8 and 16 rows can be substantially mitigated. For example, for Config. B with taps every 16 columns with SL being driven from both the ends, the worst-case error is

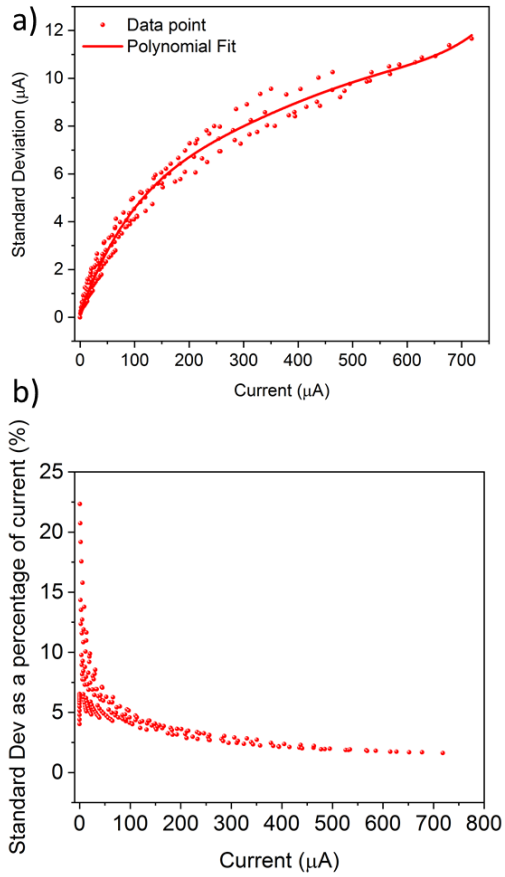


Figure 6.12. (a) Standard deviation of current due to variations in Vt of the transistors of the bitcells with increasing current for 1000 Monte Carlo simulations. A single data point shown here refers to the standard deviation in output current when 16 rows are activated and input voltages to all rows are V_{in} and weights of all elements are w. For different data points, we consider V_{in} values ranging from 0.35V to 0.675V in steps of 0.025V and weight levels ranging from 1 to 16 to capture the impact of V_T variations across the input parameter space. (b) Standard deviation as a percentage of the total current showing a decreasing trend with higher current.

contained within 9%. Further, it was observed that the error improves rapidly when the input voltages or the programmed weights are less than their maximum possible values.

6.4.3 V_T Variations

The variations in transistors can result in error in the dot-product operations. To analyze this, we perform 1000 Monte-Carlo simulations to assess the variation of the output current

for various combinations of input voltages and weights. We considered 30 mV σ variation of threshold voltage (V_T)for the minimum sized transistor and scaled the variation with width as $\sigma_L = \sigma_{min}\sqrt{W_{min}L_{min}/WL}$. Note, for random variations it is customary to include various sources of variations into effective variation in the transistor threshold voltage [203]. We ran 1000 Monte-Carlo simulations for each voltage value ranging from 0.35V to 0.675V in steps of 0.025V and each weight level ranging from 0 to 15 and obtained the standard deviation in output current for each case. This captures the impact of Vth variations for a considerable precision of gate voltages. We calculated the standard deviation about the mean current for the entire range of output current from the cases described above for 16 activated rows of the memory array. The minimum current on the x-axis in Fig. 6.12 (a) arises when the input voltages and (or) the stored weights in the memory array are zero. The next higher level of current is obtained when either the weight or the input voltage is incremented. It is worth noting that Fig. 6.12 (a) corresponds to 16 activated rows in an array of 64 rows and 128 columns. Further, for the analysis in Fig. 6.12 (a), we have neglected the effect of line-resistances for the following reasons - 1) adding line-resistances makes the standard deviation in Fig. 6.12 (a) a function of not only the random V_T variation and weights, but also makes the deviation in current spatially dependent. This leads to a non-trivial analysis problem that can quickly become intractable. 2) As shown in the previous sub-section, even the worst case error due to line-resistances was well within acceptable limits. Fig. 6.12 (a) shows that the absolute standard deviation is higher for a higher value of output current. We fit a representative standard deviation for each current value using a polynomial fit as shown by the fitted line in Fig. 6.12.

In the functional simulations to evaluate the classification accuracy with such V_T variations, we calculated the output current from every 16 rows and replaced it with a random value from a Gaussian distribution with the corresponding standard deviation of the particular output current. The final classification accuracy was 0.01% lower than the case without random variations. Table. 6.1 shows the inference accuracy on MNIST dataset for 8T dot product engine in comparison to ideal accuracy. This error resiliency is mainly due to the inherent robustness of neural networks. If we look at the standard deviation as a percentage of the total current in Fig. 6.12 (b), we observe that the errors are the highest for low currents.

Table 6.1. Comparison of 8T DPE inference accuracy on MNIST

Network Configuration	Accuracy (%)
Ideal (Software)	98.27
8T DPE (No VT variations)	98.16
8T DPE (With VT variations)	98.15

However, device to device variations can have both additive and subtractive effect on the currents through each cell. This means that for simultaneously activated rows, the resulting current deviation in individual synapses due to variations tend average-out as we are primarily interested in the sum of the currents. This results in a lower error due to variations. As such for generic dot-product operations, there exists a trade-off, lower voltage implies lower current (lesser than the electromigration limit) allowing to turn ON more number of rows simultaneously and hence leads to higher parallelism. At the same time, lower voltage also results in higher variation and hence more approximation in the dot product calculations. For neural networks / machine learning applications and specifically for the case of on-line learning such approximations can be taken care of by relying on the inherent error resiliency of the target applications.

6.5 Discussions

We would like to emphasize the fact that the present proposal aims at providing a means to enable *in-situ* dot-product computations in standard 8T SRAM cells by exploiting the isolated read-port. We believe a wide-range of applications can be accelerated using the present proposal. As such, the presented dot product engine should not be seen only in context of machine learning and neural network applications. In general, any application that can benefit from approximate vector addition and multiplication can be a possible use case for the presented proposal. This wide spectrum of possible use cases implies that the exact details of the required peripheral circuits and its complexity would depend heavily on the target application. For example, error resilient applications like neural networks can rely on low cost peripherals whereas more traditional dot-product computations as in image process-

ing might require sophisticated circuitry. Moreover, one could think of hybrid significance driven peripheral design such that the less significant computations are associated with low overhead peripherals while more significant operations are enabled by high accuracy circuits or a full digital computation without resorting to dot product acceleration. The target application would also dictate the constrains on OPAMP specification and the required precision of the resistance Rf shown in Fig. 8. In addition, the choice of Config. A versus Config. B would also depend on the target application. For example, Config. A shows better linearity as opposed to Config. B. However, the input voltages in Config. A drive a resistive load requiring complex driving circuits as opposed to Config. B which has capacitive load. The authors would also like to point that a detailed analysis of the appropriate peripherals and the associated architecture for each individual use case requires a case by case analysis and is not the focus of the present chapter. The current chapter is more of a generic proposal and a study of the effect of intrinsic non-idealities, for example, the non-linearity, the line-resistances and the transistor threshold voltage variations with respect to the present dot product engine.

Advantageously, the presented proposal resembles dot product computations in memristive crossbar arrays [122]. Memristive dot products have been extensively studied and techniques like segregation of positive and negative weights in different sub-arrays, bit-slicing, efficient sensing schemes for computations have been widely proposed [16], [204], [205]. Such memristive techniques can also be used in conjunction with the presented proposal. We would, however, like to highlight the fact that the present proposal should not be considered as a replacement for memristive crossbar arrays. Rather, we feel that memristors are more suitable for spatial architectures [16], [205], whereas the presented proposal can be used to augment on-chip storage (cache or GPU register files) with dot product acceleration.

We would now present the estimates for energy consumptions by performing 16x16 dot product operation with and without the proposed dot product engine. It is worth mentioning that the overall dot-product engine consists of DACs to generate analog inputs fed to the 8T-SRAM crossbar array, along with ADCs to detect the analog outputs and converting them back to digital bits. A cache memory of size 256KB with a basic sub-array size of 64×128 bits was modeled using CACTI [206] simulator. The energy consumption and latency of

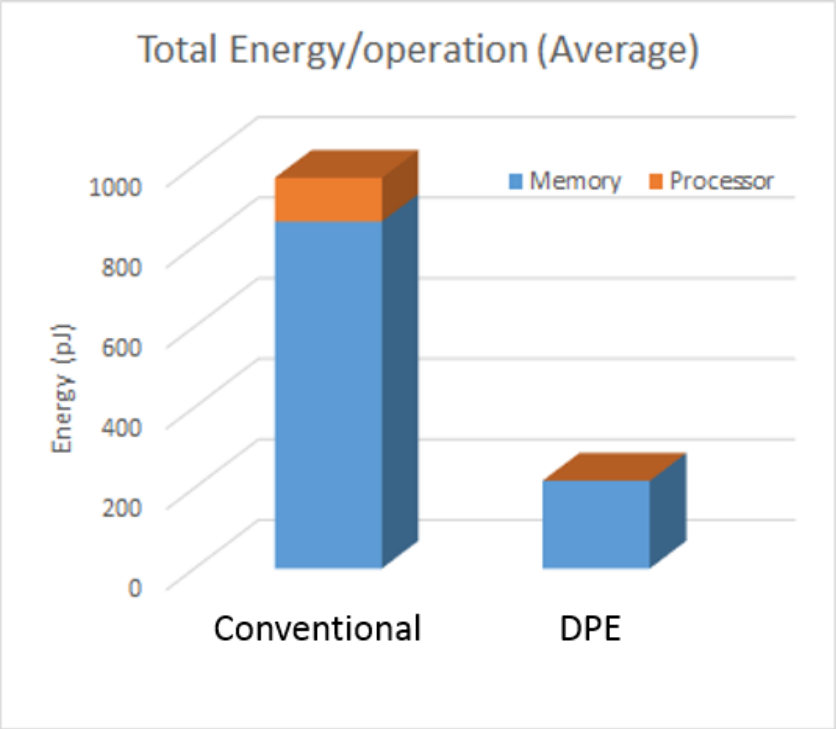


Figure 6.13. Average Energy comparison between conventional digitial sequential implementation and proposed Dot-product Engine (DPE). The energy is reported for 16x16 dot product computations wherein 16 rows are simultaneously activated and each row consists of 16 4-bit words.

the peripheral circuitry (ADCs and DACs) was appropriately incorporated in the CACTI model, referring to [207]. We assume a 16×16 crossbar operation (*i.e.* activating 16 rows at a time with each row containing 16 four bit words) at any given time, thus requiring 16 ADCs in the peripheral circuitry, per sub-array. The conversion time for the ADC operation was assumed to be 10ns and the energy estimates for the ADCs were adopted from [207]. This framework was used to evaluate the total energy consumption and latency of the proposal for a test vector of 16×16 dot-product, compared to the pure digital approach wherein the dot product was computed by sequential memory access and multiply as well as add operations were performed in dedicated adders and multipliers synthesized separately.

Fig. 6.13 shows the energy for performing a 16×16 dot-product with the proposed DPE and the conventional digital approach. This energy overhead stems from the fact that in digital approach, row-by-row access to the data from memory, followed by MAC operations

are performed sequentially to compute the same 16×16 matrix-vector dot-product, which the proposed DPE can do in a single instruction. Also, it was noted that the total energy consumption of the dot-product engine had a dominant contribution from the peripheral circuitry. Nevertheless, in general, the energy and latency overheads associated with respect to DACs and ADCs in similar dot product engines based on memristors have been extensively studied and can be found in works like [16], [205].

6.6 Conclusion

In the quest for novel in-memory techniques for beyond von-Neumann computing, we have presented the 8T-SRAM as a vector-matrix dot-product compute engine. Specifically, we have shown two different configurations with respect to 8T SRAM cell for enabling analog-like multi-bit dot product computations. We also highlight the trade-offs presented by each of the proposed configurations. The usual 8T SRAM bit cell circuit remains unaltered and as such the 8T cell can still be used for the normal digital memory read and write operations. The proposed scheme can either be used as a dedicated dot product compute engine or as an on-demand compute accelerator. The presented work augments the applicability of 8T cells as a compute accelerator in the view that dot products find wide applicability in multiple data intensive application and algorithms including efficient hardware implementations for machine learning and artificial intelligence.

7. SPARSITY AWARE COMPUTE-IN-MEMORY PROCESSOR BASED ON 8T SRAM

(Work done in collaboration with Mustafa Ali)

7.1 Introduction

The growing trends of developing domain-specific accelerators for Machine Learning applications has led to a deep exploration of compute-in-memory primitives based on SRAM arrays [11]–[14], [24], [193], [208], [209]. In the previous chapter (Chapter 6, we proposed a 8TSRAM-based compute-in-memory primitive capable of performing multi-bit dot-product. In Chapter 5, we learnt how weights in neural networks can be mapped on to these SRAM arrays and inputs are streamed to perform convolutional and linear operations through iterative matrix-vector multiplications. In this chapter, we will delve into detailed nuances of compute-in-memory with the focus on how to improve the efficiency of such systems through circuit design.

Designing efficient CIM primitives and micro-architectures necessitates examining the energy breakdown of different components that constitute the compute system. Fig. 7.1 shows the energy breakdown for a standard CIM compute block along with its peripherals. We observe that the ADC consumes 57% of energy and 81% of area in the compute block. Hence, CIM benefits can be shadowed by the significant energy, latency, and area cost of the ADC [15]. There is a need for cross-layer efforts to amortize the significant ADC energy

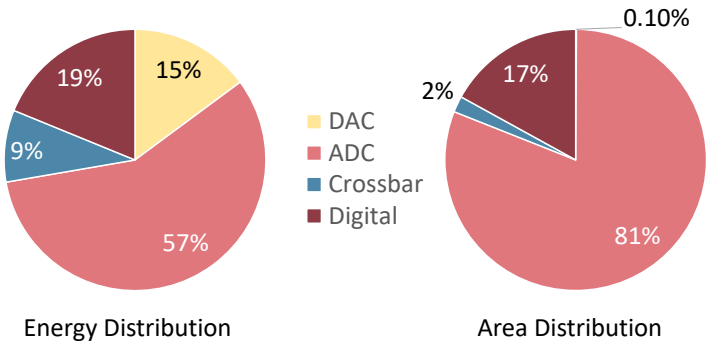


Figure 7.1. Energy and area distribution of ReRam-based MVMU

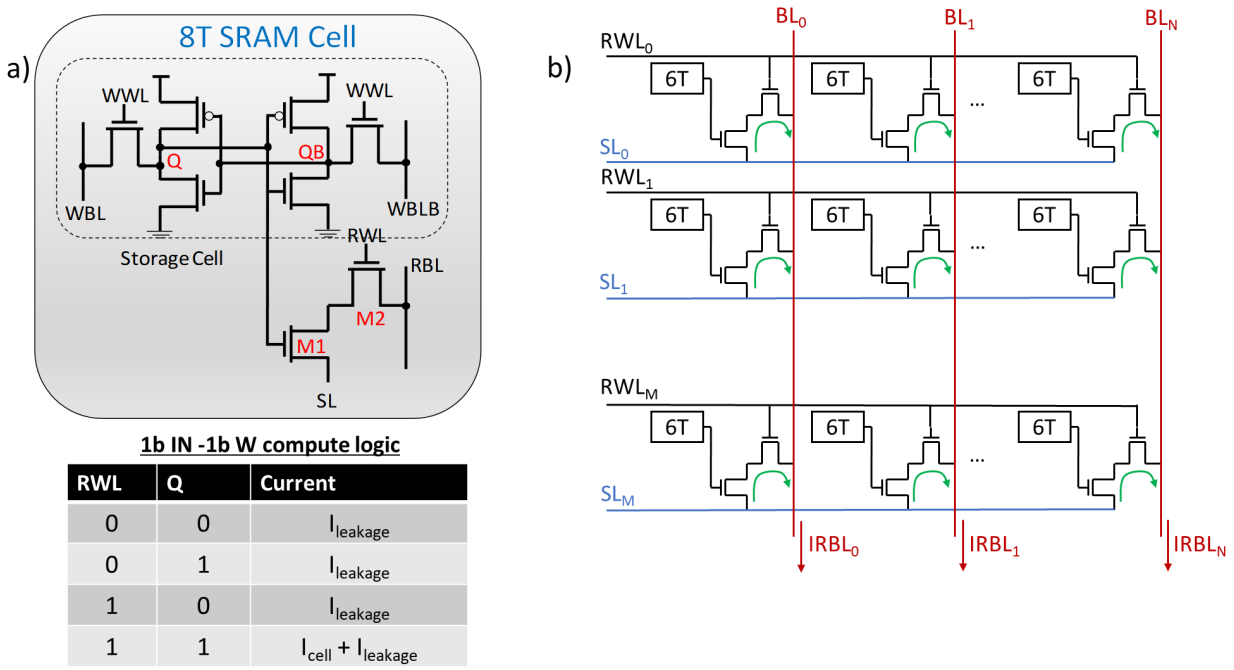
and performance cost. One such approach is geared towards leveraging sparsity in weights and activations of ML workloads. Due to the bit-serial nature of computing in CIM, it is exposed to bit-level sparsity which can be heterogenous across different layers in a workload or among various workloads. Bit-level sparsity can be exploited to reduce the precision of ADC, thereby lowering the energy and latency of CIM operations.

7.2 8T SRAM based Compute-in-Memory Cell - A Brief Recap

In Chapter 6, we introduced the concept of a 8T SRAM based compute-in-memory array capable of performing multi-bit dot-products by simultaneous application of voltages on the read word-lines (RWL). Fig. 7.2 (a) shows a 8T SRAM bit-cell. The two transistors in the read port, M1 and M2 can be re-purpose to perform multiplication in the bit-cell itself. Fig. 7.2 (b) shows a $M \times N$ 8T SRAM compute-in-memory array capable of performing matrix vector multiplication between input vector applied through activation of multiple RWLs, and the weight matrix represented by the data stored in the SRAM array. By applying a voltage on the source line (SL), the current flows from the SL based on the input at the RWL and the content of the corresponding 6T cell. The logical representation representation of a 1-bit input (IN) and 1-bit weight (W) computation is shown in the truth table in Fig. 7.2 (a). The current through each cell gets added in the bit-lines (BL) to produce $IRBL_{\mathrm{j}}$, which represents the dot-product of the inputs and weights.

7.3 Analysis of Sparsity in ML Workloads

Machine Learning workloads can be quite sparse, and several works have explored techniques [210], [211] to leverage that sparsity in order to achieve benefits in performance and energy consumption. In regard to compute-in-memory which deploys bit-serial computing, the compute primitive is exposed to bit-level sparsity which can be even higher than data level sparsity [212] in ML workloads. To motivate a sparsity-aware CIM design, we perform preliminary analysis on bit-level sparsity in a standard ML workload, i.e., ResNet-20 on CIFAR-10 dataset. Fig. 7.3b shows bit-level sparsity of input activations (considering 64x64 sized CIM operation) across first 15 convolutional layers in ResNet-20 tested on CIFAR-10



1b IN -1b W compute logic

RWL	Q	Current
0	0	$I_{leakage}$
0	1	$I_{leakage}$
1	0	$I_{leakage}$
1	1	$I_{cell} + I_{leakage}$

Figure 7.2. 8T SRAM array deploying current-domain compute to perform MAC operations, with the table showing compute logic.

dataset. We observe that activations of each layer experience different bit-level sparsity and the distribution of various sparsity levels can vary remarkably across layers. Correspondingly, the analysis shows that different ADC precisions (2-6 bit) are required to optimally leverage the varying degrees of sparsity within and across layers.

Therefore, leveraging sparsity in ML hardware has become a key solution to improve energy-efficiency. Few works have leveraged sparsity in CIM-based systems [14], [213]. In [214], offline workload profiling is required to determine the low-MAC output threshold, which is used to decide if the ADC conversion is performed or skipped to save macro energy and latency. In [213], workload training is required to induce block-level sparsity that can be leveraged using *row-gating* techniques. However, row gating does not harness the maximum potential energy and performance benefits that bit-level sparsity offers. The reason is two-fold. First, row gating reduces width of the dot-product by activating less number of rows, which can lead to lower energy efficiency considering sub-quadratic decrease in ADC energy/latency with precision [215]. Second, row gating leverages block-level sparsity (the

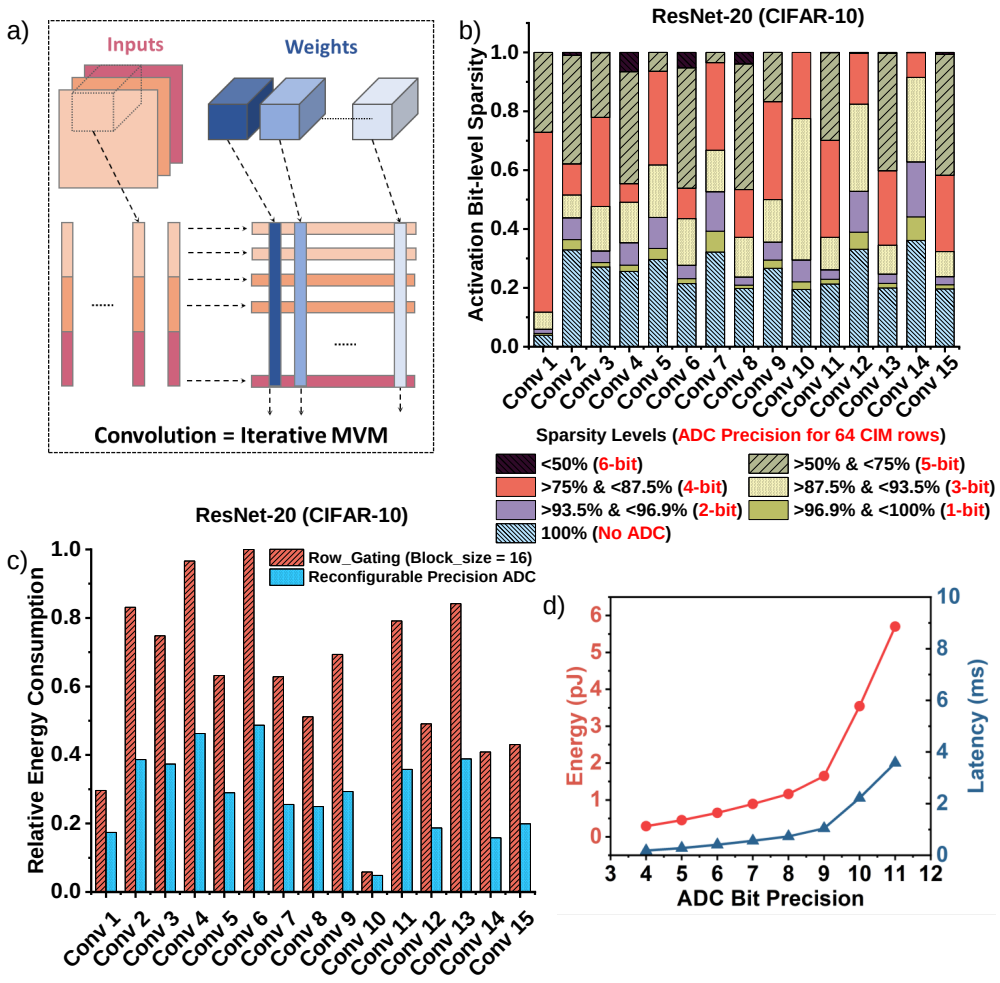


Figure 7.3. (a) CONV operation on CIM using iterative MVMs. (b) Bit-level sparsity of a ResNet-20 model running CIFAR-10 with the required ADC precision. (c) Comparison between row gating and reconfigurable-precision ADC in terms of energy. (d) Energy/latency scaling with ADC precision.

block size can be 8 input bits as in [213]) rather than bit-level sparsity which leads to coarser granularity of sparsity control. Additionally, authors in [216], [217] proposed sparsity-based input-aware sensing schemes. Our approach presents a modular and high throughput sensing than previously reported input-aware sensing schemes. Specifically, our approach enables conversions of voltages on *two* bit-lines simultaneously for ADC precisions 2, 3, or 5-bit, providing 2× throughput for those precisions, owing to the proposed reconfigurable precision ADC. This is in contrast to approaches presented in [216], [217] which depend on changing

the reference voltages or reducing conversion cycles based on input sparsity which provides lower performance than our method.

In this chapter, we propose an alternative CIM design with a novel reconfigurable-precision ADC which can leverage bit-level sparsity at the finest granularity for maximum energy and performance gains. In the proposed design, we adaptively change the precision of the ADC according to the bit-level sparsity of the input activations. Reconfigurable-precision ADC introduces an inherent advantage over row gating in CIM because it leverages sparsity at a finer granularity (bit-level). As shown in Fig. 7.3c, the reconfigurable-ADC approach provides better energy efficiency in most layers of ResNet-20 on CIFAR-10 compared to row gating adopted in [213]. Fig. 7.3d shows the energy/latency trends of SAR ADCs at various precisions [215]. To summarize, the presented CIM macro features the following contributions:

- Reconfigurable-precision ADC running at lower precision with higher sparsity and vice versa.
- Sparsity-aware TIA which increases the sense margin with lower precisions.
- Run-time irregular sparsity-aware computing with no offline profiling/specific workload training.

7.4 Sparsity-Aware Compute-in-memory Macro with Reconfigurable Precision ADC

7.4.1 Macro Structure

Fig. 7.4 illustrates the overall CIM macro structure. The memory array is comprised of 8T-SRAM cells which are repurposed to perform the multiplication operation in the analog domain [24], as shown in Fig. 7.2 (b). The 8T-SRAM array stores weight data (W_{ij}), while input activations (IN_i) are fed to the input-aware word-line (WL) decoder which activates only those read word-lines (RWLs) whose IN_i=1. The source-lines (SLs) are driven to Vbias through the SL driver. Based on W_{ij} and IN_i at the RWL of an 8T-SRAM cell, current is driven through the read-port transistors from the SL to the read bit-line (RBL). Currents

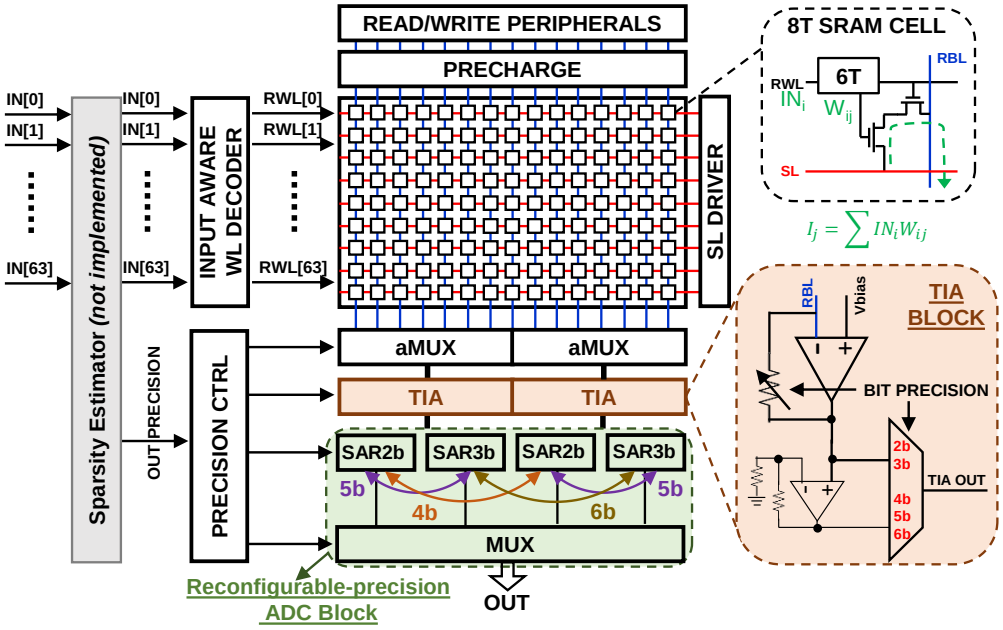


Figure 7.4. The proposed macro structure and timing diagram.

from simultaneously activated rows get accumulated in the columns (RBL) representing the dot-product ($\sum IN_{\mathrm{i}} * W_{\mathrm{ij}}$). The RBLs are fed to analog multiplexers (aMUX) to select which column to read, followed by two TIA blocks for converting current to voltage. Each TIA consists of two amplifiers as shown in Fig. 7.4 to achieve higher range and configurability. Based on the sparsity of IN_{i} data, the bit-precision of the output (2b-6b) is chosen. The TIA outputs are fed to the ADC block to be converted to digital. The proposed ADC block provides reconfigurable precision that gets selected based on input sparsity at the run time. It consists of two 2-bit and two 3-bit SAR ADCs, which can be operated in isolation to provide support for two 2-bit ADCs and two 3-bit ADCs, as well as combined to generate one 4-bit ADC, two 5-bit ADCs and one 6-bit ADC. Thus, for 2b, 3b and 5b precisions, two columns can be processed in parallel, thereby giving a higher throughput, while for 4b and 6b precisions, the processing is done one column at a time. It is worth mentioning that the required output precision can be determined by calculating input sparsity at the run time using a sparsity calculator which is simply a 64-bit counter (not implemented in the prototype chip). The sparsity calculator counts the number of ones in the input bit-stream and decides the required precision based on input sparsity. For instance, if the input stream

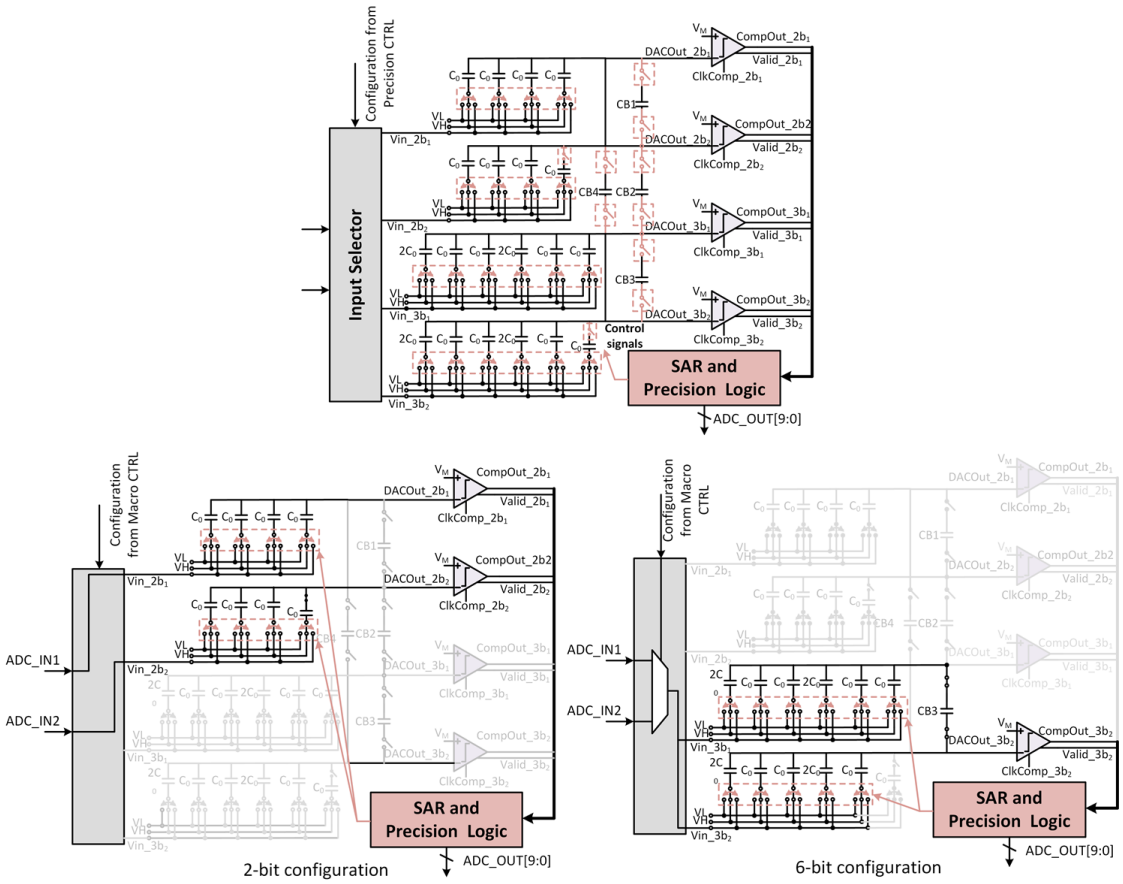


Figure 7.5. The proposed reconfigurable-precision SAR ADC with two example configurations: 2-bit and 6-bit precision.

has 50% sparsity (half the input bits are ones), then the precision will be 5-bit instead of 6-bit. Note, the cost of such precision calculator can be amortized across many macros receiving the same input activations.

7.4.2 Reconfigurable-Precision SAR ADC

Fig. 7.5 shows the proposed reconfigurable-precision SAR ADC which supports 2-bit to 6-bit precision configurations based on input sparsity. Typically, the proposed ADC consists of two 2-bit and two 3-bit SAR ADCs with their capacitive DACs connected through switches and bridge capacitors. The main sub-circuits in the ADC are: i) Input selector, ii) DAC capacitors, iii) Comparators, and iv) SAR and precision logic. The input selector

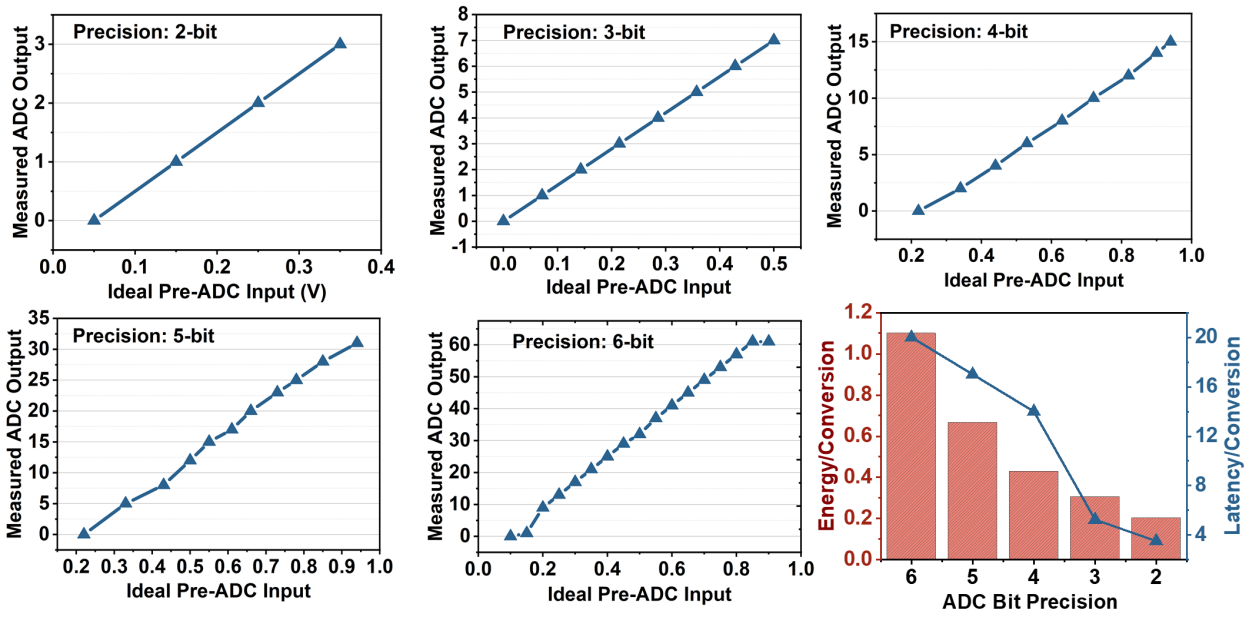


Figure 7.6. Measured ADC results showing output vs input voltage, energy and latency for various precisions.

and reconfigurable SAR and precision logic are designed to support the precision reconfigurability. The input selector comprises analog switches to decide which input is applied to the capacitive DACs based on the selected precision. Moreover, the bridge capacitors (CB1, CB2, CB3 and CB4 in Fig. 7.5) decide the equivalent DACs formed from the available 2-bit and 3-bit DACs based on the selected precision. For instance, by turning on the switches surrounding CB3, 6-bit DAC is formed from the two 3-bit DACs and 6-bit conversion can be performed. The SAR and precision logic block is responsible for the signals controlling all DACs switches and comparators along with producing the final ADC outputs. The proposed ADC features four comparators with each comparator input connected to one of the four DACs and produces the comparison output and a valid signal to indicate that the comparison is finished. Additionally, the comparators' clocks are generated in the asynchronous SAR logic block using valid signals. The DAC capacitors are arranged in the split-capacitor fashion for energy efficient capacitor switching. Note, in our design we can perform two 2-bit, 3-bit or 5-bit conversions simultaneously since we have two modular 2-bit and 3-bit DACs. However, we can only perform one 4-bit or 6-bit conversion at a time. Regarding

4-bit precision, the two 2-bit DACs are connected through CB1 to form one 4-bit DAC. Such design choice is decided due to area constraints.

Further, the TIA block is reconfigured based on the chosen precision: (1) The variable feedback resistor is set based on precision to provide the desired gain and output range. (2) At higher precision (4-6 bit), a second stage is also used for additional gain (by trading off power and latency), i.e., TIA_OUT is obtained from TIA2 for 4-6 bit and TIA1 for 2-3 bit, as shown in Fig. 7.4. Therefore, the proposed TIA allows higher signal margin at lower precision due to the fixed range.

7.4.3 Measurement Results

Fig. 7.6 shows the measured ADC outputs for a full range of input voltages with 2-6 bit precisions of the proposed reconfigurable ADC. Additionally, Fig. 7.6 presents the energy and latency per conversion of the proposed ADC. ADC energy varies from 0.2 pJ to 1.1 pJ for 2-bit to 6-bit configuration, respectively, whereas ADC latency/conversion varies from 3.5ns to 20ns. Note, 4-bit conversion latency is higher than 5-bit because 5-bit precision configuration has two conversion channels whereas the 4-b has only one conversion channel as described in the previous section due to area constraints.

Fig. 7.7 shows measurement results from the proposed CIM macro. We performed experiments with different input and weight combinations that produce the same CIM output. We observe that the ideal output and measured output show good agreement for different input sparsity (thus requiring different ADC precision). In Fig. 7.7, we show example ADC precision configurations of 3-bit and 5-bit to demonstrate the functionality of the CIM macro. We estimate the energy consumption of a ResNet-20 architecture on CIFAR-10 dataset. The energy presented only represents the measured energy consumed by the CIM macro and is simulated by considering a mapping scheme of the workload on 64x64 arrays for bit-serial compute **geniex**. Our design consumes 2.2x and 1.5x lower energy than a corresponding macro with fixed, homogeneous 6-bit and 5-bit ADC respectively. The proposed CIM macro achieves 1b/1b (Input/Weight) energy efficiency 35.5 to 127.2 TOPs/W for 6-bit to 2-bit

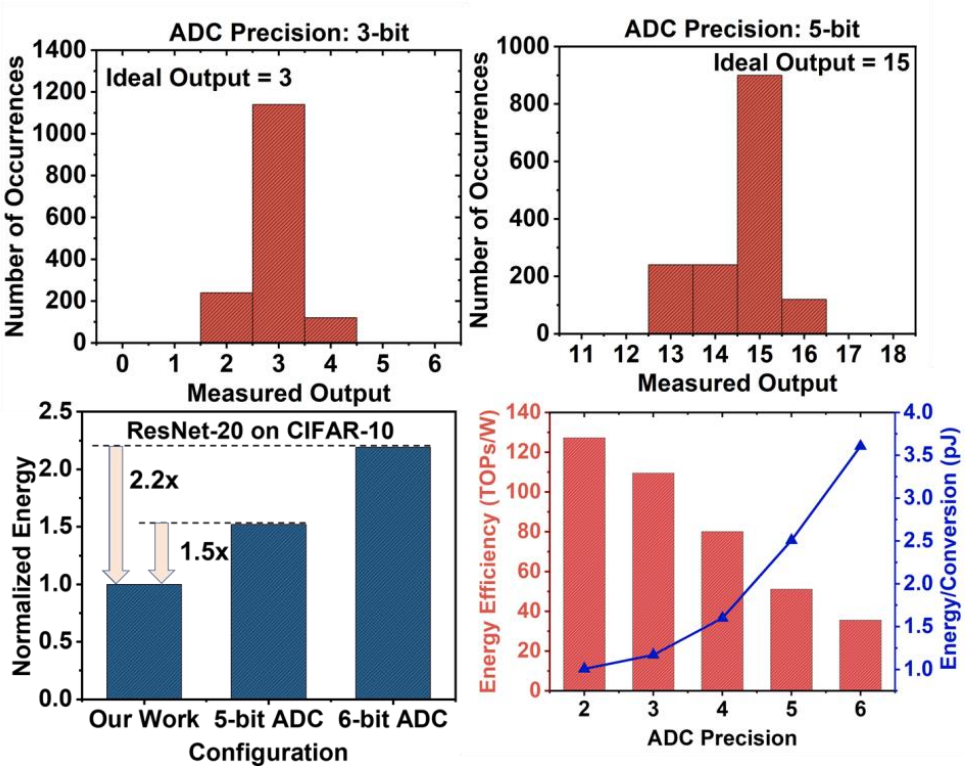


Figure 7.7. Measured CIM macro results show good agreement between expected output and measured output, energy efficiency for different precision and baseline comparison on workload.

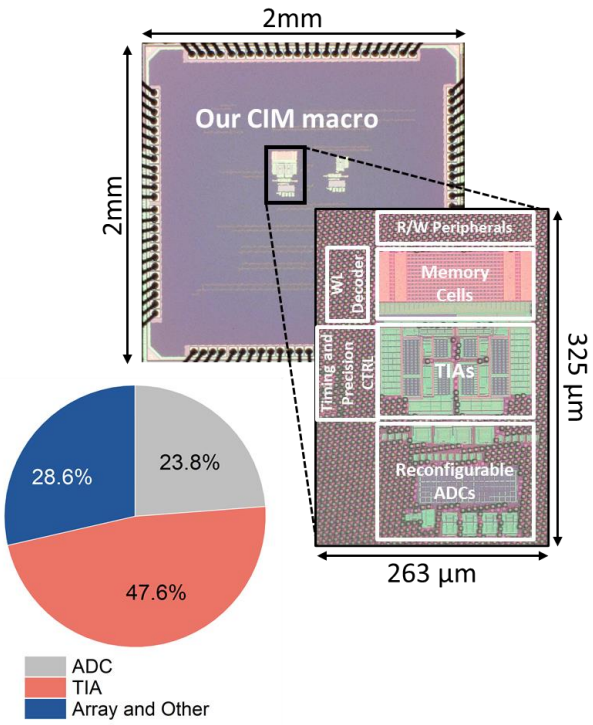
Table 7.1. Comparison with state of the art.

	TWIN-8T**si_isscc19**	CONV-RAM**conv-ram**	JSSC'19[12]	ISSCC'20[213]	ISSCC'21**su_isscc21**	ISSCC'20[13]	This Work				
Tech Node	55nm	65nm	65nm	28nm	28nm	7nm	**65nm**				
Memory capacity	3.75Kb	16Kb	2.4Mb	64Kb	384Kb	4Kb	**4Kb**				
SRAM Cell	Twin8T	10T	8T+1C	6T+Local Computing	6T+Local Computing	8T	**8T**				
MAC operation	Word-wise	Word-wise	Binary	Bit-wise	Bit-wise	Word-wise	**Bit-wise**				
Simultaneous MACs in analog domain	9*(2b*2b)	64*(1b*6b)	4608*(1b*1b)	16*(1b*2b)	16*(1b*2b)	64*(4b*4b)	**64*(1b*1b)**				
ADC precision	5b	7b	1b	5b	5b	4b	**2b**	**3b**	**4b**	**5b**	**6b**
TOPS/W	72.03	40.3	866	23.26	60.28-94.31	321	**127**	**109**	**80**	**51**	**35**
CIFAR-10	85.56%	-	84.37	91.90%	-	88.52%	**92.02%**				
CIFAR-100	-	-	-	67.60%	-	-	**68.66%**				
Sparsity Aware Computing	No	No	No	Yes[1]	Yes[1]	No	**Yes**				

[1] Requires offline workload profiling to determine the low MAC output threshold.

ADC precision, respectively. Additionally, the macro energy breakdown is shown in Fig. 7.7.

The die micrograph and chip summary are shown in Fig. 7.8. We use a design framework **geniex** to map DNN models on to the proposed CIM macro considering bit-serial compute. We measured the standard deviation for a range of expected CIM macro outputs for various configurations of the reconfigurable-precision ADC based on the sparsity of the input data, as shown in Fig. 7.9. Under such computational errors (arising due to deviations



Technology	65 nm
Macro array size	(64x64) 4 kb
Macro area	325x263 µm^2
SRAM cell type	8T
Supply voltage	1.1 V
Frequency	100 MHz
Energy-efficiency (TOPS/W)	35.5(6b-precision) 127.2(2b-precision)
Access time	20 ns (2,3-bit precision) 30 ns (4,5,6-b precision)
Accuracy (CIFAR 10)	91.49%
Accuracy (CIFAR 100)	65.35%

Figure 7.8. Die micrograph and chip summary with the macro energy breakdown

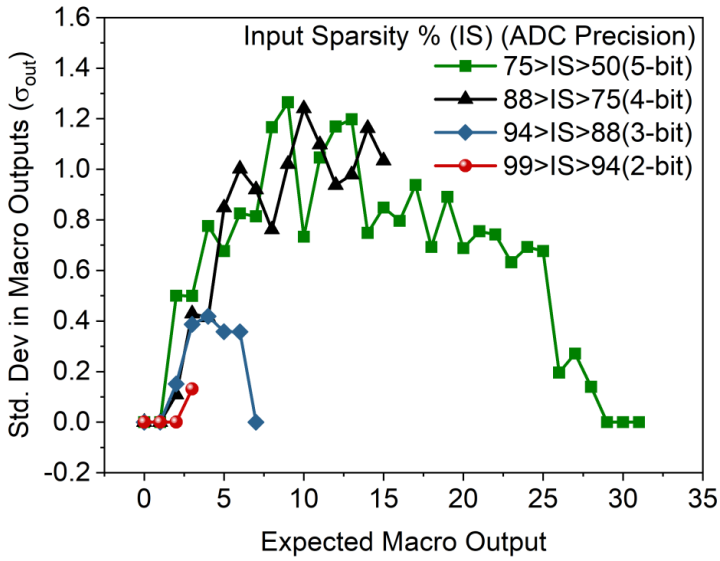


Figure 7.9. Standard Deviation in measured outputs from proposed CIM Macro.

from expected output), considering the DNN model ResNet-20, our proposed CIM macro with sparsity-aware reconfigurable precision ADC achieves accuracies of 91.49% on CIFAR-10 (software accuracy - 92.69%) and 65.35% on CIFAR-100 (software accuracy - 69.6%). Next, we compared our CIM macro with an implementation with 4b fixed-precision ADC,

Configuration	4b Fixed Precision ADC	**Reconfigurable Precision ADC**
Accuracy (CIFAR 10)	87.72%	**91.49%**
Accuracy (CIFAR 100)	61.94%	**65.35%**

Figure 7.10. Accuracy of fixed precision ADC v/s reconfigurable precision ADC.

i.e., without the support for sparsity-aware reconfigurability. In comparison to accuracies obtained considering fixed-precision ADC (say 4b), as shown in Fig. 7.10, the proposed CIM Macro with sparsity-aware reconfigurable precision ADC achieves 4-5% higher accuracy. Table 7.1 presents the comparison between our macro and related works. Our work leverages run-time irregular bit-level sparsity without the need to profile nor train the workload for maximum performance and energy benefits. When adopted in large scale systems, we believe the proposed CIM macro is capable of achieving an optimal trade-off between energy efficiency, performance and accuracy.

7.4.4 Conclusion

Bit-level sparsity is an interesting yet challenging workload feature with a potential for energy and performance benefits to CIM-based ML acceleration. We presented a CIM SRAM macro leveraging bit-level run-time input sparsity for energy and performance benefits. Typically, the proposed macro features a new sparsity-aware reconfigurable-precision ADC so that the precision can be reduced at higher sparsity. Moreover, a reconfigurable TIA is proposed to maintain the voltage range for different MAC output precisions. The proposed sparsity-aware CIM macro shows energy and performance benefits over baseline while maintaining accuracy.

7.5 Sparsity-Aware Compute-in-memory Processing Core

One of the main challenges in computing using CIM macros is the difficulty to scale up the design to a multi-macro processing core, and even to multi-core chips. Primarily, it is the nature of compute which is subject to errors at even the macro level which can magnify at the system level due to variations. Second, scaling up is also a challenge in case

of sparsity-aware CIM macros, since the granularity of sparsity needs to be explored across multiple macros in a multi-macro system. Finally, architecting the mapping and dataflow for optimal data re-use is necessary.

7.5.1 Related Work

There have been various demonstrations of CIM processors for inference acceleration based on both CMOS and RRAM technologies [218], [219]. More recently, Yue et al [213], [220] proposed CIM processors leveraging block-wise sparsity. However, most works on CIM processors which leverage sparsity do not provide support for balancing latency imbalance across various macros.

Here, we propose a hierarchical CIM processing core microarchitecture consisting of 32 sparsity-aware CIM macros, proposed in Section 7.4. The CIM processing core utilizes three key features:

- Sparsity Aware CIM Compute Unit (SCU) consisting of eight CIM Macros capable of performing matrix-vector multiplication operations between 8-bit or 4-bit weight matrices and 1-8-bit input vectors by leveraging sparsity to achieve lower energy consumption and higher throughput.
- Row-gating to achieve higher signal-to-noise ratio (SNR) necessary to maintain application accuracy
- Hardware support for input and output vector re-arrangement instructions to balance latency loads across multiple SCUs with different latencies.

7.5.2 CIM Core Features

The proposed sparsity-aware CIM processing core has the following features, necessary to address issues arising from micro-architectural extension of a sparsity-aware CIM macro: i) SNR-aware row-gating, ii) latency balancing and iii) run-time sparsity controller.

- **SNR-aware row-gating:** Analog CIM primitives are inherently erroneous and have low signal-to-noise ratio (SNR). DNN models are typically error resilient and we have

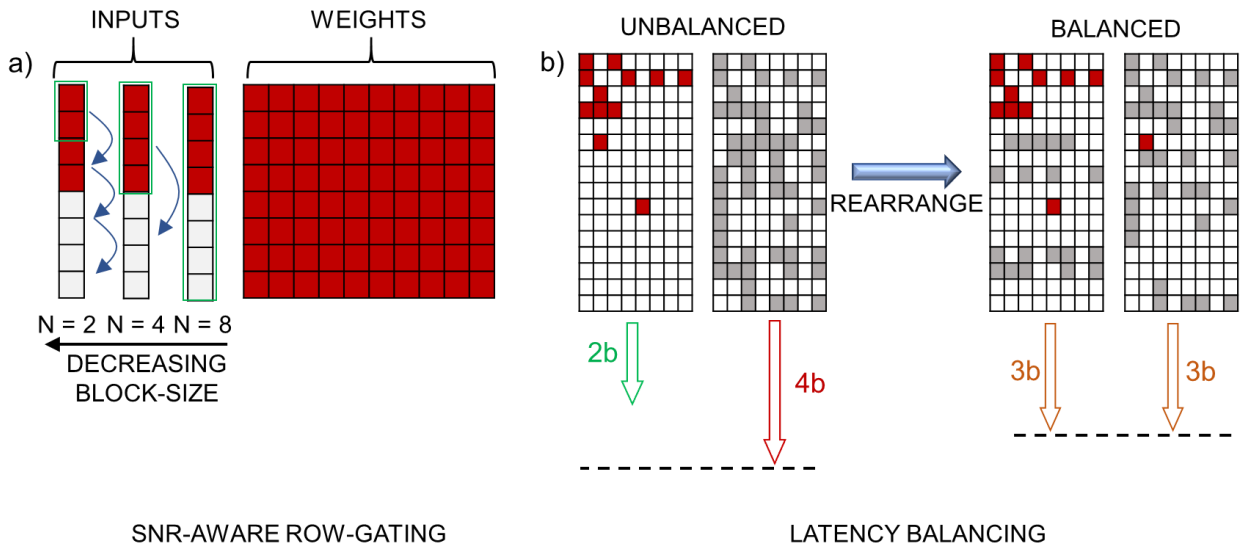


Figure 7.11. CIM Core Features: a) SNR-aware Row Gating and b) Latency Balancing

analyzed the accuracy degradation for these primitives in Sec. 7.4. The computational errors in analog CIM primitives is directly proportional to the number of simultaneous word-line (row) activations. Activating more rows simultaneously implies distinguishing more output states within a given output voltage/current range, leading to reduced sense margin. However, activating more rows can reduce peripheral overhead, leading to higher energy efficiency. In order to maintain optimal trade-offs between energy-efficiency and accuracy, we support SNR-aware row-gating. Fig. 7.11 a) shows a weight matrix and input vector. We can process the input vector by computing it at once ($N = 8$) or in steps of $N = 2, 4$. This technique is called row-gating. The CIM processing core supports various number of row activations such that we can adapt the row activations according to the SNR required by the workload.

- **Latency Balancing:** The second feature addresses a concern that arises due to scaling up sparsity-aware computing units. Fig. 7.4 b) (Left) shows two macros with varying degrees of sparsity. Notice that that red macro is sparser than the gray macro. Based, on our sparsity-aware ADC precision reconfiguration explained in Sec. 7.4, the red macro would require 2b ADC and the gray macro would require a 4b ADC. If the outputs produced by these two macros need to processed further, we have to wait

for the macro with 4b ADC to complete processing. If certain rows are exchanged between to the two macros to balance the sparsity, both macros will use a 3b ADC and finish faster than the unbalanced case. The proposed CIM processing core supports re-arrangement of input and weight elements to enable such latency balancing.

- **Run-time Sparsity Controller:** The proposed CIM processing core also features a run-time sparsity controller to calculate the data sparsity in input vectors as well as compare with pre-compiled weight sparsity to decide the precision of ADC that macros need to be operating at. Since the precision determination is performed during run-time, it needs to be fast and have low area and energy overhead.

7.5.3 Sparsity Aware CIM Compute Unit (SCU) Microarchitecture

First, we describe the microarchitecture of the Sparsity Aware CIM Compute Unit (SCU). The microarchitecture derives inspiration from the MVMU microarchitecture, described in [15], but adds sparsity control and precision-based reconfigurability features to the original design. It has four primary components:

- Sparsity Controller (SC)
- CIM Macros
- Reconfigurable Shift and Add Circuits (RSnA)
- Control Unit

The SCU can perform multiplication operation between an input vector and weight matrix. The precision of the weight matrix can be either 4 or 8 bits, whereas the precision of the input vector can be anything between 1 to 8 bits. The SCU deploys bit-serial computing. Consider a MVM operation between 8-bit input vector and 8-bit weight matrix. The input vector is divided into 8 1-bit vector streams, where as the weight vector is sliced into 8 1-bit slices, and mapped on to the 8 CIM macros in each SCU. Each CIM macro is of size 64×64. Thus, the SCU can perform 64×64 8-bit and 64×128 4-bit computations in a single instruction. The mapping of the weight matrix on to the CIM macros is such that

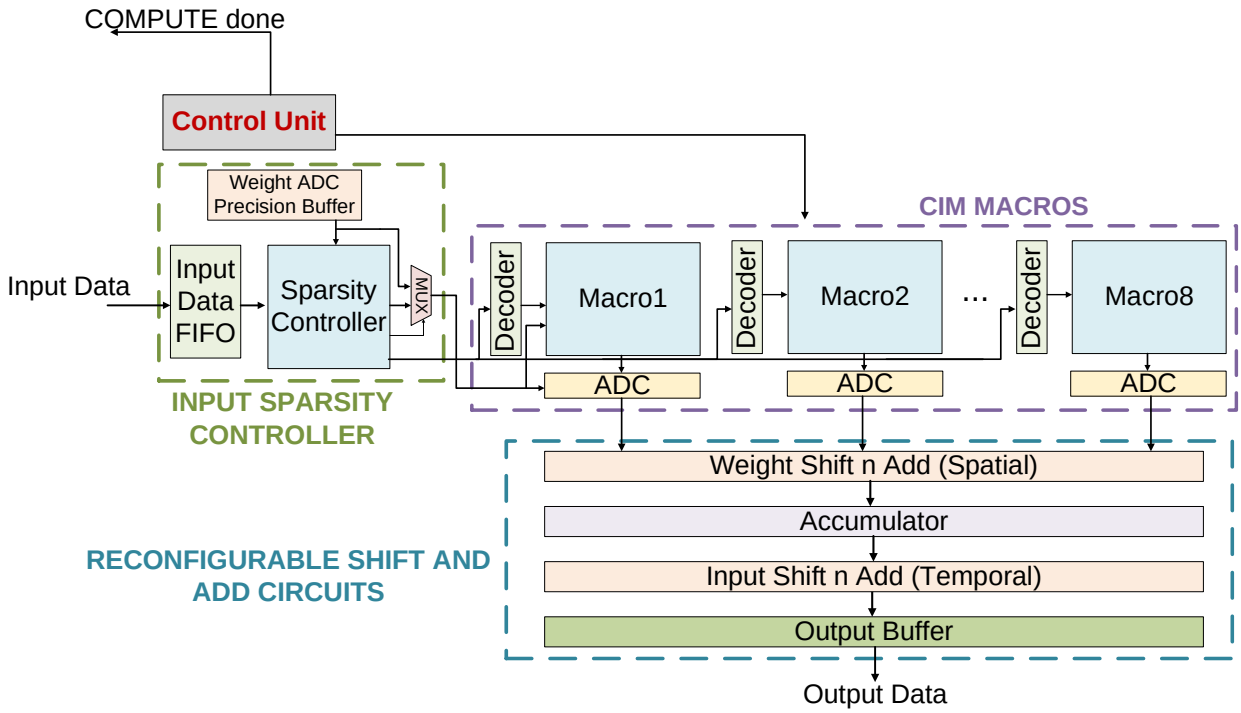


Figure 7.12. SCU Microarchitecture

Macro 1 holds the MSB, while Macro 8 holds the LSB. In case of MVM operations using 4-bit weights, we have two 64×64 matrices with 4-bit weights. The two weight matrices the 8 CIM Macros are divided into two sections of 4 CIM Macros, where Macro 1 to Macro 4 stores MSB to LSB of one weight matrix, and Macro 5 to Macro 8 stores MSB to LSB of the second weight matrix.

The shift-and-add circuits accumulates the result of the 8-bit MVM across different weight slices and input streams. Due to the sparsity-aware nature of the SCU, the shift-and-circuits need to be reconfigurable based on the precision at which the ADCs in the CIM macros are operating at.

The flow of computation occurs in four steps: i) The input Data FIFO buffer provides input streams to the SC, ii) the SC determines the precision of operation, and incorporates row-gating features and supplies data to the decoders in the CIM macros, iii) the CIM macros perform the computations and provides 1-bit MVM outputs to the RSnA, iv) The RSnA spatially accumulates the outputs from 8 macros to perform weight level shift and add operations, and temporally accumulates outputs to perform input level shift and add

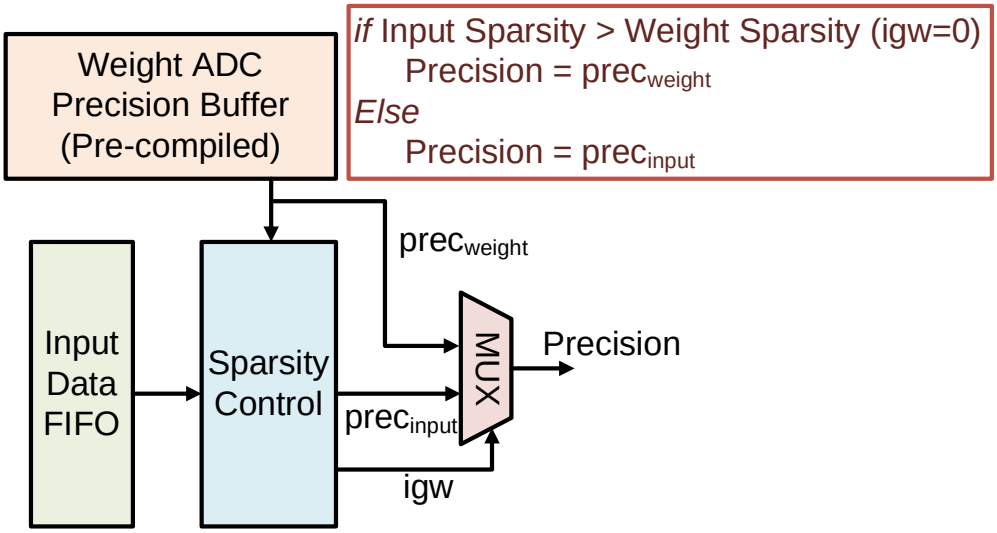
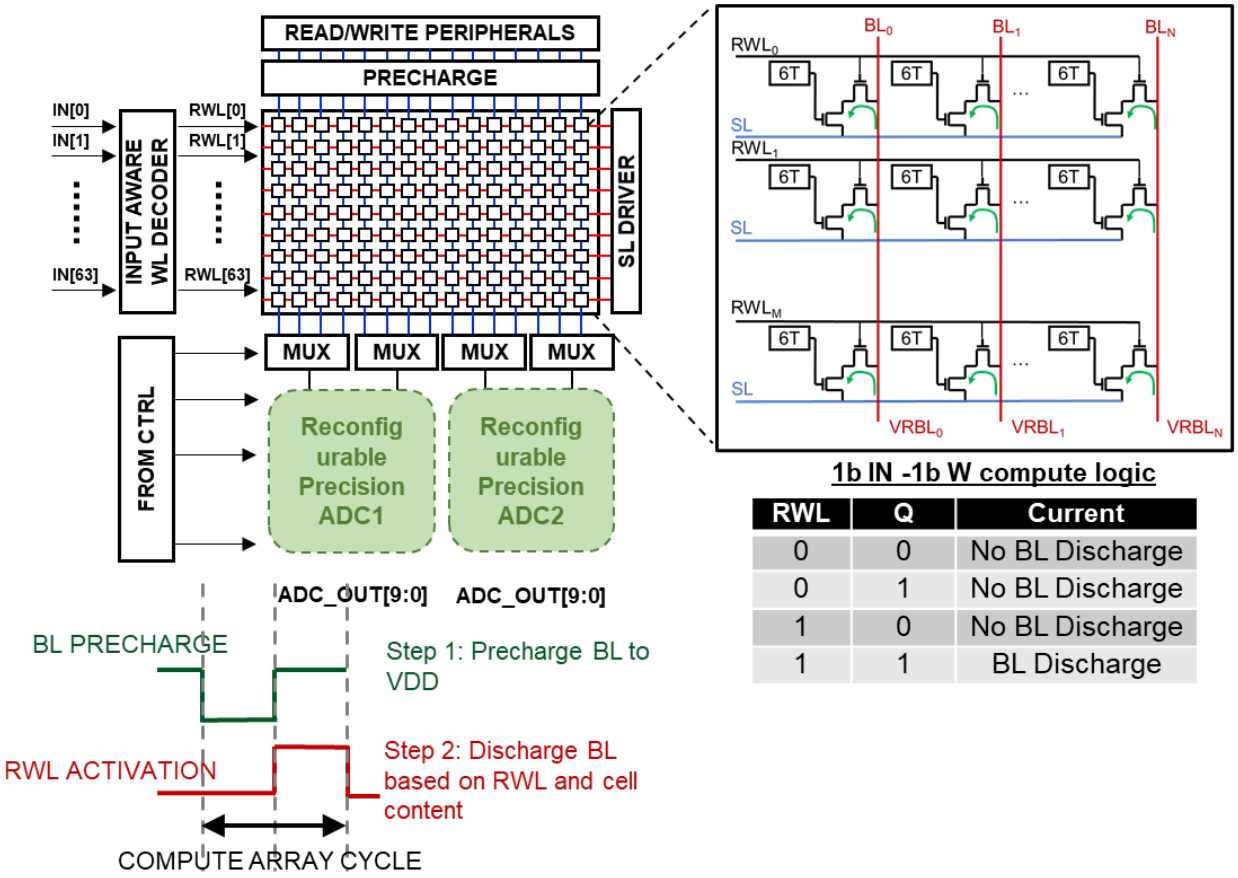


Figure 7.13. Sparsity Contoller Logical Diagram

operations to produce outputs for the 8-bit MVM. The last two steps of this computation flow are pipelined. In the next subsections, we describe the aforementioned components of the SCU micro-architecture in more detail.

Sparsity Controller (SC)

The Sparsity Controller (SC) determines the precision configuration for the ADC operation in the CIM macros. On receiving a 1-bit input vector from the input FIFO, the SC first performs row-gating based on the external row-gating signal to maintain a reasonable SNR of the CIM operation. For example, row-gating of a 64 width vector using 16 wide blocks, would result in 4 compute iterations to process the entire vector. Next, the SC deploys a ones counting circuit on the row-gated input to determine the input data sparsity. The weight data sparsity is pre-compiled and stored the Weight ADC precision buffer as shown in Fig. 7.13. A multiplexer decides which precision to use (input or weight) based on the signal 'igw', which is '1' when weight sparsity is lower than or equal to input sparsity and '0' otherwise. Thus, if input sparsity is higher than weight sparsity (igw = 0), we use the input ADC precision ($prec_{input}$) determined by SC, otherwise we use the pre-compiled weight ADC precision ($prec_{weight}$).



1b IN -1b W compute logic

RWL	Q	Current
0	0	No BL Discharge
0	1	No BL Discharge
1	0	No BL Discharge
1	1	BL Discharge

Figure 7.14. Sparsity-Aware CIM Macro with Reconfigurable Precision ADC, leveraging charge-domain computation

CIM Macro

The next component of the SCU are the CIM macros, capable of performing bit-wise MVM operations between the input vector and the weight matrix. The sparsity-aware CIM macro used in the SCU is different from the previously proposed sparsity-aware CIM macro, described in Sec. 7.4. The previous sparsity-aware CIM macro, deployed a current-based compute, whereas the CIM macro used in the SCU relies on bit-line discharge, similar to [14]. Fig. 7.14 shows the sparsity-aware CIM macro. It comprises of a 8T-SRAM array capable of performing MAC operations in the analog domain. The 1-bit compute logic performed by a single 8T-SRAM cell is shown in Fig. 7.14. First, the BL is precharged to V_{DD} using an active low pulse. When the RWL is '1', and the data stored in the cell (Q) is '1', the BL discharges based on the capacitance seen by the bit-line, along with the amplitude and

the width of the RWL activation pulse. To perform MAC operations, multiple RWLs are simultaneously activated, and that results in the BL discharge being proportional to the dot-product between the input vector and the corresponding weight column vector. The analog multiplexers (MUX) select which columns are turned on for compute. The capacitance at the BL is a combination of the intrinsic line capacitance, analog mux capacitance, and the input capacitance of the ADC capacitive DAC. The final voltage on the BL (V_{RBL}) is sampled by the previously proposed Reconfigurable Precision ADCs, described in Sec. 7.4.2. We have 4 analog Muxes and 2 ADCs, which implies we can perform four 2/3/5-bit conversion or two 4/6-bit ADC conversions in a single compute iteration. The sparsity-aware CIM macro is equipped with reconfigurable precision ADCs, which necessitates further adjustments in the BL discharge for different precisions at which the ADCs operate at. Since the BL discharge is the function of the capacitive DAC in the ADC, different precision results in different V_{RBL} for the same input and weight conditions. As a result, in order to maintain the same dynamic range of V_{RBL} for all the precision configurations, we need to either vary the amplitude or the width of the RWL activation pulse. Varying width of the pulse can be complicated in a synchronous design. Therefore, we include a input WL decoder circuit, which can select appropriate RWL pulse amplitude for each precision.

For more details on the design of Reconfigurable Precision ADCs, please refer to Sec. 7.4.2.

Reconfigurable Shift-and-Add Circuits (RSnA)

The final component of the SCU microarchitecture is the Reconfigurable Shift-and-Add Circuits (RSnA). The primary purpose of the shift-add is to perform a two level (input and weight) accumulation to account for bit-serial computing for both inputs and weights. Since the MSBs and LSBs of a particular weight are mapped on to different macros, and being computed simultaneously, the weight level shift-and-add operation is spatial in nature. The input bits are streamed one by one, and as a result require temporal shifting and accumulation. Due to the precision reconfigurability of the macro, the shift-and-add circuit needs to incorporate reconfigurability in its circuits.

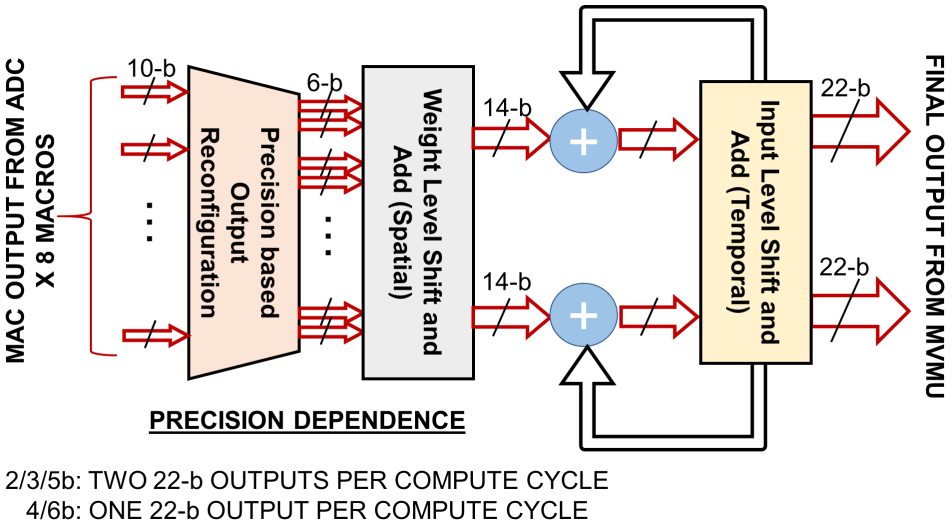


Figure 7.15. Reconfigurable Shift-and-add circuit logical flow

The ADC output is fed to the RSnA, as shown in Fig. 7.15. Note that the figure shows the case of one ADC as an example, whereas the CIM macros have two ADCs each, which would mean the same flow being replicated for the other ADC. The RSnA computation flow happens in 4 steps: i) A precision based reconfiguration is performed where based on the 10 bits received from the 8 ADCs of 8 macros, it is interpreted as either two outputs of 2/3/5 bits, or one output of 4/6 bits, from each macro, ii) Next, the outputs from 8 macros are spatially shifted and added to produce two (in case of 2/3/5 bit precision) and one (in case of 4/6 bit precision) 14-bit output(s), iii) This result is added to an accumulator, along with results for subsequent bit-streams obtained from the input level shift and add, iv) Finally, the input shift and add circuit temporally assimilates results for all the bit streams to produce two or one 22-bit outputs depending on the precision.

The entire computation flow of the CIM macros and RSnA is iterated for a given row-gated input vector, for all the columns in the macros in a pipelined fashion. Once all the columns are processed, the SC either provides the next block of the row-gated input vector or the next input vector (in case of no row-gating), and the computation flow is repeated until all the bits of inputs are exhausted. This synchronization is controlled by the control unit, shown in Fig. 7.14, which also provides timing signals to the CIM macro arrays for RWL activation, BL precharging etc, ADCs for BL sampling, and conversion cycling, and to the RSnA for its operation.

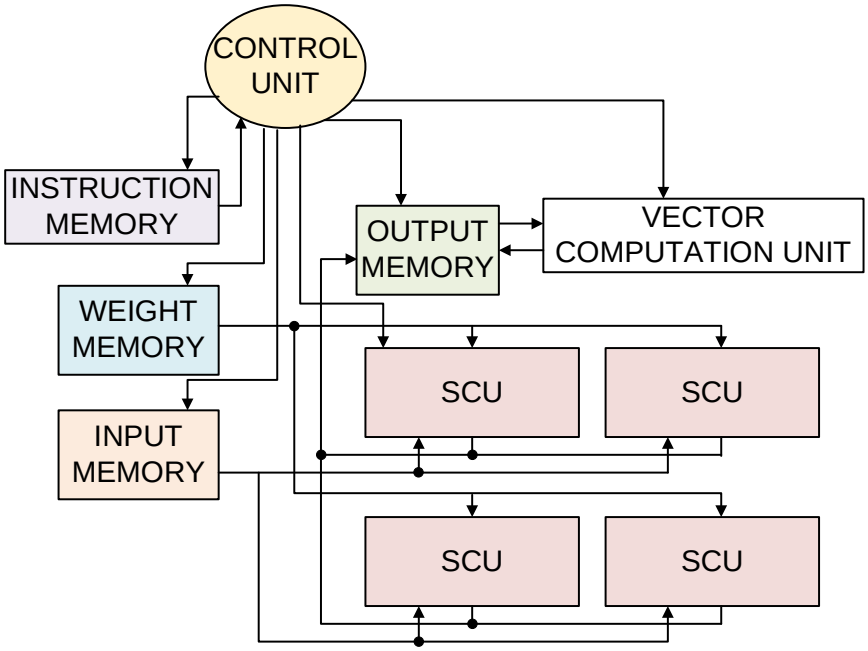


Figure 7.16. Core Microarchitecture

7.5.4 Core Microarchitecture

Having described the SCU microarchitecture, we move to the the design of the processing core microarchitecture. The top level micro-architecture of the sparsity-aware processing core is shown in Fig. 7.16. It consists of 4 SCUs, external memories dedicated to weights (Weight Memory), inputs (Input Memory) and outputs (Output Memory) respectively, an Instruction Memory to program the instructions into and a Vector Computation Unit to perform vector operations such as addition, as well as re-arrangement for load-balancing support.

7.5.5 Mapping and Dataflow

We have briefly described the internal mapping of weights and inputs in the SCU for bit-serial compute. Each SCU can execute MVM instructions to perform arithmetic operations between 64×1 n-bit ($1 \geq n \leq 8$) vector and 64×64 8-bit matrix or 64×128 4-bit matrix. Thus when 4 SCUs are combined in a $2x2$ organization, a single MVM instruction can enable MVM operation between 128×1 n-bit ($1 \geq n \leq 8$) vector and 128×128 8-bit matrix or 128×256 4-bit matrix.

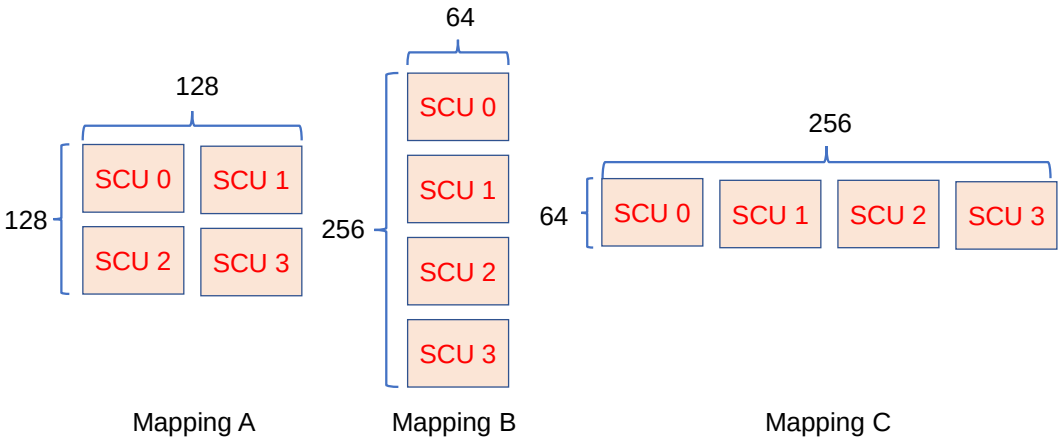


Figure 7.17. Different SCU mappings

The core microarchitecture can follow several mappings and dataflows. For example, the 4 SCUs can be rearranged in 2×2, 4×1 and 1×4 configurations. Fig. 7.17 shows the different spatial arithmetic configuration due to the 3 different mappings A, B and C. To understand different mappings and dataflows, let us consider a convolutional layer problem where the image size is 32×32 ($P \times Q$) and the layer size is $3 \times 3 \times 256 \times 256 (R \times S \times C \times M)$. First the weight matrix is flattened into a 2-D matrix 2304×256 ($RSC \times M$). Since the 4 SCUs can perform 8-bit 128×128 MVM operation in a 2×2 configuration, one possible mapping is shown below:

```
for M = 1 to 2
    for RSC(1) = 1 to 18
        for P = 1 to 32
            for Q = 1 to 32
-------------------------------
SCU Level
                for RSC(2) = 1 to 128 (Spatial)
                    for M = 1 to 128 (Spatial)
```

The above mapping deploys a weight-stationary dataflow, i.e., the weights in SCUs remain stationary while the entire input image is stridden. Then the next set of input channels (RSC) are fetched from the external weight memory and corresponding input channels are

fetched from the input memory. Finally, once all iterations of RSC is over for the weights, the next set of output channels (M) are fetched from the weight memory and the entire input image is re-fetched to perform the next set of computations.

Another possible mapping is a shown below which maximizes partial sum reduction to reduce overhead on output storage.

```
for M = 1 to 2
    for P = 1 to 32
        for Q = 1 to 32
            for RSC(1) = 1 to 18
-----------------------------------
SCU Level
                    for RSC(2) = 1 to 128 (Spatial)
                        for M = 1 to 128 (Spatial)
```

In this mapping, for the same input stride, all the input channels (RSC) are exhausted, such that all the partial sums are reduced to produce the all pixels of the first 128 output channels (M). Once the entire image is stridden, the next set of output channels (M) are fetched from the weight memory, and the inputs are re-fetched to complete the layer computations.

The sparsity-aware CIM processing core comprises of an instruction memory which stores the instructions. The instruction sequence follows the simple finite state machine of 'Fetch–>Decode–>Execute'. The execution cycles of different instruction can vary, since MVM operations tend to take significantly more cycles than load/store instructions or vector operations.

7.5.6 Tentative Floorplan

The floorplan of the SCU, presented in Fig. 7.12 and the processing core, presented in Fig. 7.16 is described next. Fig. 7.18 shows the SCU and procesing core floorplan. The SCU has 8 macros, distributed into 4 macros each on either side of the peripheral units, namely, the control unit (SCU CTRL) and the reconfigurable shift-and-add (RSnA) unit. The macro size is roughly $200\mu m \times 200\mu m$, and four macros together occupy an area $800\mu m \times 200\mu m$

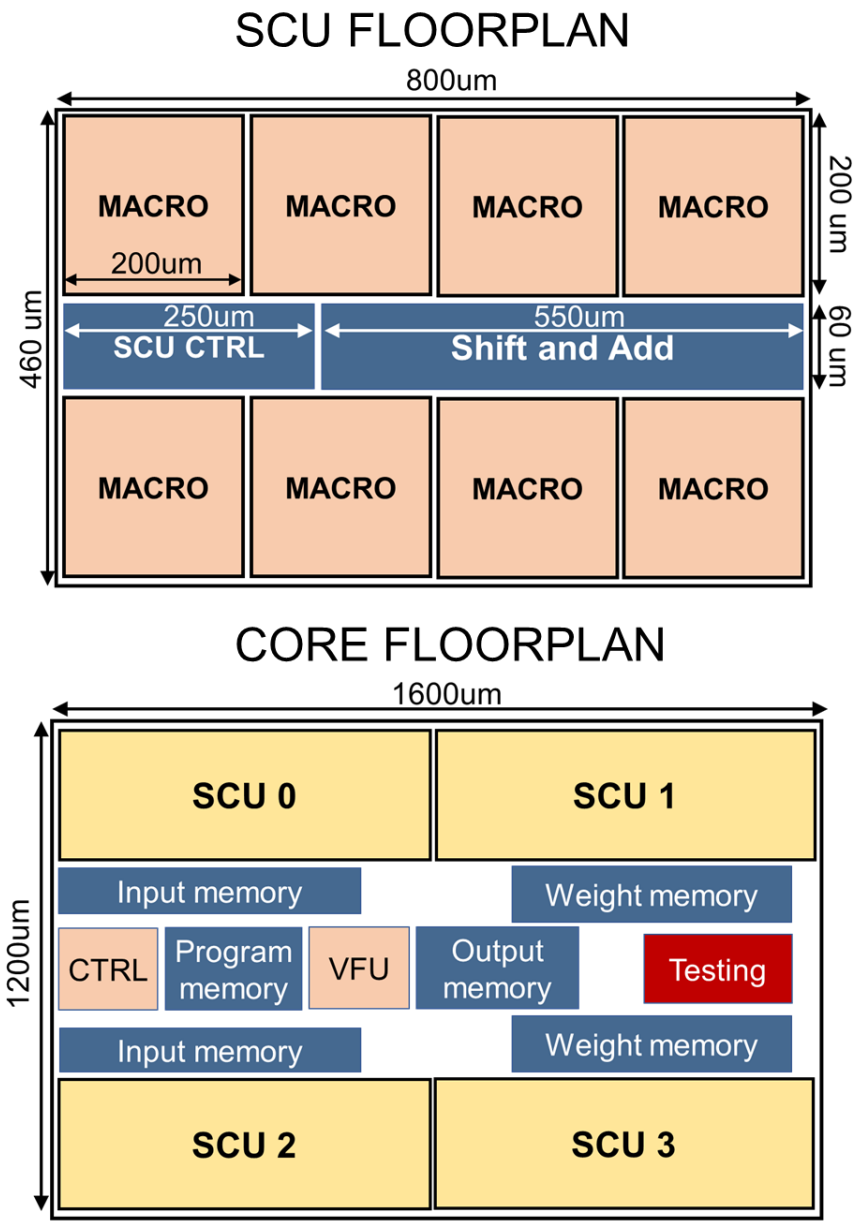


Figure 7.18. Tentative SCU and Core Floorplan

on each side. The control unit along with the input sparsity controller is roughly planned as $250\mu m \times 60\mu m$ and the RSnA unit is $550\mu m \times 60\mu m$. The total SCU area estimate is $800\mu m \times 460\mu m$. The processing core floorplan is described next. We have 4 SCUs in the processing, distributed equally on either side of the other memory and control units. Since each SCU occupies $800\mu m \times 460\mu m$, the four SCUs occupy $1600\mu m \times 460\mu m$ on either side.

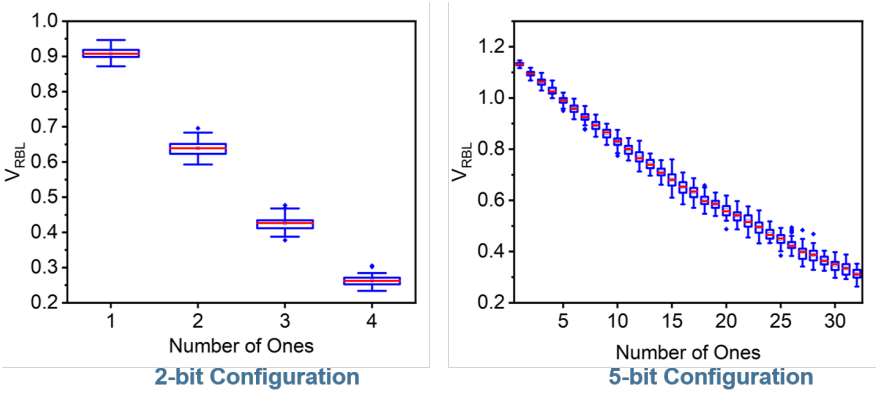


Figure 7.19. Simulation Results for Macro

The rest of the chip space is occupied by different memory units such as input memory, weight memory, output memory and instruction (or program) memory.

7.5.7 Preliminary Results

Next, we discuss some preliminary results obtained by performing simulations using the schematic design of the proposed SCU microarchitecture. For scaling up analog CIM macro to a processor micro-architecture, it is necessary to consider macro to macro variations. Fig. 7.19 shows the box-plot results using Monte-Carlo simulations for 2-bit and 5-bit configurations for the proposed macro. We observe that for 2-bit configuration, there is reasonable distinction between the different states, under V_t variations. However, for the 5-bit configuration, we observe overlap between adjacent states, which would result in computational errors. Usually, DNN workloads are error-resilient. However in case the workloads demand higher signal to noise ration, our chip can configured to operate under row-gating conditions, which will enable less number of word-lines simultaneously, achieving higher SNR as well as lower precision ADC.

Fig. 7.20 shows the macro layout and the energy and area distribution for different components of the macro. The proposed macro has a input decoder, the 8T-SRAM CIM array, and two ADCs. Contrary to the macro presented in 7.4, we do not require a TIA. Hence, we observe that the ADCs consume 53.8% of the area and 73% of the energy in the macro. Despite the high area and energy consumption of the ADC, we have reduced the total energy consumption of the macro by 20%.

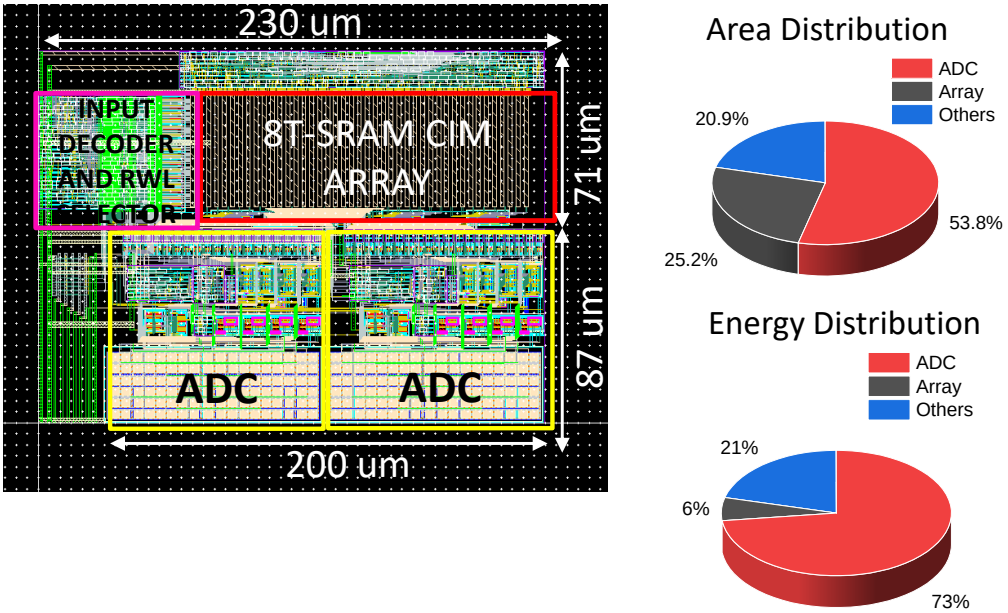


Figure 7.20. Macro Layout and Energy/Area Distribution of components

Next, we present the preliminary energy and area breakdown of the SCU in Fig. 7.21. The SCU has 4 primary components, the input FIFO, the sparsity controller (SC), the macro, the reconfigurable shift-and-add (RSnA) and the control unit. We observe that the macro consumes 79.5% and 72.4% of the total area and energy consumed by the SCU respectively. This breakdown is reasonable because we want most of the area and energy consumed by the component responsible for computing. However, we also observe that RSnA consumes 16.3% and 21.4% of the total area and energy consumed by the SCU respectively. This peripheral overhead is undesirable and needs to be optimized further.

Finally, we present the estimated performance numbers for the entire sparsity-aware processing core consisting of 4 SCUs. Each SCU can perform a 8-bit MVM operation between a 64×1 vector and a 64×64 matrix. Considering 1-bit MAC operations as 2 operations (multiply and add), an 8-bit 64×64 MVM involves $64 \times 64 \times 8 \times 8 \times 2 = 524288$ 1-bit MAC operations. The processing core achieves a performance ≈ 1.4 TOPS/s when operated under 2-bit configuration. The performance varies from $\approx 0.35 - 1.4$ TOPS/s as the ADC precision is varied from 2-bit to 6-bit.

The proposed sparsity-aware CIM accelerator core will be a scaled up demonstration of multiple analog CIM cores with various features such as row-gating and latency balancing

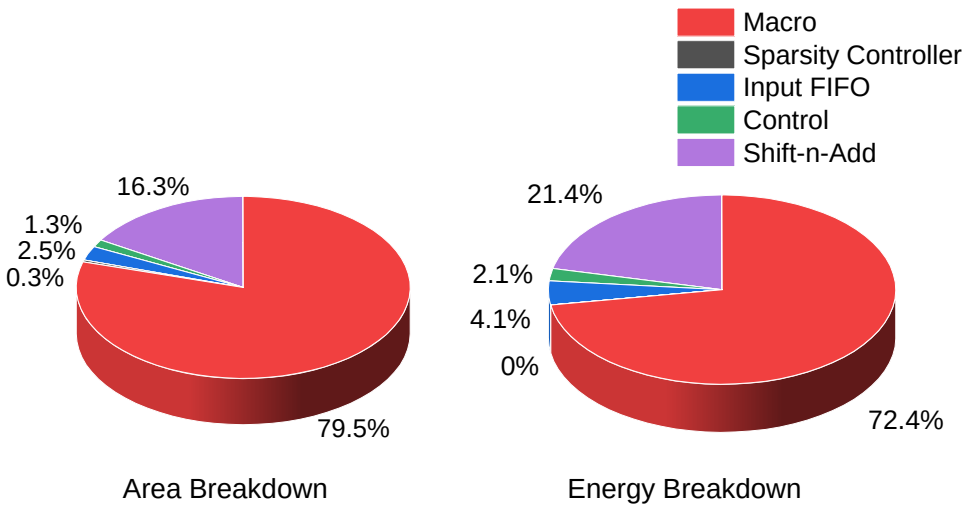


Figure 7.21. Preliminary SCU Energy/Area Breakdown

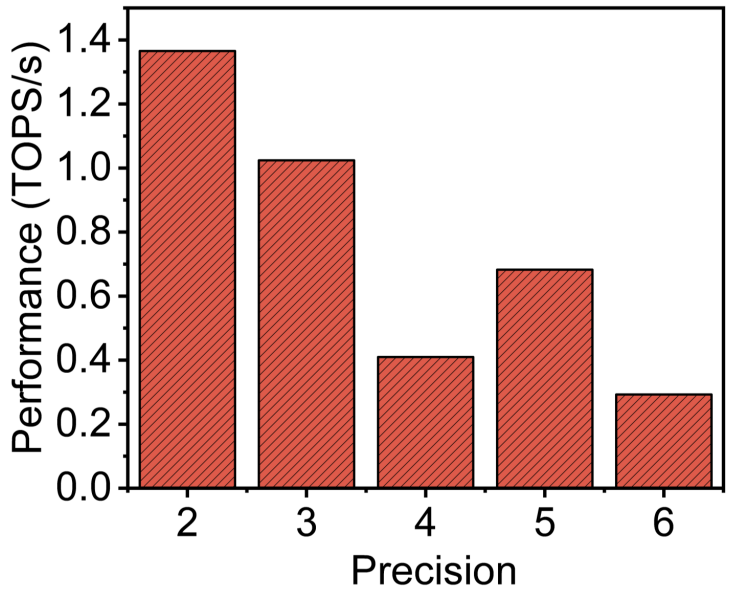


Figure 7.22. Estimated SCU Performance in TOPS/s

techniques to solve micro-architectural challenges of scaled analog CIM design. The processing core can support flexible mapping and dataflow, which will enable optimality studies of various CIM mappings and dataflows.

8. NEUROMORPHIC COMPUTING USING GST-BASED PHOTONIC PLATFORMS

8.1 Introduction

The phenomenal success in the field of Deep Learning using Artifical Neural Networks (ANN) based on analog information processing has had far reaching consequences in the past decade [221]. Machines driven by such networks have surpassed human in various tasks ranging from pattern recognitions to playing complex games such as AlphaGo [222] and Chess [223]. However, the growing complexities of computational models involved in such multi-layered neural networks have rendered the training and inferencing tasks extremely expensive in terms of memory and energy. To that effect, there have been significant advancements in the field of neuromorphic computing which largely rest on our understanding of the human brain as researchers strive to comprehend the intricacies of its complex functionalities and emulate its unparalleled energy efficiency. Despite the obvious elusiveness of the brain, neuroscientific experiments have unravelled various underlying mechanisms behind our behavorial patterns. To that effect, various studies have been performed exploring phenomena concerning the basic functional units, namely neurons and synapses, that knit the neural network in the human brain. The need to incorporate these neuroscientific findings in computing models and consequently in building bio-plausible hardware has led to extensive investigations in recent years.

Most of the available computing models that encode the information processing in a neural network are based on mathematical optimization techniques. More recently, with growing evidence of spike-based processing in the biological neural network, its event-driven nature has led researchers to explore bio-plausible hardware implementations in an effort to achieve higher energy efficiency. Spiking neural networks (SNN) comprise the third generation of neural networks and the basic principle relies on how the membrane potential of a spiking neuron rises and eventually cause the neuron to spike under the action of incident spikes. Mathematically speaking, the fundamental operations performed by such SNNs involve parallelized dot-product through the synaptic network followed by subsequent integration and

thresholding by the neurons. Neuromorphic systems attempting to leverage the sparse and event-driven nature of SNNs thus aim toward efficient emulation of these functionalities.

The initial efforts [126], [224], [225] in hardware implementations of SNNs was based on standard von-Neumann architecture [181] based on CMOS technology where the synaptic units of the neural networks are stored in the digital memory and repeatedly fetched by the processor for computing operations. However, the overhead of frequent data transport between the memory and processor have led to a shift in the computing paradigm as 'in-memory' computing platforms [24], [226] attempt to emulate the 'massively parallel' operations of the brain. Moreover, hardware implementations of various spiking neuron models such as Hodgkin-Huxley[227] and Leaky-Integrate-Fire (LIF)[228] on CMOS platforms not only fail to match the energy efficiency of the human brain but is also area-inefficient. As a result, although the term 'neuromorphic' was primarily coined [229] with CMOS technology in mind, this computing domain has branched out to non-volatile memory (NVM) technologies such as oxide-based memristors [183], spintronics [74], phase change materials (PCM) [137], [230], etc in the recent years. The natural ability of these resistive technologies to compute parallelized dot-products using crossbar structures make them promising candidates for neuromorphic systems. These novel material systems and technologies [74], [230] have been proposed to mimic the behavior of a spiking neuron thus providing direct mapping between a single device behaving as a functional neural element. However, each technology suffers from different drawbacks, such as energy-efficiency, speed, cross-talk, fabrication difficulties, etc. Integrated Photonics offers an alternative approach to standard microelectronic 'in-memory' computing platforms and promises ultra-fast neural computing and information processing. The recent advances in photonics-based neuromorphic computing has overseen implementations of various kinds [231], [232] of neural processing units on the photonic platform leveraging the inherent capability of matrix operations of integrated optical circuits. Spike-based processing systems have also been extensively explored using excitable lasers [233], [234]. However, most of the photonic systems investigated in the context of neuromorphic computing are based on volatile information processing. Non-volatility offers the ability to write and erase information dynamically desirable for large-scale implementations of neuromorphic systems. Phase change materials (PCM), in particular, have

been demonstrated [235] to have significant energy restrictions due to their high 'write' times in the electrical domain. It has been shown that either the exciting current or 'write' pulse duration has to be reduced by 10× for PCM to perform better than CMOS. However, recently PCMs, e.g. GST, have been demonstrated[236] to achieve sub-ns 'write' speeds when excited by photonic laser pulses. Due to highly contrasting optical and electrical properties in their amorphous (a-GST) and crystalline (c-GST) states, PCMs have thus offered avenues to implement all-photonic memories[237], [238], switches[236] and have been even used for mixed-mode electro-optical operations [239]. The promise of fast information processing with PCMs in the photonic domain has thus encouraged the possibilities of PCMs as a viable material for photonic neuromorphic systems. Recently, device[240] based on GST deposited on waveguides was proposed to emulate the synaptic weight update mechanism in synapses in SNN frameworks. In this work, we propose an all-photonic operation of an Integrate-and-Fire spiking neuron. We show that the proposed neuron mimics the behavior of the biological neuron. Next, we propose an all-photonic Spiking Neural Network, based on GST-based photonic neural elements, which attempts to bridge the gap between devices to system-level implementation of Photonic neural networks. We leverage the inherent wavelength division multiplexing (WDM) [241] property of optical networks to propose a non-volatile synaptic array, while exploring and mitigating the challenges arising from designs based on ring resonators of radii comparable to the wavelength of operation. Such a synaptic array can achieve higher densities compared to current state-of-art photonic computing systems. We show how the proposed synaptic computing platform can be seamlessly integrated with previously explored 'integrate and fire' spiking neurons to realize an ultra-fast and truly integrable Spiking Neural Network. Finally, we evaluate the performance of the proposed Photonic SNN in the classification task of handwritten digits.

8.2 GST on Micro-ring Resonators

The basic working principle of a ring resonator is necessary to be illustrated at first. A ring resonator is a structure with two rectangular waveguides and a ring waveguide (as shown in Fig. 8.1 (a)). Wave entering through the 'INPUT' port gets partially coupled to

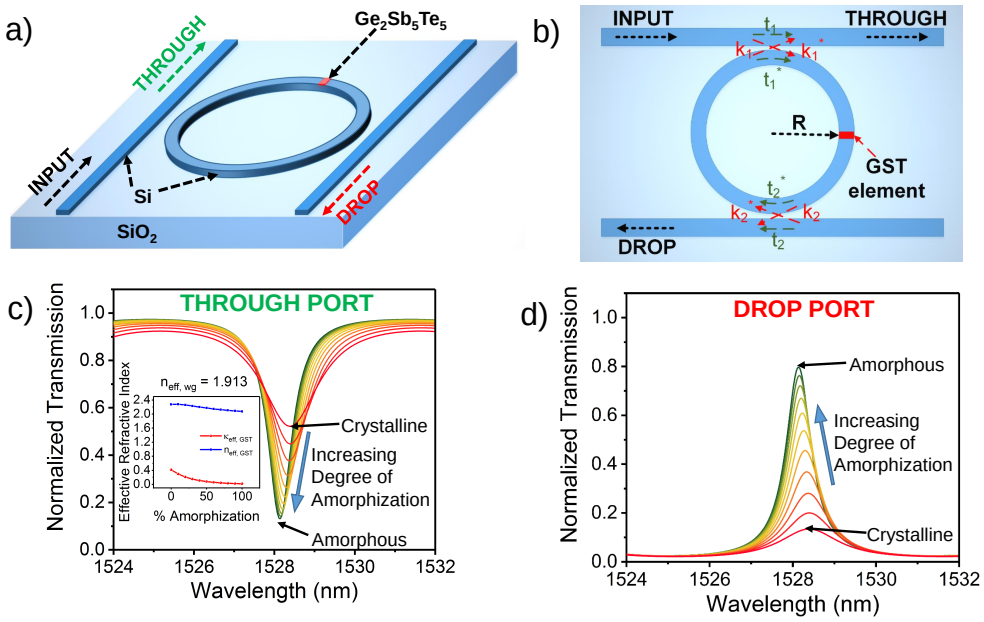


Figure 8.1. (a) A perspective view of an add-drop microring resonator with a small patch of GST on top showing its ports and materials. (b) A two-dimensional top view of the ring resonator illustrating the input, output, coupling and transmission parameters. Theoretically calculated transmission at various wavelengths for different degrees of amorphization of GST ranging from 0% (crystalline) to 100% (amorphous) showing that the transmission at the (c) 'THROUGH' ((d) 'DROP') port decreases (increases) with increasing degree of amorphization.

the ring waveguide and interferes constructively inside the ring when the following condition, called resonant condition, is met:

$$2\pi R n_{eff,wg} = m\lambda_m \tag{8.1}$$

Eq.1 provides the resonant condition (at wavelengths λ_m) for the ring resonator of radius R where the effective refractive index of the waveguide-substrate material system is $n_{eff,wg}$. By controlling the coupling and attenuation parameters, t_1, t_2 and k_1, k_2, as shown in Fig. 8.1 (b), light can be conditionally guided through the 'THROUGH' and 'DROP' ports.

Introducing a GST element (shown in red in Fig. 8.1(a)) on top of the ring waveguide in the ring resonator described above allows us to control light propagation through the ports by merely changing the state of the GST. Light passing through the waveguide get evanescently coupled to the GST element and gets differentially absorbed by the GST in

its low-loss amorphous state and high-absortion crystalline state [237]. The difference in attenuation arises due to the contrasting imaginary refractive index (κ_{GST}) of GST in its two states. Theoretically, the transmission of the 'THROUGH' and 'DROP' ports can be expressed as:

$$T_t = \frac{t_2^2\alpha^2 - 2t_1t_2\alpha cos(\theta) + t_1^2}{1 - 2t_1t_2\alpha cos(\theta) + (t_1t_2\alpha)^2} T_d = \frac{(1-t_1^2)(1-t_2^2)\alpha}{1 - 2t_1t_2\alpha cos(\theta) + (t_1t_2\alpha)^2} \tag{8.2}$$

where α is the attenuation factor, θ is the phase factor, t_1 and t_2 are coupling parameters. α and θ can be expressed as:

$$\alpha = exp(-\frac{2\pi}{\lambda}[\kappa_{eff,wg}(2\pi R - L_{GST}) + \kappa_{eff,GST}L_{GST}]) \approx exp(-\frac{2\pi}{\lambda}\kappa_{eff,GST}L_{GST}) \tag{8.3}$$

$$\theta = \frac{2\pi}{\lambda}[n_{eff,wg}(2\pi R - L_{GST}) + n_{eff,GST}L_{GST}]. \tag{8.4}$$

Here $\kappa_{eff,GST}$ ($\kappa_{eff,wg}$) and $n_{eff,GST}(n_{eff,wg})$ are effective imaginary and real parts of the refractive index of the waveguide material with (without) GST. R is the radius of the ring waveguide and L_{GST} is the length of the GST element. The refractive indices of partially crystallized GST are estimated from effective permitivities approximated by an effective-medium theory[146], [242]:

$$\frac{\epsilon_{eff}(p) - 1}{\epsilon_{eff}(p) + 2} = p \times \frac{\epsilon_c - 1}{\epsilon_c + 2} + (1-p) \times \frac{\epsilon_a - 1}{\epsilon_a + 2} \tag{8.5}$$

where ϵ_c and ϵ_a are the permittivities in the crystalline and amorphous states respectively calculated from the refractive indices of GST[237] by $\sqrt{\epsilon(\lambda)} = n + i\kappa$. p is the degree of crystallization. The effective refractive indices of the Si waveguide- SiO_2 substrate system with and without GST was calculated using COMSOL Multiphysics simulations, shown in the inset of Fig. 8.1 (c).

These equations depict the theoretical backdrop of a ring resonator system with GST. As the GST element crystallizes (amorphizes), $\kappa_{eff,GST}$ and hence its absorption increases (decreases) and as a result the transmission at the 'THROUGH' ('DROP') port increases. Fig.

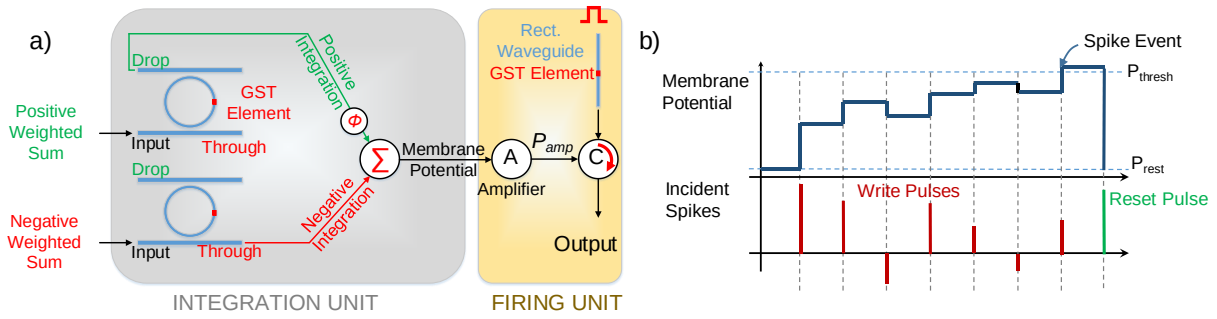


Figure 8.2. (a) Schematic of a bipolar integrate and fire neuron based on GST-Embedded Ring resonator devices showing the integration and firing unit. (b) Timing diagram showing the integration of membrane potential for various incident pulses demonstrating the operation of the proposed neuron.

8.1 (c) and 8.1 (d) shows that the theoretically calculated transmission at the THROUGH and 'DROP' ports in a ring resonator increases with p.

8.3 Toward Fast Neural Computing using All-Photonic Phase Change Spiking Neurons

We first propose an integrate-fire spiking neuron leveraging these characteristics of the GST-ring resonator system. Information processing in neural networks usually involve multiplication of inputs with the significance metric of the synapses, namely 'weight' and feeding the corresponding output to a neuron. For most neural network applications, weights can assume negative values. It is thus necessary to realize a bipolar neuron which can receive inputs of both polarity for all practical purposes. Let us now consider a GST embedded ring resonator described above. The GST initially is in crystalline state, denoting the highest (lowest) transmission level through 'THROUGH' ('DROP') port. During the 'write' phase, an off-resonance pulse is input which writes into the GST element, thereby reducing (increasing) its degree of crystallization p (amorphization $(1-p)$). During the 'read' phase, as p reduces, 'THROUGH' port transmission T_t decreases and 'DROP' port transmission (T_d) increases. Thus, with incoming pulses, the transmission through the 'DROP' and 'THROUGH' ports get positively and negatively integrated respectively. We combine these properties of the device to propose a bipolar integrate and fire neuron. The integration

unit of the neuron body consists of two ring resonators as shown in Fig. 8.2 (a) and pulses of amplitudes proportional to the positive (O_j^+) and negative (O_j^-) weighted sums, received from the synapses, are fed to the positive and negative ring resonators respectively. The details of entire network framework is discussed later. Note, the resultant amplitude of the incident pulse to the neuron is the difference of the positive and negative inputs fed to the two devices: $O_j = O_j^+ - O_j^-$. Thus, the two ring resonators integrate in opposite direction to emulate the resultant integration which should ideally be proportional to O_j. The output from the 'DROP' and 'THROUGH' ports of the positive and negative devices respectively are passed to an interferometer. We place a phase modulator (ϕ) in the path of the positive ring resonator and the interferometer to tune the output of the interferometer to produce the sum of the two incoming pulses. As the two ports integrate in the opposite direction, the output of the interferometer is the resultant integration based on both the positive and negative inputs to the neuron body and can be treated as membrane potential of the integrate and fire neuron. Thus at every time-step, the membrane potential of the j^{th} neuron can be represented by:

$$V_j[t] = V_j[t-1] + O_j[t] \tag{8.6}$$

Fig. 8.2 (b) shows the operation of the proposed neuron such that the membrane potential integration is proportional to the amplitude of the resultant incident spike to the neuron. Once the GST reaches full amorphization, the membrane potential crosses its threshold (P_{thresh}). The 'firing' action of the neuron involves the generation of a spike which is implemented by an additional photonic circuit as shown in Fig. 8.2 (a). This circuit consists of an photonic amplifier, a circulator and a rectangular waveguide with a GST element on top initially in crystalline state. For a rectangular waveuguide with GST, the transmission is low (high) in crystalline (amorphous) state. The 'read' and 'write' phases for the 'integration unit' and the 'firing unit' alternate in successive cycles. This essentially means that during the 'write' cycle of the integration unit of the neuron, a read pulse is passed through the firing unit. On the other hand, during the 'read' cycle of the integration unit, the 'read' pulse is passed through the ring resonators and based on the output of inteferometer and subsequent amplification, the resulting pulse attempts to write into the GST of the rectangular waveguide in

the firing unit. A circulator C directs the incoming and outgoing pulses into the rectangular waveguide. When the GST elements in the integration unit are initially in crystalline state, the output of the amplifier A (P_{amp}) is not sufficient to amorphize the GST on rectangular waveguide and hence, a spike is not transmitted through the rectangular waveguide. However, when the membrane potential integrates, on incidence of several 'write' pulse, enough to the cross the threshold, P_{amp} is ensured to be high enough to amorphize the GST on the rectangular waveguide and a spike is transmitted. Once the neuron fires, a 'RESET' pulse is passed to reset the states of the devices to their initial states and the membrane potential drops to the resting potential (P_{rest}) as shown in Fig. 8.2 (b). Thus, the operation of a bipolar integrate and fire neuron can be achieved using the setup described in Fig. 8.2.

The dynamics of the spiking neuron is primarily governed by the phase-change dynamics of GST. GST partially absorbs the wave passing through the ring waveguide and its low thermal conductivity [243]causes a considerable increase in temperature. The growth of the amorphization region in the material occurs when the concerned region is above the melting temperature, which is around 877K [244]. For a particular incident pulse, the amorphous region heats up less than the crystalline region. Thus the change in amorphous thickness will decrease as the amorphous thickness increases. Thus, change in amorphization thickness is a function of the current state of the GST and the amplitude of the incident pulse.

Results

The 'write' operation of the spiking neuron is investigated using the modal profiles of the incident EM waves and the resulting temperature profiles in the GST-Si-SiO_2 stack. The 'read' operation, on the other hand, is explored from the point of view of the entire GST-ring resonator system. The modal profile of input EM wave and subsequent heat dissipation framework was implemented in COMSOL[250]. The temperature profiles were used to simulate the phase change characteristics of GST in MATLAB®. The optical response of a ring resonator was obtained using a commercial-grade simulator Lumerical FDTD Solutions based on the finite-difference time-domain (FDTD) method[251]. Table. 8.2 lists the parameters used for each simulation.

Table 8.1. Dimensions and Material parameters

Dimensions		Material Parameters	
Parameters	Values	Parameters	Values
Ring Resonator Radius (R)	$6\mu m$	Si Refractive Index (n_{Si})[245]	3.5
Si Waveguide Cross-section	0.4×0.18 μm	SiO_2 Refractive Index (n_{SiO_2}) [246]	1.4
Upper Coupling Gap (L_{upper})	0.1 μm	c-GST Refractive Index ($n_{c-GST} + i\kappa_{c-GST}$)[247]	7.2+1.9i
Lower Coupling Gap (L_{lower})	$0.1\mu m$	a-GST Refractive Index ($n_{a-GST} + i\kappa_{a-GST}$)[247]	4.6+0.18i
GST Length (L_{GST})	$0.3\mu m$	c-GST Specific Heat,(C_{c-GST})	217J/kg.K
GST Width (W_{GST})	$0.3\mu m$	a-GST Specific Heat,(C_{a-GST})	217J/kg.K
GST Thickness (t_{GST})	20 nm	c-GST Thermal Conductivity,(k_{c-GST})[238]	0.59W/m.K
		a-GST Thermal Conductivity,(k_{a-GST})[248]	0.19 W/m.K
		c-GST Density,(ρ_{c-GST})[249]	6270 kg/m^3
		a-GST Density,(ρ_{a-GST})[249]	5870 kg/m^3

Phase change dynamics of GST

The electromagnetic power absorption and subsequent temperature rise in GST is analyzed in detail using Finite Element Method (FEM) simulations in COMSOL Multiphysics. Firstly, to validate our simulation framework we simulated a GST embedded Si_3N_4-SiO_2 ridge-waveguide system and compared its transient response of temperature in GST with experimental data[238] under same excitation conditions. Fig. 8.3 (a) shows good agreement between the results from our simulation and corresponding experimental data, thus validating our simulation setup. Next, we built a 3D model of a section of the ring resonator with GST as shown in Fig. 8.3 (b) and studied the electromagnetic characteristics and subsequent temperature profiles using the validated simulation setup. The dimensions of the waveguide were fixed to ensure single fundamental mode propagation for a input optical wave of 1550 nm length. The electric field distribution at the surface of the waveguide embedded with c-GST and a-GST are shown in Fig. 8.3 (c) and (d) respectively. We observe optical atten-

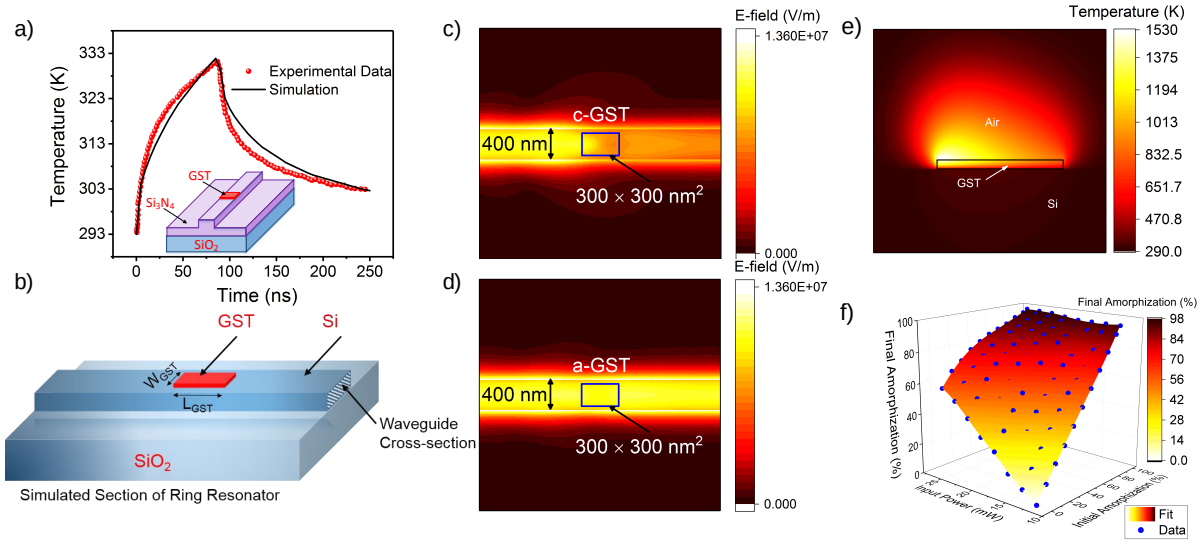


Figure 8.3. (a) Experimental benchmarking on a Si_3N_4-SiO_2 ridge-waveguide system, validating our simulation framework. (b) Simulated volume of the GST section in the ring resonator described in Fig. 8.1 (a) delineating the different materials used. Surface electric field propagation of (c) c-GST and (d) a-GST shows significant contrast. (e) Temperature distribution along the length of cGST. (f) Plot of final percentage amorphization as a function of initial percentage amorphization and input power.

uation of - 3.71 dB in the waveguide for c-GST of 0.3 μm length and 20 nm thickness while similar dimensions of a-GST give us negligible (- 0.26 dB) attenuation. This implies strong optical absorption in c-GST and also validates the fact that it is an order of magnitude higher than that of amorphous state[238]. This property allows us to progressively amorphize our device while keeping the state of the already amorphized volume undisturbed for our chosen range of input optical power.

Next, we analyze the thermal response of the GST upon optical excitation using finite element simulation. We incorporate optical heating by modeling GST as local heat source. An optical pulse of amplitude $26mW$ and duration $200ps$ is injected from the front facet of the waveguide. The GST is initially considered to be in crystalline state and absorbed energy in GST is taken as the heat energy for that local heat source. However, as heat is not generated uniformly within the GST volume, we designed the heat source to decrease exponentially[238] with a factor, $A = exp(-|\alpha_x| \cdot x \cdot ln(10)/10)$ along the length of the GST

$(0 \leq x \leq L_{GST})$ where α_x is the optical attenuation per unit length of GST. Resulting temperature distribution at the end of the pulse is shown in Fig. 8.3 (e). From inspection of this profile, an exponential temperature distribution along the GST length becomes evident. We also observe that there exists a significant portion of GST whose temperature is above the melting temperature (877K) and hence will become amorphized (e.g. 57% amorphization for given conditions) after removal of optical pulse. This simulation was performed multiple times keeping the pulse width same but varying the pulse power (amplitude) and initial level of amorphization and results are plotted in Fig. 8.3 (f). We find that below 12mW (200 ps) input pulse, irrespective of initial amorphization state, no further amorphization happens. Thus, we choose a input power range (26mW to 12mW) for the operation of the proposed all-photonic spiking neuron.

Optical response of ring resonator

The 'read' operation of the spiking neuron concerns with the optical response of the ring resonator or more precisely, the transmission characteristics at the 'THROUGH' and 'DROP' ports of the device. FDTD simulations were performed in Lumerical. Inc on a ring resonator with Si waveguides and SiO_2 substrate with a patch of GST on top of the ring waveguide as illustrated in Fig. 8.1 (a). Fig 8.4 (a) and (b) shows the normalized transmission at the 'THROUGH' and 'DROP' ports for different amorphization levels of GST. The insets of Fig 8.4 (a) and (b) show the variation of transmission at a resonant wavelength $\lambda_{read} = 1529nm$ with increasing degree of amorphization for the two ports respectively and results show consistency with our theoretical discussions above. The variation in transmission results from the decreasing absorption co-efficient (α) as the GST amorphizes. We observe a FWHM of 1.68 (2.23) nm for a-GST and 2.97 (2.97) nm for c-GST and an extinction ratio contrast of 7.5 (6.03) dB between the fully amorphous and fully crystalline states in the 'THROUGH' ('DROP') port. Fig. 8.4 (c) and (d) shows the visible contrast in electric field absorption by the GST element in the ring resonator for the amorphous and crystalline states of GST for an on-resonance incident wave. The slight shift in the resonance peaks can be attributed to

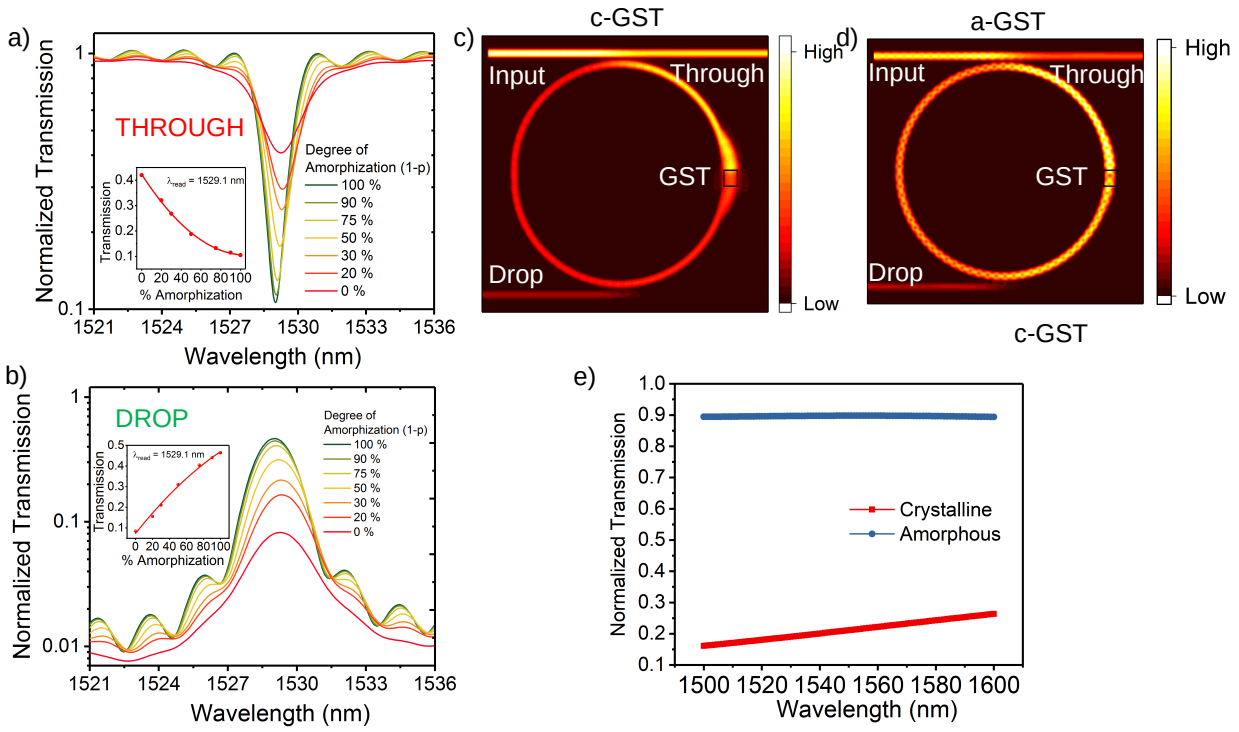


Figure 8.4. Normalized Transmission at the (a) 'THROUGH' and (b) 'DROP' ports with increasing degree of amorphization for a particular range of frequencies including a resonance peak at $\lambda_{read} = 1529.1nm$. As the degree of amorphization increases, transmission at 'THROUGH' ('DROP') port decreases (increases) thus realizing negative (positive) integration action of the neuron. (c) and (d) shows the top-view E-field distribution of a GST-embedded ring resonator for c-GST and a-GST showing higher field absorption for the former when the wave passes the GST region. (e) High contrast between c-GST and a-GST for the rectangular waveguide in the firing unit of the neuron.

the minor variations in the real part of the effective refractive indices of the GST at different states, which can be expressed as[236]:

$$\Delta\lambda_{read} \approx \frac{\Delta n_{eff,GST}}{n_{eff,wg}} \cdot \frac{L_{GST}}{2\pi R} \tag{8.7}$$

These characteristics show that the outputs at the 'THROUGH' and 'DROP' ports decrease and increase respectively with increasing degree of amorphization which is a desirable characteristic for integration in both the positive and negative direction. We leverage this characteristic by connecting the outputs from the 'THROUGH' and 'DROP' ports of two

devices to an interferometer, as shown in Fig. 8.2 (a) to obtain the resultant integration of the membrane potential as described earlier. Thus, the progressive optical responses of the ring resonator for various percentage amorphization are in agreement with the desired characteristics for the neuronal system to show integrating action. Finally, the contrast between transmission of a-GST and c-GST for a rectangular waveguide is shown in Fig. 8.4 (e).

Spiking Neural network inferencing framework

A neural network is comprised of multiple layers of neurons connected through synapses. The operation of any layer in a neural network involves computing the dot-product of the inputs and weights of the synapses, which gets transferred through the neuron to the next layer. To that effect, the synaptic network can be represented as a dot-product engine that multiplies the inputs with the corresponding synaptic weights and computes a weighted sum which is received by the neuron. Such a dot-product framework can be potentially implemented by GST-based photonic synapses. Such a synapse can draw its inspiration from a GST-based on-chip photonic synapse[240] recently proposed. The proposed integrate-and-fire spiking neuron can be integrated with these photonic synapses in an all-photonic implementation of a spiking neural network. To analyze the performance of such an all-photonic neural network, we built a device to algorithm framework by mapping the device characteristics to implement the proposed neuron in an algorithm level neural network inferencing setup. Such a system-level simulation is quintessential to validate the operation of the proposed integrate-and-fire neuron. For the current analysis, we assume ideal operation of the dot-product engine. We consider a fully connected network consisting of 3 layers, the input layer, the hidden layer and the output layer as shown in Fig. 8.5 (a). In such a network, each neuron receives inputs from all the neurons of the previous layer. We study the performance of the aforementioned fully connected neural network in a standard handwritten digit recognition task based on the MNIST dataset[252]. The MNIST dataset consists of 60000 training images and 10000 testing images. The weights of the synapses are trained using the Backpropagation algorithm [253] as in case of traditional Artificial Neural networks (ANN). During inferencing, we use a conversion mechanism [254] from ANN to SNN where

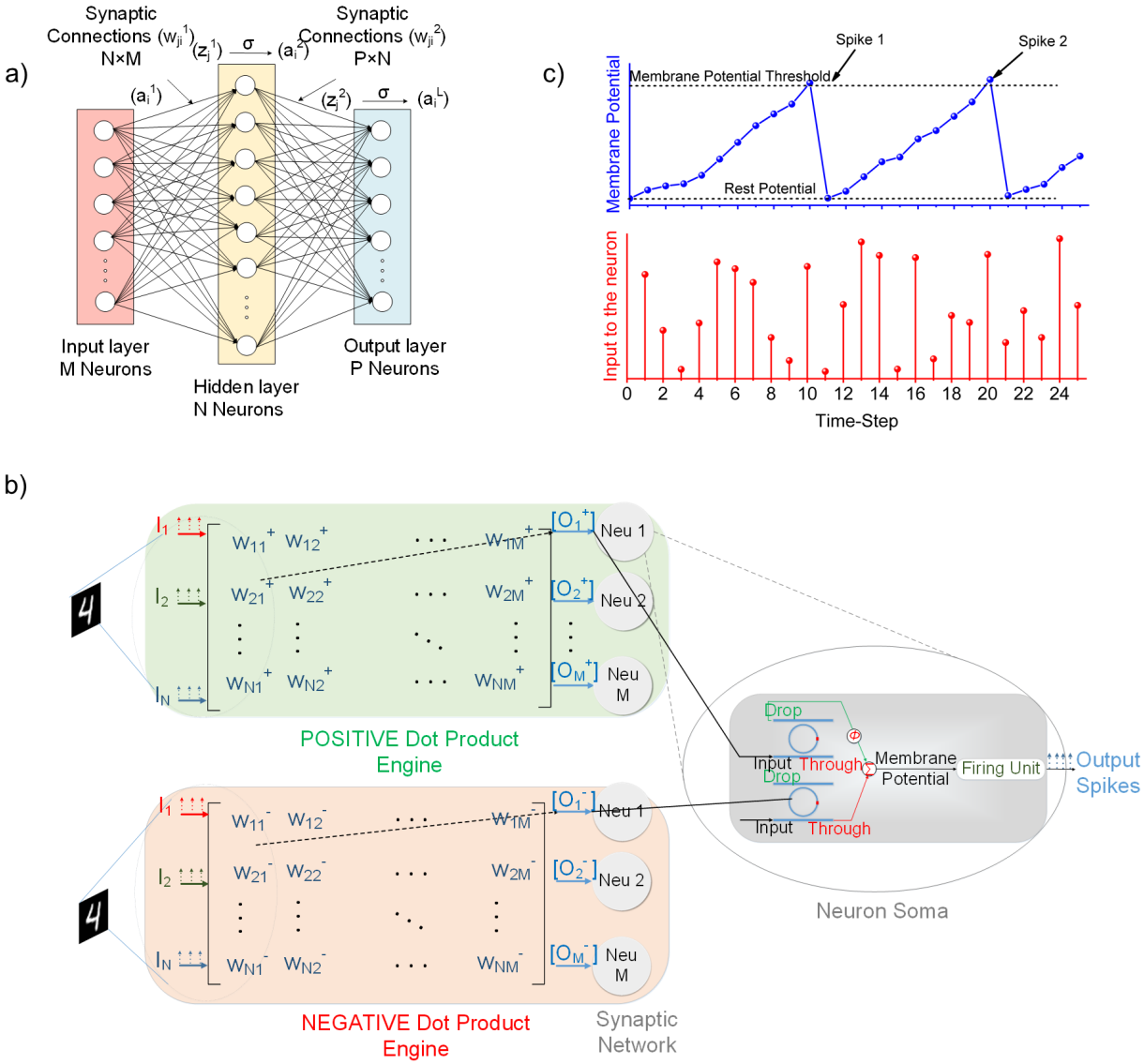


Figure 8.5. (a) Fully connected ANN topology showing 3 interconnected layers, namely, the input layer, the hidden layer and the output layer[22], (b) Schematic of potential integration of an integrate-and-fire neuron in a spiking neural network framework consisting of bipolar weights. The positive and negative weighted sums are computed using two separate dot-product engines and input to two different ring-resonators. The bidirectional integrating action of the two ports of the ring resonator is leveraged to calculate the effective membrane potential under the action of the bipolar weighted sums. Output spikes are generated when the effective membrane potential of the neuron crosses a threshold by the spike generation mechanism described. (c) The behavior of the proposed integrate-and-fire neuron in the simulated SNN showing the variation of the membrane potential under the action of incident pulses thus showing integrate and firing action.

the neurons with 'ReLU'[148] activation functions in the ANN are replaced by the proposed integrate-and-fire neurons. The dependence of final state of the device on the input and initial state of the device as shown in Fig. 8.3 (f) was used to determine the state of each neuron after each time-step. Then, the transmission characteristics of the ports of the ring resonators in the proposed neuron as shown in Fig. 8.4 (a) and (b) was used to determine the final membrane potential of each neuron. Each pixel of a 28×28 input image is divided into a stream of spikes whose frequency is proportional to the pixel intensity. The proposed integrate-and-fire neurons receive the dot product of the input spikes in a certain time-step t and the corresponding weights of synapses connecting the neuron and the inputs as shown in Fig. 8.5 (b). Upon receiving the dot product stimulus, the neurons integrate its membrane potential at that time-step. Mathematically, for j^{th} neuron, this can be represented similar to Eqn.6:

$$V_{\mathrm{j}}[t] = V_{\mathrm{j}}[t-1] + \sum_{\mathrm{i}} I_{\mathrm{i}}[t] w_{\mathrm{ij}} \tag{8.8}$$

where $V_{\mathrm{j}}[t]$ is the internal state or the membrane potential of the j^{th} neuron at time t, $I_{\mathrm{i}}[t]$ is the i^{th} input at time t, w_{ij} is the weight of the synapse connecting the i^{th} input to the j^{th} neuron. The details of the synaptic network implementation in the photonic domian will be a future course of study, however, similar concepts have been well-explored in the electrical domain [74]. Any synaptic network is essentially a dot-product engine performing element-wise multiplication of the inputs and the synaptic weights. Such a dot-product engine receives an N-dimensional input vector and provides an M-dimensional output vector which can be mathematically represented as:

$$\begin{bmatrix} O_1 \\ O_2 \\ \vdots \\ O_M \end{bmatrix} = \begin{bmatrix} I_1 & I_2 & \dots & I_N \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1M} \\ w_{21} & w_{22} & \dots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \dots & w_{NM} \end{bmatrix} \tag{8.9}$$

where $[w_{\mathrm{ij}}]$ is a $N \times M$ weight matrix.

To account for weights of either polarity, we represent the weights in two different dot-product engines as shown in Fig. 8.5 (b). We can interpret the weight w_{ij} to possess a positive and negative component:

$$w_{ij} = w_{ij}^{+} - w_{ij}^{-} \tag{8.10}$$

$$w_{ij}^{-} = |w_{ij}|, w_{ij}^{+} = 0, \text{when } w_{ij} < 0 \tag{8.11}$$

$$w_{ij}^{+} = w_{ij}, w_{ij}^{-} = 0, \text{when } w_{ij} > 0 \tag{8.12}$$

This gives us two matrices $W^{+} = [w_{ij}^{+}]$ and $W^{-} = [w_{ij}^{-}]$. These matrices are represented in the dot-product engines such that they return the corresponding dot products:

$$O_{j}^{+} = \sum_{i} I_{i} w_{ij}^{+} \tag{8.13}$$

$$O_{j}^{-} = \sum_{i} I_{i} w_{ij}^{-} \tag{8.14}$$

The positive and negative integrating ring resonators in the proposed neuron take these inputs separately and integrate in opposite direction such that the resulting integration mimics the desired integration that a biological neuron performs, given by Eqn. 7 because $\sum_{i} I_{i} w_{ij} = \sum_{i} I_{i} w_{ij}^{+} - \sum_{i} I_{i} w_{ij}^{-}$. The resulting membrane potential is fed to a Firing Unit as described in Fig. 8.2 (a). A behavorial model of the SNN inferencing framework described above was simulated using the MATLAB Deep Learning Toolbox [163] using a well-explored network topology [254]. Fig. 8.5 (c) shows the progression of the membrane potential of the proposed integrate-and-fire neuron in the hidden layer of the simulated SNN under the action of weighted incident spikes with time. The magnitude of the weighted incident spikes is essentially equal to $\sum_{i} I_{i}[t] w_{ij}$ for the j^{th} neuron at time-step t. It can be observed that once the membrane potential of the neuron reaches its threshold, it goes back to its rest potential. In the process, it generates a spike that gets fed to the next layer. The same integration process happens in case of the output layer neurons as well and the spike activities of all the neurons are monitored. The 10 output layer neurons correspond to the 10 classes of image being classified. The neuron with the highest spiking activity over a number of time-steps is

compared with the test image label and if it matches with the neuron number, the image is classified correctly. This device to system level analysis helps us validate the operation of the proposed integrate-and-fire neuron. The accuracy of recognition was calculated to be 98.06% after 25 time-steps on the testing set. The accuracy suffers a 0.24% degradation with respect to the testing accuracy (98.3%) of a SNN based on an ideal integrate-and-fire neurons. This can be attributed to the non-linear transmission characteristics shown in Fig. 8.4 and the dependence of the final state on the initial state of the device. Such device inaccuracies can be accounted for by modifying the training algorithm [22].

The important metrics for performance evaluation on a neuromorphic hardware system are energy efficiency and speed. To that effect, the energy and delay performance of the proposed neuron merits discussion. Each 'write' cycle is considered to be $1.5ns$ and each 'read' cycle for the proposed neuron was considered to be $500ps$. The durations of the 'read' and 'write' pulses were $200ps$. The additional times in the 'write' and 'read' cycles is to ensure that the GST temperature settles to its initial value after the excitation. The 'write' times are constrained by the transient response of GST to an amorphization pulse, which is shown to achieve times as low as $200ps$, experimentally [236] when excited with $1ps$ pulses. The average energy of a 'write' step considered for the simulation of the neural network was 4 pJ per neuron per time-step whereas the average 'read' energy was 1 pJ per neuron per time-step. The energy consumption in the 'write' cycles of the neuron can be further reduced by optimizing the feature size of the GST element. PCM devices of similar feature sizes[71], [255] in the electrical domain can consume upto 14-19 pJ of 'write' energy while operating at speeds of 40-100ns. Writing into the GST through evanscent coupling with photonic waveguides thus achieves a higher energy efficiency and speed, thus promising to rekindle the viability of PCMs for fast neuromorphic processing.

Discussion

Neuromorphic engineering has evolved heavily from its dawn as researchers have explored various kinds of technologies to mimic the functionality of the brain on an energy-efficient hardware platform. In the electrical domain, such technologies have been demonstrated to

possess limitations such as speed, energy, process integration etc. Phase change materials, in particular, have hit the scaling bottleneck where further improvements in energy-efficiency would require reducing 'write' speeds significantly. To beat CMOS in terms of energy-efficiency a 10× reduction[235] in current pulse amplitude or increase in pulse duration is necessary. As a solution, we propose an all-photonic integrate-and-fire neuron based on the phase change dynamics of GST which promises to achieve 'write' speeds of sub-ns orders. To the best of our knowledge, this is the first demonstration of a biologically plausible spiking neuron in the photonic domain involving phase change materials. We also showed that the proposed neuron can be potentially integrated with synapses in an all-photonic spiking neural network inferencing framework without any significant drop in classification performance. The proposed design opens up a host of possibilities for future implementations of all-photonic SNNs. By modulating the resonant wavelength by varying dimensions offers us the opportunity of wavelength multiplexing in an all-photonic spiking neural network. This offers substantial benefits such as elimination of cross-talk between neighboring neural elements thus allowing the provision of a denser network and in addition, could possibly allow us to implement larger networks on the same chip. With the recent advances in Photonic Neuromorphic, the proposed integrate-and-fire neuron fills the void of an all-photonic neuron that can be interfaced with photonic synapses[240] to build a truly integrated all-photonic neuromorphic system that leverages the aforementioned advantages of photonic devices to perform ultrafast neuromorphic computation.

8.4 Photonic In-Memory Computing Primitive for Spiking Neural Networks Using Phase-Change Materials

The core computational units of any neural network are neurons and synapses. In SNNs, information is encoded in form of spikes and the neurons and synapses are capable of processing information through these spike trains. As shown in Fig. 8.6 (a), the input trains of spikes get multiplied by the synaptic weights $w_1, w_2, ..., w_n$ and the weighted sum is received by an 'Integrate-and-Fire' neuron. The internal state of the neuron, known as the 'membrane potential' (V_{mem}) integrates based on the incoming weighted spikes and is compared with a threshold (V_{th}) at every time-step. The neuron outputs a spike once V_{mem} reaches

V_{th}. The synaptic functionality essentially corresponds to a multiplication operation of the inputs and the corresponding weights of the synapses. The basic operation performed by a single synapse can be represented as $I_{\mathrm{i}}w_{\mathrm{i}}$. We show how a single bus microring resonator with a GST element embedded on top of it can operate as such a synapse. The device under consideration is a Si-on-insulator structure consisting of a rectangular waveguide and a ring waveguide as shown in Fig. 8.6 (b). A GST element is deposited on one arm of the ring waveguide, which takes the shape of an arc and the length of the arc is denoted as the length of the GST element (L_{GST}). The fabrication technique of building such a structure has been well explored [236], [256]. Wave in the rectangular waveguide gets partially coupled to the ring and constructively interferes when the round-trip phase shift equals an integer multiple of 2π leading to the resonant condition:

$$2\pi R_{ring} n_{\mathrm{eff},wg} = m\lambda_m \tag{8.15}$$

where R_{ring} is the radius of the ring waveguide, $n_{\mathrm{eff},wg}$ is the effective refractive index of the ring waveguide and λ_m is the resonant wavelength. The transmission through the 'PASS' port is dependent on the device dimensions and material such that:

$$T_p = \frac{a^2 - 2arcos\theta + r^2}{1 - 2arcos\theta + a^2r^2} \tag{8.16}$$

where a is the attenuation factor and r is the self-coupling coefficient as shown in Fig. 8.6 (c). θ is the single-pass phase shift. Under resonance, θ equals 2π and the transmission is given by $T_{min} = ((a-r)/(1-ar))^2$.

We leverage the contrasting optical properties of GST in its amorphous (a-GST) and crystalline (c-GST) states to manipulate the attenuation in the ring waveguide and thus vary the transmission T_{min} at the resonance wavelength. The varying imaginary refractive indices of a-GST and c-GST leads to differential absorption of evanescently coupled light. The difference in optical absorption can be visibly observed through the cross-section view of the fundamental mode profiles in GST-embedded Si waveguide when excited by a TE mode electromagnetic (EM) wave as shown in Fig. 8.7. c-GST introduces a significant change in

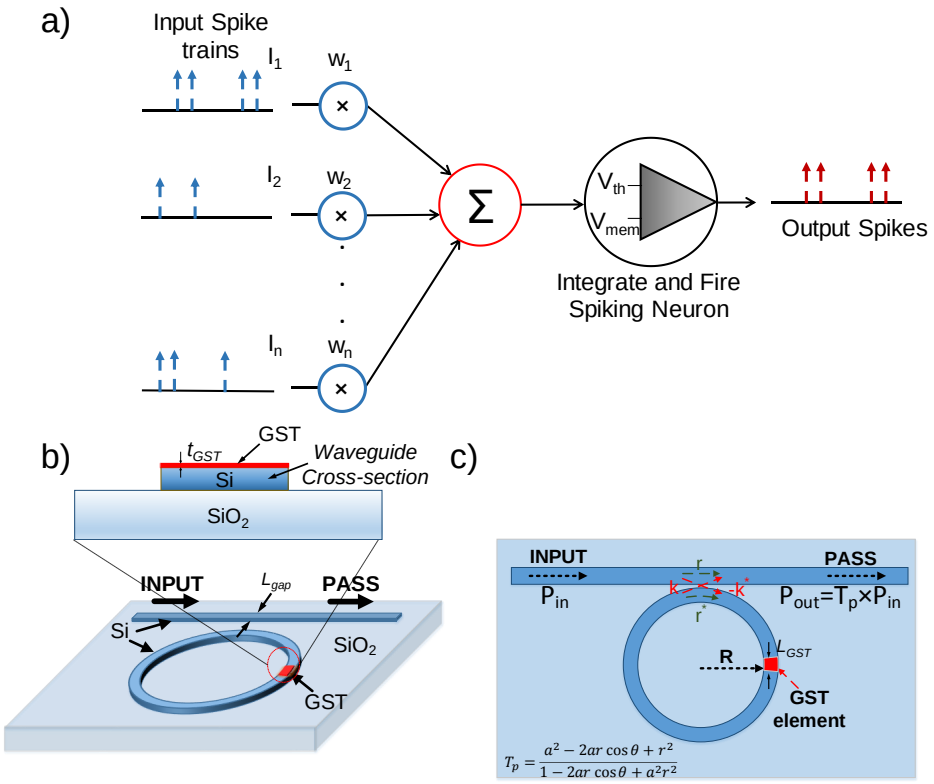


Figure 8.6. (a) The basic functional elements of an SNN are spiking neurons and weighted synaptic connections. At each time instant, the inputs are weighted by the synaptic weights to produce a resultant output represented as $\sum_i P_i w_i$. The 'integrate-and-fire' neuron's membrane potential (V_{mem}) is updated according to the weighted sum and compared with a threshold value (V_{th}). (b) GST-embedded single bus microring resonator structure with Si waveguides on SiO_2 substrate. (c) Top view of the device illustrating the different parameters pertaining to the ring resonator structure. The synaptic device performs an analog multiplication of input P_{in} and transmission T.

waveguide mode in contrast to a-GST due to higher absorption in the GST element. The attenuation factor (a) in Eqn. 2 can be related to the imaginary refractive index as:

$$a = exp(-\frac{2\pi\kappa_{eff,GST}L_{GST}}{\lambda} + Loss) \tag{8.17}$$

where $\kappa_{eff,GST}$ is the effective imaginary refractive index of the GST on Si-SiO_2 stack, L_{GST} is the length of the GST element, and the term '$Loss$' refers to other propagation losses such as bending losses, etc. The GST element can be programmed to partially crystallized levels

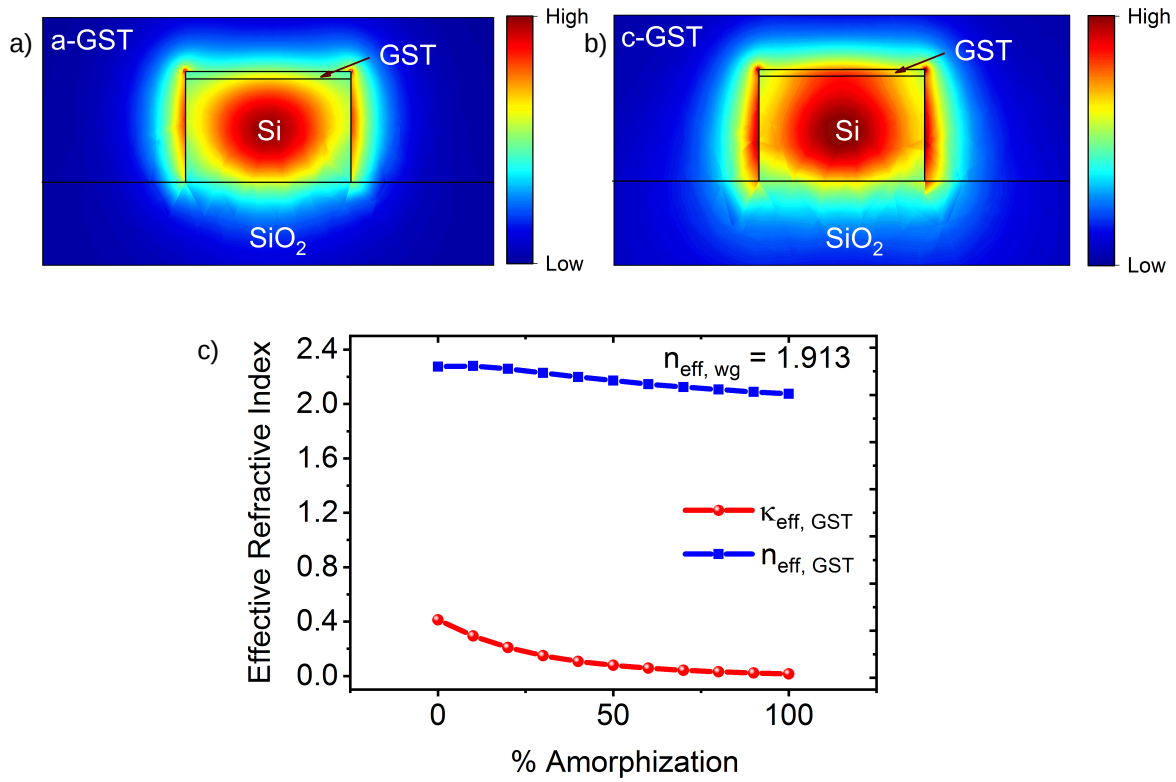


Figure 8.7. Cross-section view of Fundamental Mode profiles for a GST-embedded Si-SiO_2 waveguide section for (a) a-GST and (b) c-GST showing visible contrast in optical absorption for the two boundary states of GST. (c) The variation of the real ($n_{eff,GST}$)) and imaginary ($\kappa_{eff,GST}$) refractive indices of GST with degree of crystallization.

such that multi-level states can be achieved [238], [256]. To note, from the perspective of neural networks, significant progress have been made towards proposing training algorithms [257], [258] which preserve performance even with binarized synapses. Thus, although multi-level states would be desirable from a device point of view, modified training techniques can enable reasonable performance with low-precision synapses.

The refractive indices of partially crystallized GST can be calculated from effective permittivities approximated by an effective-medium theory [146], [242]:

$$\frac{\epsilon_{eff}(p)-1}{\epsilon_{eff}(p)+2} = p \times \frac{\epsilon_c - 1}{\epsilon_c + 2} + (1-p) \times \frac{\epsilon_a - 1}{\epsilon_a + 2} \tag{8.18}$$

where ϵ_c and ϵ_a are the complex permittivites of c-GST and a-GST respectively calculated from the refractive indices of GST[237] by $\sqrt{\epsilon(\lambda)} = n + i\kappa$. p is the degree of crystallization. Thus, the different levels of crystallization of GST leads to various levels of $\kappa_{eff,GST}$ thus leading to different levels of transmission. We leverage the multi-level transmission to implement an all-photonic synapse. Considering an incident optical pulse of power P_{in}, the synaptic functionality is realized such that the output power P_{out} is given by:

$$P_{out} = T_{\lambda_m} P_{in} \tag{8.19}$$

where T_{λ_m} is the transmission at resonant wavelength λ_m. T_{λ_m} represents the weight of the synapse and the various levels of transmission with varying degree of crystallization states of GST can be leveraged to represent a entire range of synaptic weights with appropriate discretization. We critically couple the resonator to the amorphous state such that the transmission is minimum in the amorphous state and increases with the degree of crystallization. While individual synapses represent a simple multiplication, the weighted inputs from multiple synapses are received by a neuron as shown in Fig. 8.6 (a). To emulate such a behavior, it is important to connect these synapses in an integrated fashion. Such a synaptic network would perform the most ubiquitous functionality of any neural network, a dot-product.

8.4.1 Photonic Dot Product Engine

We leverage the characteristics of the proposed non-volatile photonic synaptic device to map the synaptic weights of a neural network in a Photonic Synaptic Network capable of performing the dot-product of the inputs and the weights.

Network Design

We leverage the Wavelength Division Multiplexing (WDM) technique to compute dot product operations between incoming spikes and synaptic weights. We represent the synaptic weights in terms of the transmission T_λ of the microring resonator as discussed in the previous

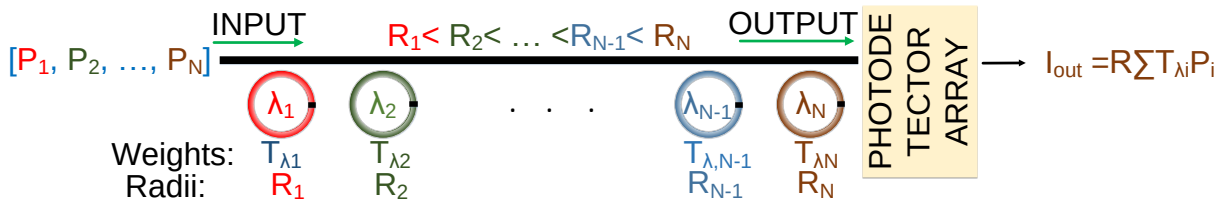


Figure 8.8. Synaptic dot product engine showing arrangement of ring resonators with increasing radii representing the transmission vector $T_\lambda = \{T_{\lambda_1}, \ldots, T_{\lambda_N}\}$. WDM signals gets modulated by weights corresponding to respective wavelength and the photodetector array collects the signals to generate a current I_{out} representing the dot product of transmission vector T_λ and inputs $P = \{P_1, \ldots, P_N\}$.

section. To represent multiple wavelengths, we use multiple ring resonators of increasing ring radii to represent different synapses in a row as shown in Fig. 8.8. The number of synapses (N) in each row is dependent on the Free Spectral Range (FSR) of the ring resonator and this governs the dimension of the input vector of the dot product engine. A WDM spike enters the straight waveguide through the 'INPUT' port and the GST element on each ring resonator modulates the amplitude of corresponding wavelength by the representative synaptic weight according to Eqn. (5). Thus at the 'OUTPUT' port we obtain a multi-wavelength spike comprising of different $T_{\lambda_i} P_i$ products corresponding to different wavelengths. This spike is then fed to a photodiode array (PD) which produces a current given by the sum of all the amplitudes given by:

$$I_{out} = R \sum_i T_{\lambda_i} P_i \tag{8.20}$$

where R is the responsivity of the PD expressed as A/W. This current is equal to the dot product of the input vector P and weight vector T_λ. The operation is illustrated in Fig. 8.8.

Synapse Design constraints

Using the WDM technique for the proposed photonic synaptic array imposes certain constraints on the design of the synaptic devices. For accurate dot-product operation, it is necessary to achieve significant isolation between the channels in order to minimize channel-to-channel interaction. The important parameters which constrain the design space of the

synaptic device are finesse (F) and channel spacing (λ_{diff}). Finesse is the ratio of free spectral range (FSR) and full-width at half maximum (FWHM). For a single bus ring resonator, FWHM and FSR are expressed as [259]:

$$FWHM = \frac{(1-ra)\lambda_m^2}{\pi n_g L\sqrt{ra}} \tag{8.21}$$

$$FSR = \frac{\lambda_m^2}{n_g L} \tag{8.22}$$

$$Finesse = \frac{FSR}{FWHM} \tag{8.23}$$

where $L = 2\pi R_{ring}$ is the circumference of the ring, n_g is the group index and rest of the parameters bear the same meaning as defined earlier. The interference due to adjacent channels can be modeled as:

$$\begin{aligned} T_{\lambda_i}|_{\lambda=\lambda_i} &= T_{\lambda_i}|_{\lambda=\lambda_i} \times T_{\lambda_i}|_{\lambda=\lambda_{i+1}} \times T_{\lambda_i}|_{\lambda=\lambda_{i-1}} \\ T_{\lambda_i}|_{\lambda=\lambda_i} &= \alpha_{\lambda_i} T_{\lambda_i}|_{\lambda=\lambda_i} \end{aligned} \tag{8.24}$$

Here, $T_{\lambda_i}|_{\lambda=\lambda_i}$ is the modified transmission due to interference from the adjacent resonant wavelengths, $T_{\lambda_i}|_{\lambda=\lambda_i,\lambda_{i+1},\lambda=\lambda_{i-1}}$ are the transmissions of i^{th} ring at the i^{th}, $(i+1)^{th}$ and $(i-1)^{th}$ resonant wavelengths respectively. α_{λ_i} represents the non-ideal factor which should ideally be close to 1. α_{λ_i} decreases with decreasing channel spacing (λ_{diff}) and increasing FWHM. For our design, we decided the minimum radius of the ring to be 1.5 μm in order to achieve a high density synaptic array for better scalability. Rings of similar size have been demonstrated previously [260] with certain modifications that we will discuss next. The rest of the parameters concerning the synapses were chosen to maximize the number of rings in a single row (N) while maintaining α_{λ_i} close to 1 under the condition that $N\lambda_{diff} < FSR$.

A number of challenges arise for rings of radius comparable to the wavelength of operation. Firstly, to achieve a critical coupling in the low-loss amorphous state, the power coupling gap between the bus and the ring waveguide needs to be small ($< 100nm$). This is because the interaction length between the ring and the straight waveguide is quite short and hence to achieve reasonable coupling, even to match the small intrinsic loss in the ring in low-loss amorphous state of GST, we require a small power coupling gap. Such gaps become

extremely difficult to fabricate. An alternative to using lower gaps has been demonstrated [260] for rings of small radii. Reducing the width of the bus waveguide increases the spatial period of the propagating mode due to the lower effective refractive index. This results in a better phase match with the mode in the tightly curved ring waveguide. For the rest of our analysis, we have used a bus waveguide of width 0.35 μm and a coupling gap of 135 nm.

8.4.2 Operation of All-Photonic Spiking Neural Network

Implementation of a SNN based on the Photonic Dot-Product Engine (PDPE) and 'integrate-and-fire' neurons described above involves integration of the proposed structures. As elucidated above, the basic computational function of a neural network is a dot product. To realize parallel instances of such a functionality using the aforementioned PDPE, we use a splitter (SPL) to feed the WDM input spikes to multiple PDPE rows with the input vector and obtain the dot-products of each rows from respective PD arrays as shown in Fig. 8.9. Essentially, the output vector thus obtained from the PD arrays gives us the multiplication of the vector of input spikes P_{i} with a $N \times M$ synaptic network T_{ij}. The M outputs I_{j} obtained from the PD arrays are fed to laser diodes (LD) which converts the electrical current to optical spikes thus completing the parallel dot-product operations and can be represented as:

$$\begin{bmatrix} O_1 \\ O_2 \\ \vdots \\ O_M \end{bmatrix} \propto \begin{bmatrix} P_1 & P_2 & \dots & P_N \end{bmatrix} \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1M} \\ T_{21} & T_{22} & \dots & T_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \dots & T_{NM} \end{bmatrix} \tag{8.25}$$

We now present how such a photonic synaptic network based can be integrated with the proposed bipolar IF Neurons to realize a photonic SNN. The schematic of such a photonic SNN is illustrated in Fig. 8.10. To account for negative weights in a neural network, we

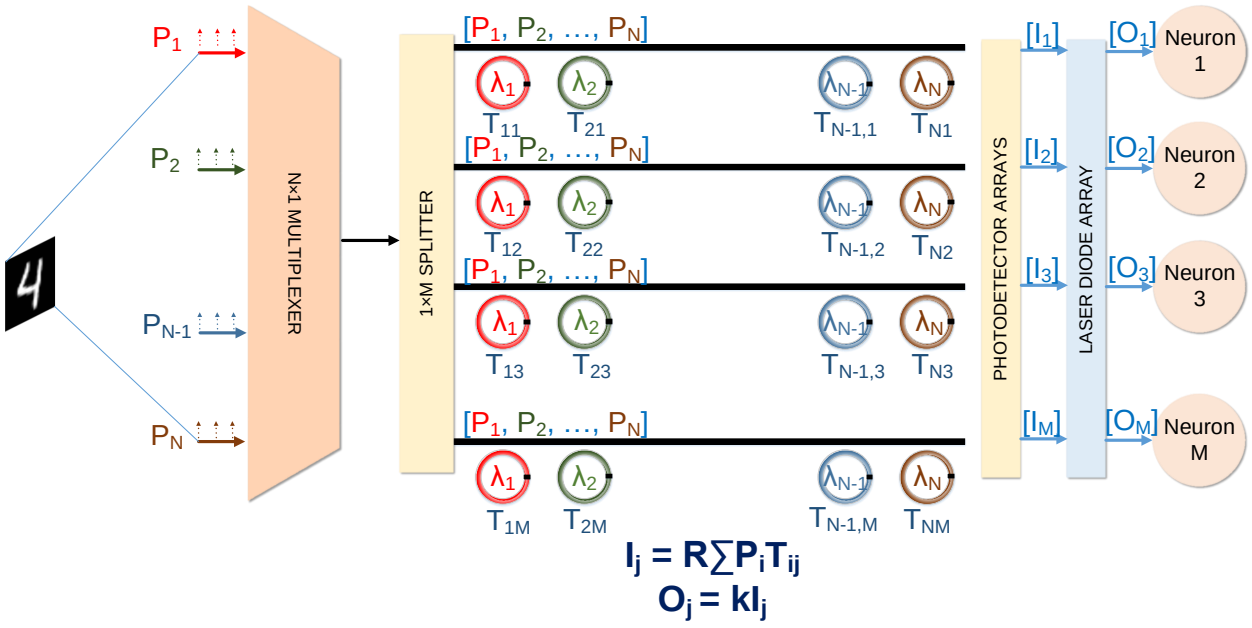


Figure 8.9. Synaptic dot product engine showing arrangement of ring resonators with increasing radii representing the transmission vector $T_\lambda = \{T_{\lambda_1}, \ldots, T_{\lambda_N}\}$. WDM signals gets modulated by weights corresponding to respective wavelength and the photodetector array collects the signals to generate a current I_{out} representing the dot product of transmission vector T_λ and inputs $P = \{P_1, \ldots, P_N\}$. k is an amplification factor.

represent the element of the weight matrix T to be comprised of a positive and negative component:

$$\begin{aligned} T_{\mathrm{ij}} &= T_{\mathrm{ij}}^{+} + T_{\mathrm{ij}}^{-} \\ T_{\mathrm{ij}}^{+} &= T_{\mathrm{ij}}, T_{\mathrm{ij}}^{-} = T_{low}, \text{when } T_{\mathrm{ij}} > 0 \\ T_{\mathrm{ij}}^{+} &= T_{low}, T_{\mathrm{ij}}^{-} = |T_{\mathrm{ij}}|, \text{when } T_{\mathrm{ij}} < 0 \end{aligned} \tag{8.26}$$

Here T_{low} is the transmission corresponding to the lowest programmable state considered. Two PDPE arrays are deployed for mapping the positive and negative components respectively as depicted in Fig. 8.10. The dot-product outputs from the LD arrays of the two DPE arrays can be represented as:

$$\begin{aligned} O_{\mathrm{j}}^{+} &= \sum_{\mathrm{i}} P_{\mathrm{i}} T_{\mathrm{ij}}^{+} \\ O_{\mathrm{j}}^{-} &= \sum_{\mathrm{i}} P_{\mathrm{i}} T_{\mathrm{ij}}^{-} \end{aligned} \tag{8.27}$$

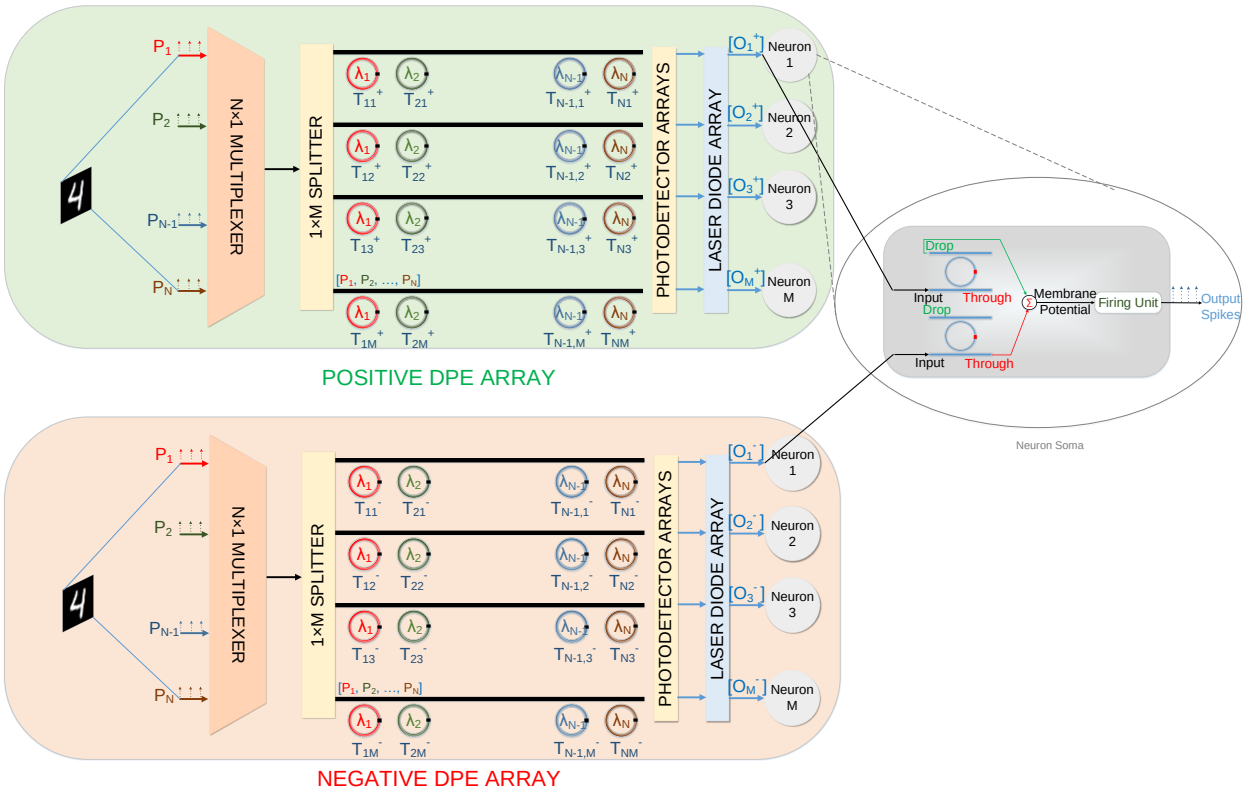


Figure 8.10. Schematic of an All-Photonic Spiking Neural Network. Two DPE arrays are deployed to represent the positive and negative components of the weights. The outputs of the DPE arrays are converted to optical spikes and passed to integrate-and-fire neurons. The structure of an integrate-and-fire neuron is illustrated in a circle. Each neuron has two inputs corresponding outputs from the positive and negative DPE arrays. The neuron outputs a spike when the membrane potential crosses its threshold.

These outputs from the jth rows are received by the jth IF neuron discussed earlier. The outputs from the positive and negative PDPE arrays are received by the positive and negative integrating ring resonators in the neuron respectively. The two ring resonators integrate in the opposite direction based on the two inputs and the resulting integration mimics the desired integration that a biological 'integrate-and-fire' neuron performs, given by:

$$V_{mem,\mathrm{j}}[t] = V_{mem,\mathrm{j}}[t-1] + \sum_{\mathrm{i}} P_{\mathrm{i}} T_{\mathrm{ij}} \tag{8.28}$$

Here, $\sum_{\mathrm{i}} P_{\mathrm{i}} T_{\mathrm{ij}} = \sum_{\mathrm{i}} (P_{\mathrm{i}} T_{\mathrm{ij}}^{+} - P_{\mathrm{i}} T_{\mathrm{ij}}^{-})$. $V_{mem,\mathrm{j}}[t]$ is the internal state or the membrane potential

Table 8.2. Simulation Parameters

Parameters	Values
Si Ring Waveguide X-Section	0.45×0.25 μm^2
Si Bus Waveguide X-Section	0.35×0.25 μm^2
Coupling Gap (L_{gap})	0.135 μm
GST Length (L_{GST})	170 nm - 220nm
GST Thickness (t_{GST})	10 nm
GST Width (W_{GST})	0.44 μm
Si Refractive Index (n_{Si}) [245]	3.5
SiO_2 Refractive Index (n_{SiO_2}) [246]	1.4
c-GST Refractive Index ($n_{c-GST} + i\kappa_{c-GST}$) [247]	7.2+1.9i
a-GST Refractive Index ($n_{a-GST} + i\kappa_{a-GST}$) [247]	4.6+0.18i

of the j^{th} neuron at time t. The resulting membrane potential is passed to a Firing Unit as described in Fig. 8.2 such that the neuron produces an output spike once the $V_{mem,j}[t]$ reaches a threshold. The output spikes from all the neurons of the current layer are then fed to the next synaptic array layer. Fig. 8.10 delineates the operation of basic building blocks of a neural network. We perform large scale system-level simulations by emulating the behavorial model of the proposed spike processing system to assess the performance of neuromorphic systems based on this fabric.

It is important to consider the architecture-level facets of any computing primitive. The proposed design is analogous to memristive crossbars, where the high fan-in into the neurons is resolved by the inherent parallelism of the computing framework. In our design, each neuron receives two inputs, from the positive and negative synaptic array, and the output of that neuron is fed to one of the 16 inputs of the synaptic array of the next layer. In reality, neural networks are of far bigger sizes than what the proposed design can accommodate. As a result, multiple instances of the proposed primitive can be used with time-multiplexing to perform the entire vector-matrix multiplication operation. The partial sums from these instances are collected and added before being fed to the neuron. Output from a neuron is again served as inputs to the synaptic arrays storing the weights of the next layer of the neural network. Similar architectures have been explored using memristive technologies

[16], [261]. This work is concerned with device and circuit primitive of a spike-based photonic non-volatile inferencing engine which will act as a computing core of a large-scale system similar to technologies in the electrical domain.

8.4.3 Results

Device Simulations

We evaluated the performance of the proposed all-photonic SNN fabric by designing a device-circuit-algorithm co-simulation framework. First, the device characteristics of each ring resonator in a DPE row is simulated for 4 different degrees of crystallization of the GST element using commercial-grade simulator Lumerical FDTD Solutions[251] based on the finite-difference time-domain (FDTD) method. The fixed parameters used for these simulations are listed in Table 8.2. The mode-profiles were obtained through Electromagnetic simulations using the Finite Element method in COMSOL Multiphysics [250].

Device to System Framework

The device characteristics, obtained from the FDTD simulations are analyzed and a Gaussian fit is applied on the data for interpolation. We develop a device to system co-design framework by building behavorial models of the proposed synapses and neurons based on the fitted device characteristics. The models are used to evaluate the inferencing performance of the standard neural network topology on standard digit recognition task based on the MNIST dataset using the Deep Learning Toolbox[163] in MATLAB. The MNIST dataset consists of 60000 images in the training set and 10000 images in the testing set.

8.4.4 Device Simulations

We considered 16 ring resonators of radii linearly increasing from 1.5 μm to 1.59 μm in any particular DPE row. The choice of number of devices, N, in a single row is discussed earlier. The length of the GST element is increased accordingly and chosen iteratively to ensure uniform transmission characteristics across the wavelength range of operation. We

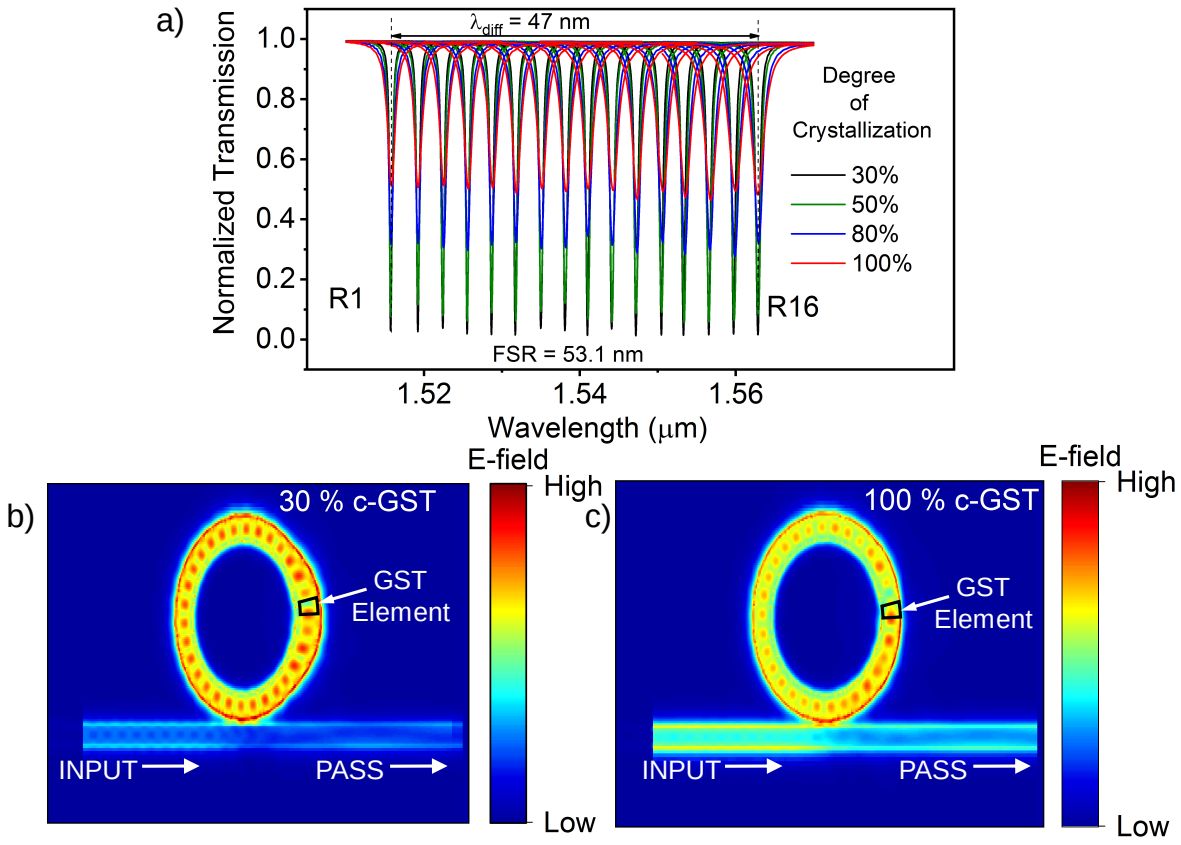


Figure 8.11. (a) Normalized transmission for 16 different rings for 4 degrees of crystallization (30 %, 50 %, 80 %, 100 %) showing a decreasing trend with decreasing degree of crystallization. The range of wavelength for the 16 rings is less than the FSR for the design. (b) and (c) shows the electric field profile in the ring resonator system showing visible contrast in optical absorption and field transmission at the 'PASS' port in the GST element for c-GST and 30% c-GST respectively.

performed FDTD simulations for each device with 4 different degrees of crystallization of GST (30%, 50%, 80%, 100%) and the observed transmission characteristics for the rings are shown in Fig. 8.11 (a). Expectedly, the transmission for each device decreases with decreasing degree of crystallization. The observed FSR was 53.1 nm and difference between the highest and lowest resonant wavelength was 47nm, which is well within the FSR, thus ensuring no interference from resonant wavelengths beyond the region of operation. Fig. 8.11 (b) and (c) show the contrast in electric field absorption by the GST element in the ring resonator for 30% and 100% crystallized GST. We observe certain variations across different

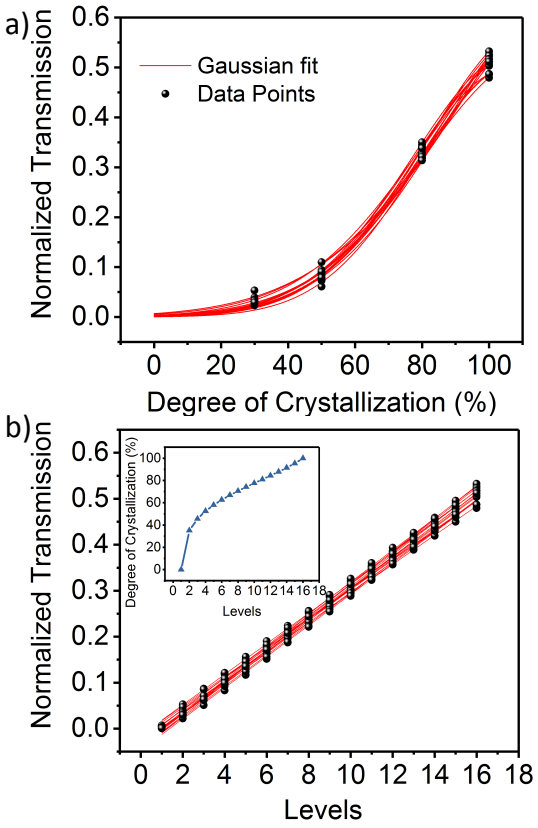


Figure 8.12. (a) Gaussian fit of simulated data points across degrees of crystallization ranging from 0 % and 100 %. (b) Linearly varying transmission across 16 different programmable states (Levels) of the GST. Inset shows the degrees of crystallization corresponding to the Levels.

wavelengths which can be minimized by further adjustments of lengths of the GST element. However, from the perspective of neuromorphic applications, these variations prove to be insignificant. We will explore the impact of such variations in our evaluation of the proposed neuromorphic processing engine. We exploit the dependence of transmission on degree of crystallization to realize the synaptic behavior of the rings. Fig. 8.12 (a) shows the Gaussian fit of the simulated data across degrees of crystallization varying from 0% to 100%. Note, the Gaussian fit provides a fairly accurate representation of the observed data and is a powerful tool to speed up our analysis in light of the computationally expensive FDTD simulations. It can be observed that transmission has a non-linear relationship with p and hence, operation of the rings as synapses would require the GST element to be programmed to states with non-

linearly increasing p. This can be achieved with appropriate amplitude of the programming stimulus. Fig. 8.12 (b) shows the transmission levels for each ring corresponding to 16 discretized programmable states or Levels. The degrees of crystallization, p, for each state is shown in the inset of Fig. 8.12 (b). The linear relationship between transmission and Levels is a necessity for the target application, i.e., a dot-product operation for neuromorphic computing which led us to the choice of programmable states with the non-linear distribution of p.

Interference Errors

The transmission characteristics of the different rings for varying states of the GST element is used to evaluate the accuracy of the dot-product operation performed using the proposed synaptic network. The error in the computation stems from the premise of overlapping frequency response between adjacent channels. The advantage of the proposed implementation over electrical counterparts is that in the electrical domain, the losses due to line resistance is a function of input and the weights thus rendering them difficult to model. The impact of the error in this setup is only dependent on the weight level and hence, can be easily modeled, analyzed and even corrected in light of the proposed application. In Eqn. 9, we have formulated a behavorial model of the error arising from interference due to adjacent channels. Fig. 8.13 shows the map of non-ideality factor α_{λ_i} for all 16 rings for 16 different levels. This was calculated through fitting of the extracted α_{λ_i} from Fig. 8.11 (a) based on Eqn. 9. We observe that errors are highest for rings of higher radius and for the highest levels. This can be attributed to higher FWHM for rings of higher radius due to the longer lengths of the GST element used to achieve uniform transmission levels across the operating range of wavelength. We include these error characteristics corresponding to each ring for our system level evaluation of the proposed photonic SNN inferencing framework.

System Level SNN performance

We develop a device to algorithm level framework to perform system level analysis of the photonic SNN implementation. A SNN, like any other neural network, consists of multiple

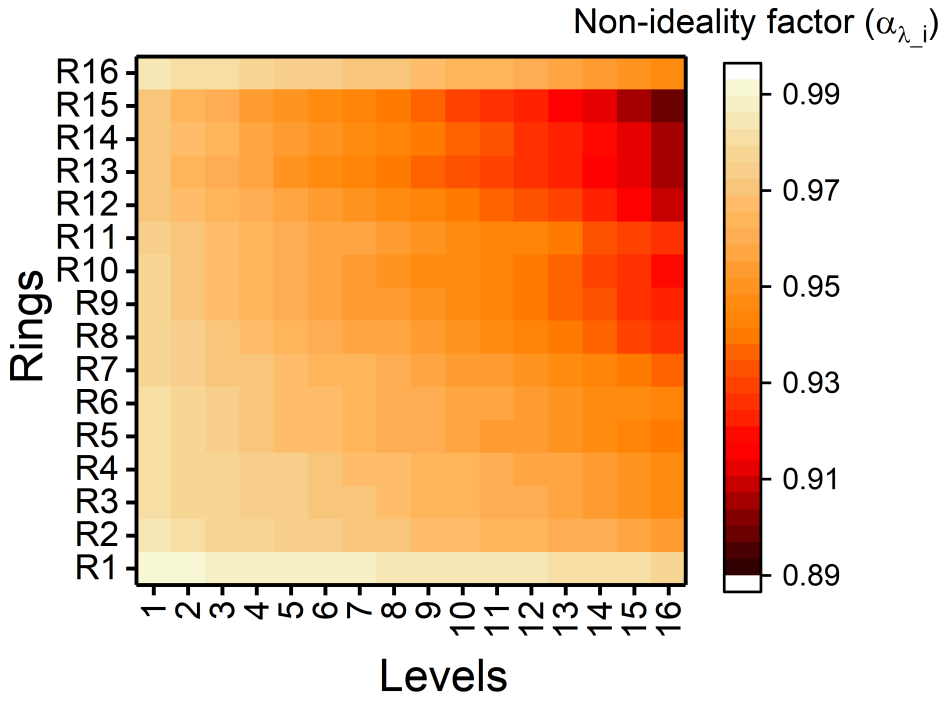


Figure 8.13. Map of non-ideality factor (α_{λ_i}) arising due to interference from adjacent rings for each ring in the DPE row.

layers of neurons connected through synapses. The unique property of SNNs is that the inputs to the network are discretized spike events instead of analog values. The synapses act as weights which get multiplied with amplitude of the incoming stimulus and the resulting weighted-sum, i.e., dot-product of all impulses coming from different synapses is received by the neuron. We map the device characteristics of each individual synapse and 'integrate-and-fire' spiking neurons discussed previously to explore the validity of operation of the proposed devices as synapses and neurons in such a SNN. Let us now explain how we perform the evaluation of a SNN on the proposed PCM-based photonic inferencing framework. We consider a fully connected neural network consisting of 3 layers, namely, the input layer, the hidden layer and output layer as shown in Fig. 8.14 (a). This type of topology is well explored [254]. For our analysis, we consider a network with $M = 784$, $N = 500$, $P = 10$. We analyze the accuracy of such a network in a standard handwritten digit recognition task based on the MNIST dataset [252]. A popular way of implementing spike-based inferencing systems is to train a network as an Artificial Neural Network (ANN) and then convert it to a SNN by well

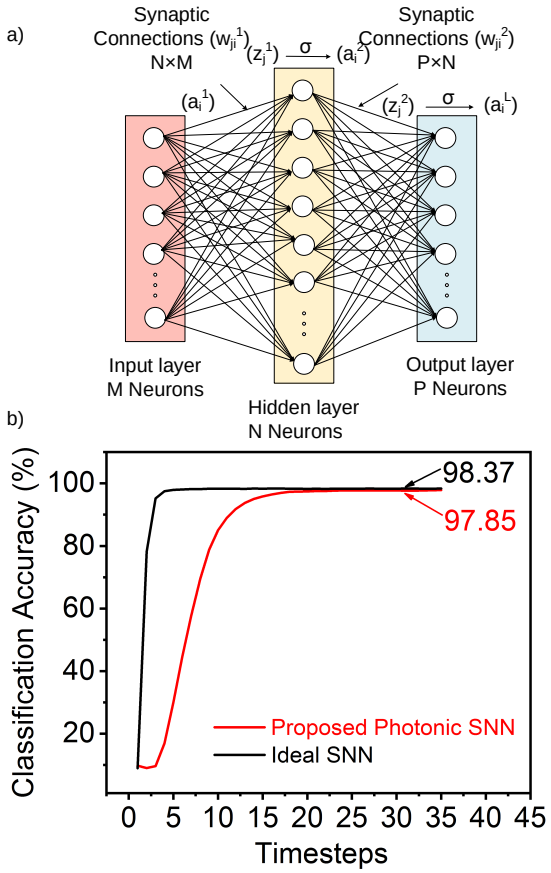


Figure 8.14. (a) Fully connected neural network topology consisting of an input layer (M), a hidden layer (N) and an output layer (P) of neurons. The resulting synaptic networks are of sizes $N \times M$ and $P \times N$(b) Evolution of classification accuracy of handwritten digit recognition task based of MNIST dataset comparing our proposed Photonic SNN to ideal SNN performance. Here ideal SNN corresponds to software-level functionalities without considering device characteristics.

explored conversion algorithms [254], [262]. The weights of the network are trained using the Backpropagation algorithm [263] as in case of Artificial Neural Networks (ANN). The neurons in ANNs are usually non-linear mathematical functions, such as Rectified Linear Units (ReLU) [148], sigmoid or tanh with ReLU being the most popularly chosen neuron functionality. During conversion, an artificial neuron with ReLU functionality can be directly converted to an IF neuron, mathematically [254]. The details of the operation of the IF neuron has been elucidated in our earlier work[26]. The trained weights of the network after

the ANN is converted to a SNN are mapped to the observed characteristics of each synaptic device in the proposed synaptic network. The synaptic network has the provision of operating 16 synapses simultaneously. To perform the dot-product of larger dimensions, the synaptic network needs to be time-multiplexed as discussed earlier. To simulate large-dimension operations with the proposed synaptic network, we repeat the device characteristics every 16 synapses. The weights of the network can be negative. To account for negative weights, two dot-product engines are deployed, shown in Fig. 8.10 as described earlier.

The pixels of input images of size 28×28 are divided into streams of spikes whose frequency is proportional to the pixel intensity. At every time-step, the input can either be '0' when there is no spike or '1' in the event of a spike. The behavorial model of the SNN inferencing framework described above was implemented using the MATLAB Deep Learning Toolbox [163] using the network topology shown in Fig. 8.14 (a). The network is evaluated at every time-step by passing the inputs through the forward path from the input layer to the output layer through the synaptic network and activity of the network was recorded. Finally, the output neuron with the highest spiking activity is compared with the label of the input image to determine the accuracy of the recognition system. The classification performance of the proposed photonic SNN is compared with an ideal SNN in Fig. 8.14 (b). Here, ideal SNN essentially means software-level evaluation without taking device characteristics into consideration. We observe that there is a degradation in accuracy of 0.52 % after 35 time-steps from the ideal case arising from the different variations in device characteristics discussed earlier. To note, the concept of time-steps here correspond to how many times we evaluate the network over the Poisson-distributed input spikes generated from the image. The duration of a time-step is not relevant in this context as we do not include any temporal dynamics in the system. We further attempted to isolate the contribution of synaptic device variations to the observed degradation in accuracy by considering a comparison test case: ideal synapses with proposed neurons. That accuracy degradation amounted to 0.1% after 35 time-steps. This implies 0.42% degradation due to synaptic variations.

We evaluated the energy consumption of the the basic building blocks for our system, the synaptic array and the neurons. The energy consumed by each synapse can be estimated by the transmission (or the weight) of the synaptic device. As the information being processed

is based on spike events, the input can either be '1' or a '0'. Experimental demonstrations [238] have shown that readout for GST-based Si photonic devices can be achieved by pulse energies of 0.48 pJ. For our case, due to smaller GST footprints, we consider input '1' to correspond to a pulse of amplitude 0.25 mW. The power consumed by the synapse is thus given by (1-T) mW where T is the transmission of the synapse. As these read pulses will eventually write into the neurons, we choose a pulsewidth of 200 ps, which is the minimum pulsewidth required to write into the GST, as we observed previously [26]. Considering these metrics for the read pulses and power calculations for each synapse, we estimated the energy consumption of the entire classification operation described above. The resulting average energy consumption for first layer of the neural network in the synaptic array was calculated to be $\sim 12.5fJ$ per synapse per time-step of evaluation. For the second layer, the energy consumption was $\sim 1.6fJ$ per synapse per time-step. The difference is energy consumption in the two layers is due to more sparse spiking activity in the second layer. The energy consumed by each neuron was calculated in our previous work to be $5pJ$ per time-step. The writing energies for PCM devices of similar feature sizes [71], [255] in the electrical domain can amount upto 14-19 pJ while operating at speeds of 40-100ns. The total energy consumption for an image classification was calculated $\sim 261nJ$ ($178nJ$ consumed by the synaptic operations and $83nJ$ consumed by the neurons). Although the energy consumption is comparable to CMOS technology [264], photonics potentially offers a faster operation at sub-ns speeds. To note, in this work, we have considered a significantly high read pulse (0.25 mW) through the synapses which is reflected in the high energy per inference operation. The proposed synapses can be potentially read with a pulse of lower amplitude based on the sensitivity of the photodetectors and that will significantly improve the energy requirements of the system. Moreover, the speed of operation in the photonic domain is significantly higher since read latencies of the neuromorphic systems based on memristors usually occur in orders of ns. These benefits encouraged us to further explore the possibility of neuromorphic hardware design based on this technology.

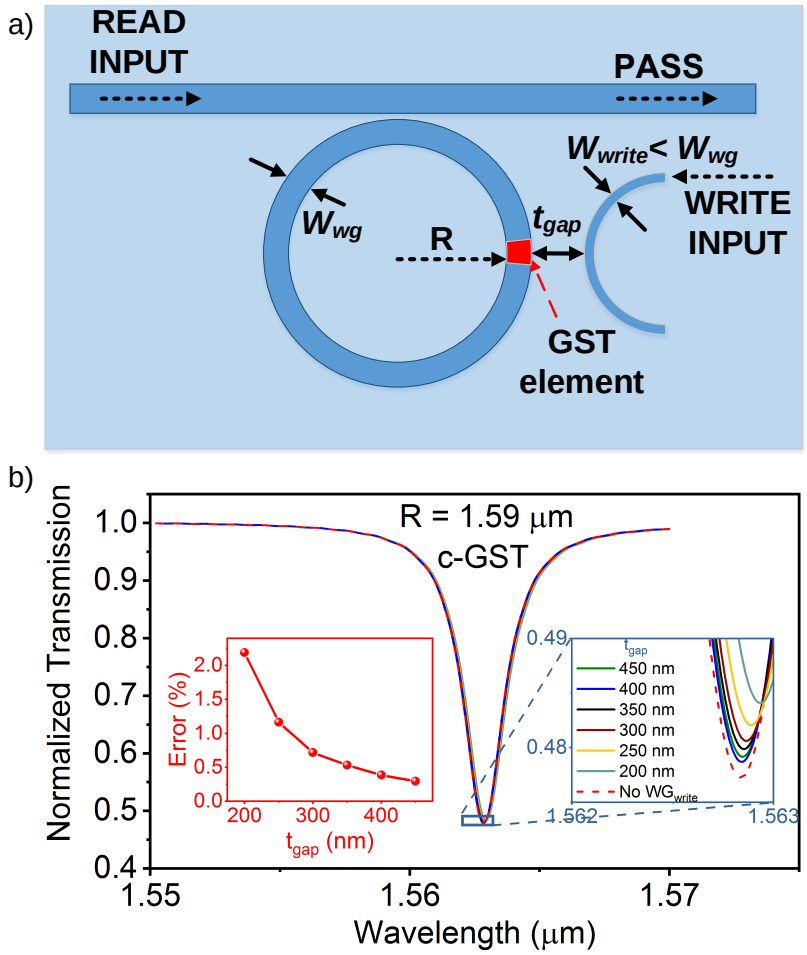


Figure 8.15. (a) Structure and arangement of input write waveguide at a distance t_{gap} to the synaptic device. The width of the write waveguide (W_{write}) is smaller than that of the ring waveguide (W_{wg}) for asymmetric coupling. (b) Transmission characteristics of 1.59 μm ring for different values of t_{gap} compared with the case without a write waveguide. Inset 1 (Blue) shows a zoomed-in view of the transmission characteristics to show the different cases clearly. Inset 2 (Red) shows the variation of percentage error in transmission at read wavelength 1562.85 nm with t_{gap}.

8.4.5 Discussion

The proposed photonic SNN inferencing framework fills a major void of scaling from device to systems in current state-of-the-art photonic neuromorphic works based on PCMs. However, few challenges stand in the way of physical demonstration of the proposal that need to be overcome. Firstly, reconfigurability of the proposed non-volatile synaptic array

is a necessity. Various reconfigurability schemes have been explored on the phase-change based photonic platforms [237], [256]. We explored the possibility of adding an input bend waveguide (WG_{write}) as a writing port for each synapse at a distance such that the inferencing framework is unaffected. The width of WG_{write} (W_{write}) is intentionally considered to be much lower than the ring waveguide of the synaptic device. This is done to achieve asymmetric coupling such that during writing, the wave leaks out of WG_{write} appropriately for efficient writing while during standard inferencing operation, the wave remains mostly confined within the ring. Fig. 8.15 (a) shows the structure and arrangement of WG_{write} adjacent to the proposed synaptic device. t_{gap} denotes the distance between the ring waveguide and WG_{write}. We observe that error in transmission during normal inferencing operation due to the presence of the WG_{write} is around 0.5 % for $t_{gap} \sim 300nm$. For the same distance, we calculated the transient field coupling from the WG_{write} to the ring to be 70 %. Thus, this writing scheme is a viable option for achieving reconfigurability in the proposed network.

The dimensions chosen for our analysis are catered towards achieving desirable functionality for ring resonators of small radii of around $\sim 1.5\mu m$. The main motivation behind using small ring resonators was to achieve high area density for scalability. We have explored a number of challenges arising from such small rings such as non-uniform bending and coupling losses across the range of wavelength and fabrication difficulties to achieve critical coupling. We have attempted to mitigate such challenges by appropriate design. Further, we delineated the design constraints for scaling individual synapses to a network of synapses which is necessary for large-scale neuromorphic systems. GST-based photonic platforms also experience a small resonance shift between the different programmable states of the PCM. The resonance shift between the any two states can be quantified by [236]:

$$\frac{\Delta\lambda_m}{\lambda_{m,in}} = \frac{\Delta n_{eff,GST}}{n_{g,eff}} \cdot \frac{L_{GST}}{2\pi R_{ring}} \tag{8.29}$$

Here, $\lambda_{m,in}$ is the resonant wavelength in the initial state, $\Delta n_{eff,GST}$ is the difference in effective refractive index between the states, $n_{g,eff}$ is the group index. For our case, it amounts to approximately 0.012 nm. In addition to the variations arising from device characteristics, we also explored errors arising due to interference from adjacent channels and their impact

on the performance of the proposed photonic SNN. From our analysis, it can be observed that the network size, N considered in our synaptic fabric is a rather conservative design. N can be further increased which would result in higher errors. However, the effect of such variations have been modelled in Eqn (9) and the resulting accuracy degradation can be recovered by modifying the training algorithm as explored for memristive technologies [22].

The challenges of errors arising due to interference between adjacent rings essentially stems from the usage of WDM-based computation. To that effect, the limitations of array size due to WDM merits discussion. WDM, while introducing parallelism in the system, is constrained by the finesse of the rings. In this work, we have shown that we can use 16 rings in a single dot-product engine row which implies that the array can process 16 inputs in parallel. The size of the array is thus limited to $16 \times N$ where N would be limited by the area and not design constraints. However, analogous computing units in the electrical domain using memristive crossbars are also limited in size due to electro-migration limits, sneak-paths and line-resistances. The photonic array on the other hand, although limited in one direction due to finesse, can be possibly extended to larger sizes in the direction of N. Moreover, time multiplexing is a popular practice when implementing large scale neural networks on memristive networks, as alluded to earlier. The possibility of fast writing into PCMs can potentially make these photonic arrays more suitable for temporally scalable architectures.

An alternative way to implement Photonic Neural Networks is through the use of inteferometers [232] where the weights of the network are controlled through phase-shifters. Such phase-shifters can consume significant amount of power per synapse to maintain the weight. On the other hand, non-volatile elements based on PCMs can potentially encode the weights without requiring any power to maintain their states. However, we do not use the concept of phase-shift for our design. We encode the weights in terms of levels of partial crystallization. Non-volatility is necessary for large-scale neuromorphic systems for primarily two reasons: i) it eliminates the need for phase-shifters as constant tuning is not required, and ii) it provides a platform for in-memory computing rather than storing the synaptic weights in a separate memory. In this work, the intention to use non-volatile material based memory primitive is to eliminate the need for thermal tuners. To the best of our knowledge, this is the first

proposal of photonic neuromorphic platform from a scalable system point of view based on a non-volatile memory primitive. Recent proposals [265], [266] have looked at scalable systems to realize complex neural dynamics using for dynamic learning. However, the flux-based memory in such systems are dependent on temperature and also on the run-time of operation. Such detailed neuro-biological functionalities make them more suitable for brain-like simulations similar to NeuroGrid [267] in the electrical domain. In this work, we do not incorporate complex biological dynamics of SNNs in our system and rather focus on leveraging the inherent sparsity of spike-based processing while performing image classification for energy efficiency. The primary motivation behind exploring this primitive stems from building a potentially reconfigurable neuromorphic system which performs energy-efficient inferencing. For building such neuromorphic platforms to perform spike-based processing in standard architectures, in-memory computing offers significant promise. To that effect, non-volatile memory primitives are quintessential and more suitable as they potentially eliminate the need for off-chip DRAM accesses, thus alleviating memory bottlenecks.

A popular way of implementing such spike-based inferencing systems is to train a network as an Artificial Neural Network (ANN) and then convert it to a Spiking Neural Network (SNN) by well explored conversion algorithms[254]. This method has seen considerable success [262] in image classification, far beyond the scope of spike-based training algorithms. The neurons in ANNs are usually non-linear mathematical functions, such as Rectified Linear Units (ReLU), sigmoid or tanh with ReLU being the most popularly chosen neuron functionality. During conversion, an artificial neuron with ReLU functionality can be directly converted to an IF neuron, mathematically [262]. This explains why we have chosen IF neuron as the spiking neuron in our proposal. IF neurons are not associated with time-constants as it does not include leak factors and the operations are fairly simple unlike other spiking neurons. The proposal concerns with building spike-based photonic neuromorphic inferencing platform for image classification task. Note, the neuron does not bear exact resemblance to biological neuron, however, the design leverages the event-driven behavior of biological neurons. The aim of this work is to build a fast neuromorphic inferencing platform in the spiking domain to perform machine learning tasks such as image classification. Sev-

eral works [267] have previously explored brain-like neuron and synaptic functionalities with more significant resemblance for complex neural simulations, albeit in the electrical domain.

The major advantage of building neuromorphic systems based on Photonics rests in its speed of operation. The primary bottleneck in 'write' latencies arise from the programming time of the IF neuron which can also be performed at $200ps$. Although the current technology is power expensive during writing, the speed of writing still enables us to achieve a reasonable energy efficiency. With further optimization of switching techniques or by use of alternative PCMs with lower switching power, further energy benefits can also be aimed for to achieve comparable energy consumption to other technologies in the electrical domain. In turn, the proposed photonics computing platform eliminates various drawbacks usually faced in the electrical counterparts such as metal wire resistance, electromigration, sneak paths, etc. Despite the inherent challenges in the design and implementation, our proposed SNN framework based on GST-on-silicon photonics neuromorphic fabric enables parallelism through integration of a synaptic network with IF neurons. Such a design paves the way for scalable photonic architectures suitable for large-scale neuromorphic systems catered to perform fast computations.

8.4.6 Conclusion

We have proposed a photonic Spiking Neural Network computing primitive through seamless integration of non-volatile synapses and 'Integrate-and-Fire' Neurons based on Phase-change materials. The microring resonator devices explored for such synapses and neurons leverage the differential optical absorption of GST for non-volatility. We use the WDM technique to scale individual synapses into a large-scale synaptic array capable of performing parallelized dot-products. Our design is based on ring resonators of radius comparable to the wavelength of operation in order to achieve high area density while maintaining performance. We explore several challenges involved in such small ring resonators and proposed certain design modifications to achieve uniform and desirable characteristics across the entire operating range of wavelength. Finally, we developed a device to system level framework to evaluate the performance of the proposed photonic in-memory computing primitive and IF

neurons as an SNN inferencing engine by building behavioral models of the photonic neuromorphic fabric and achieve comparable performance to an ideal network. Neuromoprhic systems based on Integrated Photonics offer an alternative dimension to the current wave of exploring beyond von-Neumann computing frameworks and our proposed photonic SNN inferencing engine achieves a significant step towards proposing individual non-volatile devices capable of performing in-memory computing and scaling to a network of such devices to realize a truly integrated Spiking Neural Network.

9. SUMMARY AND FUTURE WORK

9.1 Summary

The primary goal of this research has been addressing challenges towards enabling intelligence in edge devices from the perspective of various levels of the design stack which include designing low complexity algorithms as well as enabling low-power hardware primitives. We started by exploring model compression techniques such as quantization in DNNs to reduce their computational complexity and memory footprint. We proposed a PCA-driven methodology to construct mixed precision neural networks which can achieve significantly higher accuracy than binary neural networks with comparable energy efficiency.

Next, we focused on the hardware aspect of reducing the energy consumption and area efficiency. In that regard, we worked toward building feasible analog in-memory computing primitives based on both CMOS and emerging technologies such as memristors. On one hand, we addressed the functional errors in analog computing primitives arising from various device and circuit non-idealities. We developed analytical models and mitigation techniques to account for such non-idealities. Furthermore, we built a more detailed data-based model, GENIEx, using a neural network to capture the co-dependence of the input paramaters and the non-idealities. We showed that GENIEx can model the functional errors in analog computing crossbars more accurately than analytical models. We further developed a functional simulation framework to evaluate large scale DNNs on such approximate crossbars using GENIEx. The proposed framework has been used as a tool to evaluate non-idealities in memristive crossbars in design adversarially robust hardware [268] and hardware-software co-design using Network Architecture Search [269] among other works.

Next, we designed compute-in-memory primitive and processing core based on 8T-SRAM arrays and implemented them in TSMC 65nm technology. The proposed primitive can perform matrix-vector multiplication operations within the memory array. Moreover, it can leverage sparsity in inputs and weights to adaptively reconfigure the peripheral ADC overhead based on sparsity and achieve higher energy efficiency. Further, we propose a multi-macro CIM processing micro-architecture which can not only leverage sparsity to reduce energy and latency cost, but also address concerns of low SNR analog computing using CIM

primitives as well as support re-arrangement of input and weight vectors to mitigate latency mismatch due to varying sparsity.

Having explored techniques to enable edge computing at the algorithm and circuit level, we further delved into the next level of stack, the devices. Here, we proposed bio-mimetic photonic devices based on phase-change materials which mimic the basic fundamental units of a neural network, such as neurons and synapses. We use Si micro-ring resonators with a GST element embedded on top of the ring waveguide to implement the neuronal and synaptic functionalities in the photonic domain. Next, we developed a SNN framework using the proposed devices to perform image classification tasks. Overall, all the aforementioned efforts and investigations can pave the way for energy-efficient algorithms and hardware platforms amenable for enabling intelligence at the edge.

9.2 Future Work

In the current state of the research, we have delved into crucial aspects of enabling low-power computing and algorithms for edge computing. As a future proposition, we will focus on addressing another important challenge that currently plagues analog computing. We have alluded to the high power consumption of ADCs in the analog computing primitives in Chapter 3, which drastically reduces its benefits over digital computing. We proposed a large-scale CIM processing (Chapter 7) core comprised of MVM units with reconfigurable precision ADCs which can leverage sparsity in the data to achieve high energy-efficiency while preserving accurate functionality. Such a CIM processing core can also enable flexible mapping and dataflow for DNN workloads. Although, CIM processors have been explored, the optimality of mapping, dataflow and corresponding architectural configuration is yet to be investigated. The proposed processing core can act as a platform for performing a thorough design-space exploration to search for optimal CIM accelerator configurations.

In this research, we have focused on accurate modeling of functional errors in analog computing primitives. We have also looked at mitigation techniques for analytical models. However, there is a further need of looking at corrective measures in the degradation in accuracy observed for large-scale DNNs when evaluated in a functional framework consisting

of key architectural facets such as bit slicing and bit streaming. There is a need to develop hardware mapping techniques, which could be error comepnsation circuits as well as offline training algorithms to counter computational errors in large-scale DNNs due to device and circuit non-idealities. In summary, future focus in analog CIM design can answer the aforementioned open-ended questions regarding optimality as well as functionality which would compliment this research effectively and move us closer to making it feasible for deployment in edge devices.

A. ENERGY EFFICIENCY AND MEMORY CALCULATIONS FOR DNNS

A.1 Energy Efficiency

The primary model-dependent metrics that affect the energy consumption of classification task are the energies consumed by the computations (multiply-and-accumulate or MAC operations) and memory accesses in our calculations for energy efficiency. We exclude energy consumed due to data flow and instruction flow in the architecture. For a convolutional layer, there are I input channels and O output channels. Let the size of the input be $N \times N$, size of the kernel be $k \times k$ and size of the output be $M \times M$. Thus, in Table A.1 we present the number of memory-accesses N_{M-FP} and computations N_{C-FP} for standard full-precision (FP) networks:

The number of binary memory accesses ($N_{M\mathrm{i}}$) and computations ($N_{C\mathrm{i}}$) in a binary layer is same as the corresponding number in full-precision layer of equivalent dimensions. As explained in Eq. 1, we consider additional full-precision memory accesses and computations for parameter α, where α is the scaling factor for each filter bank in a convolutional layer. Number of accesses for α is equal to the number of output maps, O. Number of full-precision computations are $M^2 \times O$. Table A.1 lists the number of k-bit and full-precision memory access and computations of any layer. We calculated the energy consumption from projections for 45 nm CMOS technology [36], [270]. Considering 32-bit representation as full-precision, the energy consumption for both binary and 32-bit memory accesses and computations are shown in Table. A.1.

Then, energy consumed by any layer with k-bit weights and activations is given by

$$E_{\mathrm{i}} = N_{A-F}E_{A-32} + N_{A-k}E_{A-k} + N_{C-F}E_{C-kF} + N_{C-k}E_{C-kI} \tag{A.1}$$

Note, this calculation is a rather conservative estimate which does not take into account other hardware architectural aspects such as input-sharing or weight-sharing. However, our approach concerns with modifications of network architecture and we compare the ratios of energy consumption. These aspects of the hardware architecture affect all the networks

Table A.1. Number of operations in a k_b-bit layer

Operations in neural networks		
Operation	Number of Operations	
Input Read	$N^2 \times I$	
Weight Read	$k^2 \times I \times O$	
Computations (MAC)	$M^2 \times I \times k^2 \times O$	
Memory Write	$M^2 \times O$	
Number of operations of k_b-bit layer		
Operation	Term	Number of Operations
k-bit Memory Access	N_{A-k}	$N^2 \times I + k^2 \times I \times O$
k-bit Computations (MAC)	N_{C-k}	$M^2 \times I \times k^2 \times O$
FP Memory Access	N_{A-F}	O
FP Computations	N_{C-F}	$M^2 \times O$
Energy Consumption Chart		
Operation	Term	Energy (pJ)
k-b Memory Access	E_{A-k}	$2.5k$
32-b MULT FP	E_{M-F}	3.7
32-b MULT INT	E_{M-I}	3.1
32-b ADD FP	E_{AD-F}	0.9
32-b ADD INT	E_{AD-I}	0.1
k-bit MAC INT	E_{C-kI}	((3.1*k)/32+0.1)
k-bit MAC FP	E_{C-kF}	4.6

equally and hence can be taken out of consideration. Further, FP MAC operations can be optimized for lower energy consumptions. In our calculations, we have bluntly taken it as the sum of a 32-b FP Multiply and 32-b FP Add operations. These optimizations are catered towards FP networks, and reduce the FP energy consumption. This, in turn, will reduce the energy efficiency of the binary and hybrid networks. In this work, we are focused on comparing different kinds of binary and hybrid network, and hence, this assumption of FP MAC energy is not going to affect the analysis.

A.2 Memory Compression

The memory required for any network is given by product of the total number of weights in the network multiplied by the precision of the weights. The number of weights in any layer is given by:

$$N_{w-\mathrm{i}} = I_{\mathrm{i}} \times O_{\mathrm{i}} \times k^2 \tag{A.2}$$

considering usual notations describer earlier. Thus, the total memory requirements can be simply written as $M_{\mathrm{i}} = \sum_{\mathrm{i}} N_{w-\mathrm{i}} * k_{b-\mathrm{i}}$ where $k_{b-\mathrm{i}}$ is the precision of weights in the i^{th} layer. We can estimate memory compression (M.C) with respect to a full-precision network and normalize it with respect to an XNOR-Net network which is an entirely binary network except the first and final layer.

Note that the assumption for the energy and storage calculations for binary layers hold for custom hardware capable of handling fixed-point binary representations of data, thus leveraging the benefits offered by quantized networks.

B. CROSSBAR NON-IDEALITIES

As discussed earlier, the analog nature of computing in NVM crossbars can lead to approximations in their functionality as MVM units. Such approximations arise from device and circuit non-idealities originating from metal-lines, peripheral circuits and non-ideal devices. It is important to delineate the sources of these approximations and their impact on the functionality. In this section, we provide a detailed analysis and perspective on the implication of the imperfections in resistive crossbars on MVM computations. In Section 4, we have delineated several techniques to overcome the effects of these non-idealities. The mentioned device-circuit non-idealities can be categorized into 2 types: 1) Read non-idealities, and 2) Write non-idealities.

B.1 Read non-idealities

Generally, read non-idealities refer to the imperfections in the devices and circuits that can lead to functional read errors in MVM computations. Further, read non-idealities can be categorized into two parts:

1. Linear read non-idealities
2. Non-linear read non-idealities

B.1.1 Linear read non-idealities

Linear read non-idealities comprise three kinds of parasitic resistances: i) Wire resistance, ii) Source resistance, iii) Sink resistance. Figure B.1 shows a typical memristive crossbar with parasitic resistances.

Wire resistances originate from the interconnects between memristive cells in BLs and WLs. Typically, the wire resistance can be calculated as: $R_{wire} = \frac{\rho l}{wh}$, where ρ, l, w, and h are the interconnect resistivity, wire length, width, and height, respectively [271]. Therefore, the wire resistance parasitic effect increases with crossbar size due to the increase in wire lengths. Further, with technology scaling the wire resistance increases at more advanced nodes due to the reduced width [271].

Source and sink resistances represent the WL analog drivers output resistance and the input resistance of the current sensing circuitry, respectively. Such parasitic resistances contribute significantly to current-resistance (IR) drops across the BLs and WLs of the crossbar, thus degrading the output current from its ideal value.

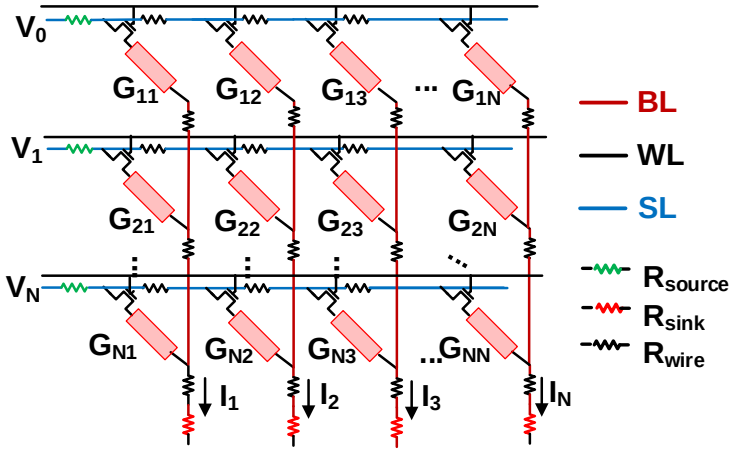


Figure B.1. resistive crossbars with parasitic resistances, arising from bit-line and word-line wire resistances (R_{wire}), input driver resistance (R_{source}), and sensing resistance (R_{sink}).

B.1.2 Non-linear read non-idealities

Non-linear read non-idealities originate from the non-linear behavior of both NVM devices and access transistors. The read operation is affected by the non-linear current (I) vs voltage (V) characteristics of the NVM devices. Besides NVM devices, the transistor characteristics can also introduce non-linearity in the computations.

a) Device read non-linearity: Different device technologies can exhibit different kinds of non-linear behavior due to the fundamental dissimilarities in current conduction. For example, transport in amorphous PCM devices have been described by Poole-Frenkel transport [272] of carriers which results in the current being linear at small voltages and exponential at higher voltages. In crystalline state, PCM devices exhibit Ohmic behavior at low voltages, and non-Ohmic behavior at high voltages due to Joule heating. The compact model for such non-linear I-V behavior has been previously studied. [273], [274].

Unlike PCMs, RRAM conduction relies on tunnelling mechanism which results in an exponential dependence of current on voltage. This can be expressed as [176]: $I(d, V) = I_0 exp(\frac{d}{d_0}) sinh(\frac{V}{V_0})$ Here, d is the gap-size between the tip of the filament and electrode, I_0, d_0 and V_0 are fitting parameters.

Similarly, spin devices show voltage-dependent resistive characteristics. Additionally, the non-linear I v/s V characteristics in spin devices can be modeled using Non-equilibrium Green's Function (NEGF) [275]. Figure B.2 (a) shows the non-linear I-V characteristics of (i) PCM, (ii) RRAM, and (iii) spintronic technologies for $R_{ON} = 10k\Omega$. It can be observed that all the NVM technologies exhibit I-V characteristics that deviate from the expected linear resistive behavior. Such non-linear behavior can result in functional errors when MVM units are constructed using these devices as MVM operations demand linearity of the output (current) with the input variables (V and G). Indeed, non-linearities during the read process in resistive crossbars can pose a challenge toward accurate MVM functionality.

b) Access device non-linearity:

The non-linearity of access devices can play a role in introducing functional inaccuracies in MVM units. Various access devices have been adopted in crossbar-based MVM units as mentioned in Section II.B.1. Such access devices (two-terminal selectors or transistors) impose different nonlinear characteristics on the analog computing operations in crossbars.

Two-terminal selectors: Two-terminal selectors can switch the current passing into the memory device by exponential switching or threshold switching as mentioned in Section **??**. Interestingly, 1ES-1R based crossbars show more nonlinearity effects over different technologies than 1TS-1R based crossbars. According to [95], 1S-1R cell $I - V$ characteristics strongly depend on the selector type (ES or TS) after the RRAM state has been changed, which is clearly shown in the linear scale I–V curves in Figure B.2 (b). ES inherently shows a slow slope of several hundred mV/decades due to the exponential switching nature. Therefore, a certain amount of the cell voltage is continuously applied to ES even after it is turned on during the 1ES-1R operation, resulting in the nonlinear I–V curves, as shown in Figure B.2(b) (i). On the other hand, the applied voltage to 1TS-1R cell beyond TS threshold voltage is fully transferred to the RRAM due to the fast slope that is at most 5 mV/decade

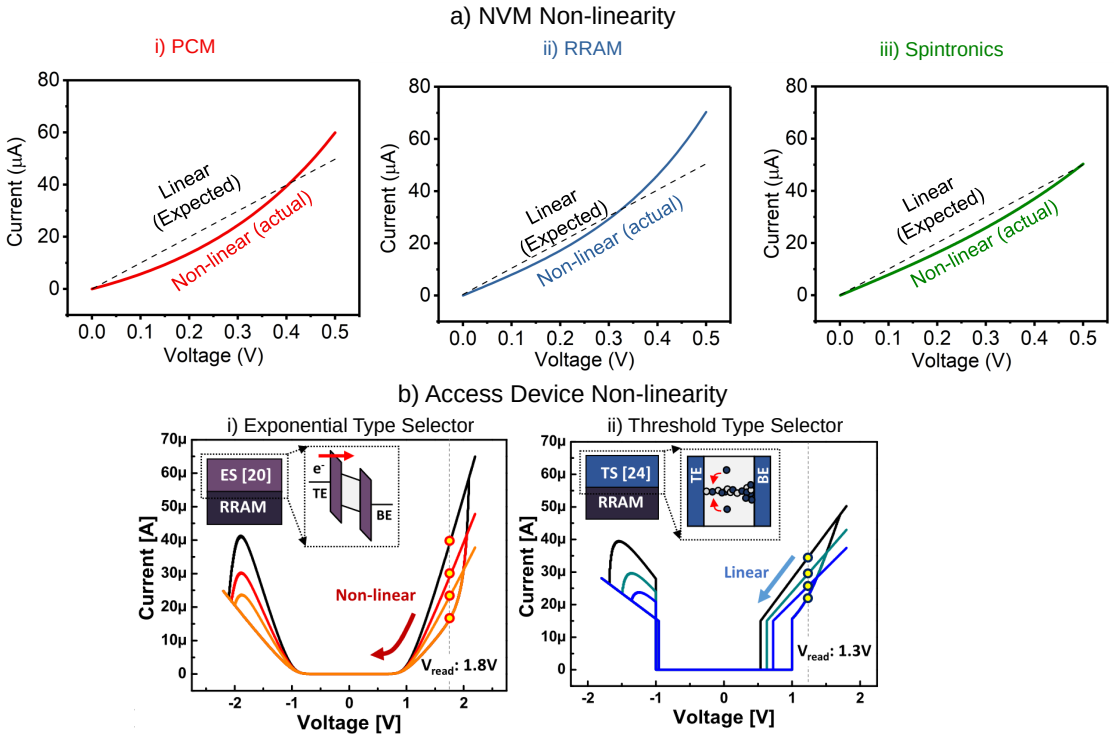


Figure B.2. a) I-V characteristics for a typical i) PCM, ii) RRAM and iii) Spintronic device for $R_{ON} = 10k\Omega$ exhibiting non-linear behavior, resulting in significant deviation from expected linear characteristic. b) i) I–V trace of the 1ES-1R follows the exponential curve of the ES, ii) the semi-linear I–V curve of the 1TS-1R caused by the RRAM [95].

[95]. As a result, the linear I–V trace of the 1TS-1R device governed by the RRAM is shown in Figure B.2(b) (i).

Transistors as access devices: generally, 1T-1R cells behave more linearly than 1S-1R. The reason for that is transistors in 1T-1R cells operate in linear region during the MVM operation. Moreover, there is no threshold for the applied voltage to reach before switching since it is a three-terminal switch. To that end, 1T-1R exhibits less significant nonlinearity than two-terminal selector based cells with the cost of large cell area.

B.2 Write non-idealities

The conductance tuning operation in NVM devices is desired to be linear and symmetric, especially for on-line weight update algorithms. However, a typical curve between the con-

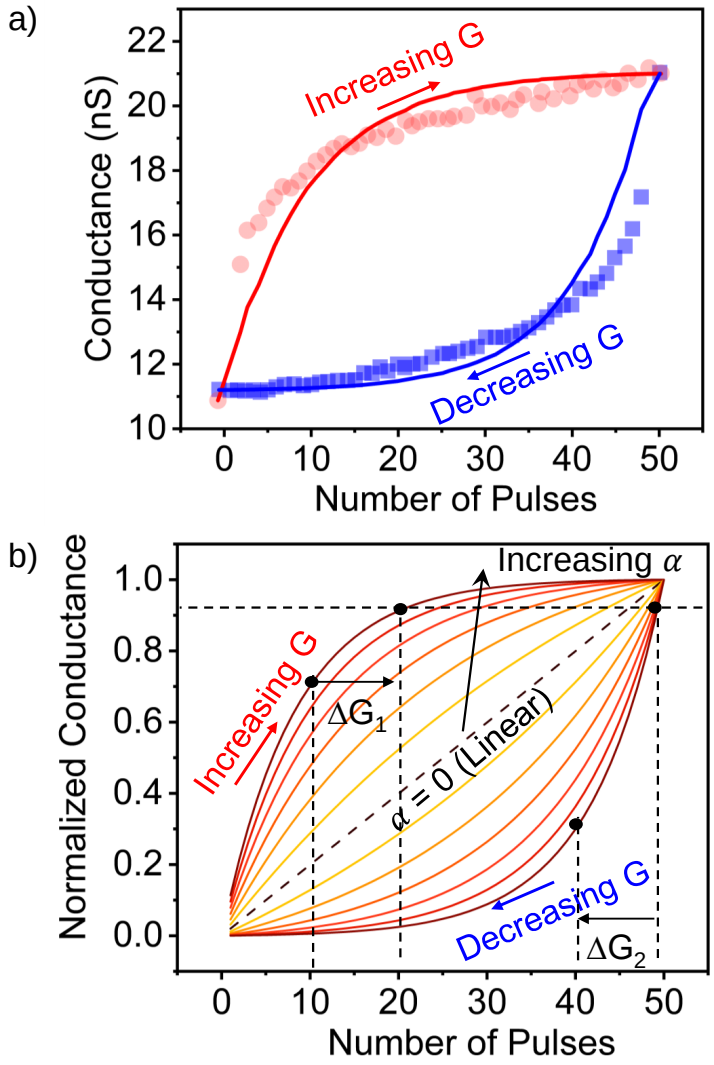


Figure B.3. a) Conductance (G) evolution curve with respect to number of programming pulses [276]. (b) Non-linear conductance curve model vs number of programming pulses for varying degree of non-linearity, showing asymmetry between increasing and decreasing conductance trajectories.

ductance and the number of programming pulses can significantly deviate from the desired

linear and symmetric behavior, as shown in Figure B.3 (a). This behavior has been quantitatively modeled by researchers for RRAMs [171], [276] through the following equations:

$$G_{+} = B(1 - \mathrm{e}^{-\frac{P}{\alpha}}) + G_{min} \tag{B.1}$$

$$G_{-} = -B(1 - \mathrm{e}^{\frac{P - P_{max}}{\alpha}}) + G_{max} \tag{B.2}$$

$$B = (G_{max} - G_{min})/(1 - \mathrm{e}^{-\frac{P_{max}}{\alpha}}) \tag{B.3}$$

Here P is the number of applied pulses, $G_{+}(G_{-})$ is the conductance trajectory in the increasing (decreasing) direction, $G_{min}(G_{max})$ is the minimum (maximum) conductance, and P_{max} is the maximum pulse number required to entirely switch the device between 2 extreme states. The parameter α controls the non-linearity behavior, where as B is a fitting function. Typically, α can range from +6 to -6, where $\alpha = 0$ denotes linear behavior. Figure B.3 (b) shows the non-linear conductance trajectory for varying range of α. In this figure, we also observe that due to the asymmetric trajectory of increasing and decreasing conductance, same difference in number of applied pulses can lead to drastically different conductance update.

The non-linearity and asymmetry in NVM devices adversely affect the on-line learning schemes due to the continuous and gradual weight update in such schemes. For off-line trained neural networks, the conductances of the NVM devices in the crossbar are predetermined, and hence can be reliably written using the popular read-verify-write mechanism [277].

B.3 Impact of non-idealities on the output current

The impact of the aforementioned non-idealities arising from device characteristics and circuit parasitics introduces functional errors in MVM computations (read non-idealities) as well as weight updates (write non-idealities) during on-line training.

Table B.1. Crossbar Simulation Parameters

Category	Parameter	Value Ranges	References
Design Parameters	Crossbar Size	16, 32, 64	Design Choices
Circuit Non-idealities	R_{source}	1000Ω	
	R_{sink}	150Ω	
	R_{wire}	2.5Ω	
Device Parameters	I_0	0.1 mA	[179], [180]
	g_0	0.25 nm	
	V_0	0.25 V	
	Number of bits	1, 2, 4	[59], [61], [65], [74]
	ON Resistance	50k, 100k, 300k	
	ON/OFF Ratio	2, 6, 10	

B.3.1 Read non-idealities

We study the impact of the read non-idealities on the output current of the crossbar. We consider RRAM device characteristics for simplicity. To study the cumulative impact of each non-ideality, we consider a 64x64 1T-1R crossbar and perform SPICE simulations for various input V and G matrix combinations. The parameters for SPICE simulations are presented in Table B.1. Figure B.4 shows that different voltage (V) and conductance (G) conditions which lead to similar ideal current (I_{ideal}) can result in a varying range of non-ideal current ($I_{non-ideal}$) outputs, causing errors in computations. The difference can be quantified using a non-ideality factor (NF) which is the relative error between the I_{ideal} and $I_{non-ideal}$. NF is calculated as:

$$\frac{I_{ideal} - I_{non-ideal}}{I_{ideal}} \tag{B.4}$$

Figure B.4 (a) and (b) shows the I_{ideal} v/s $I_{non-ideal}$ characteristics for the crossbar under the influence of first, only linear non-idealities (blue), and second, both linear and non-linear non-idealities (red), for different supply voltages (V_{supply}). We observe that the effects of

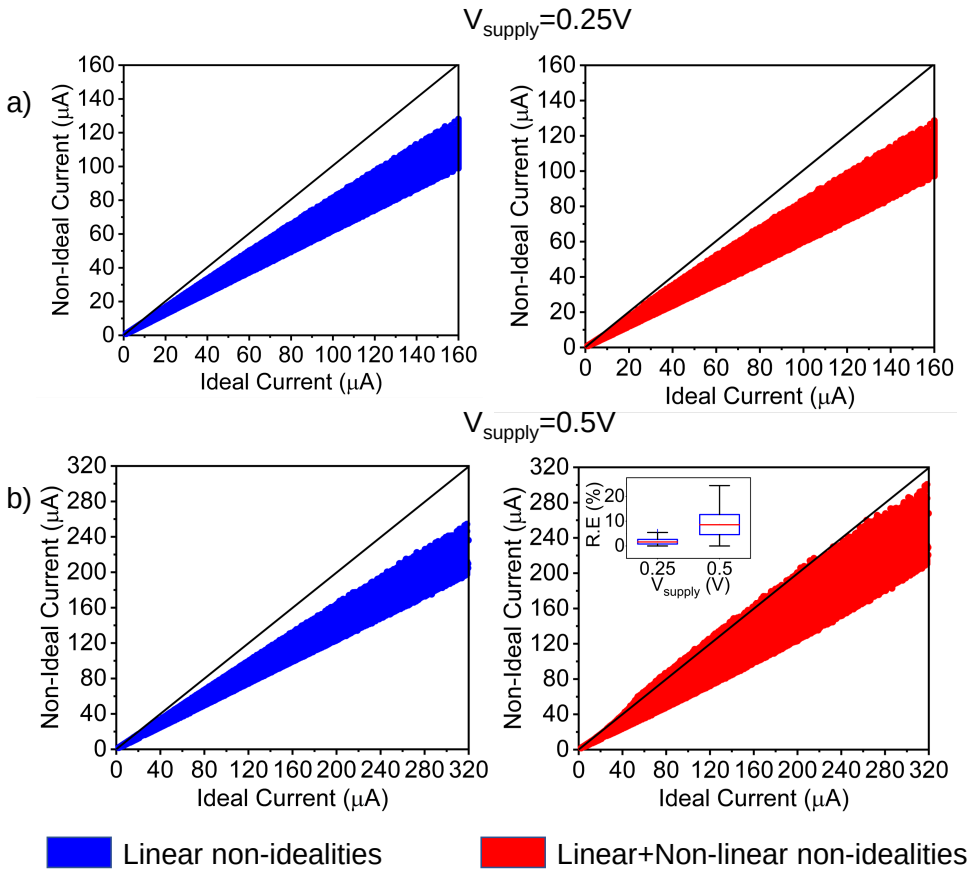


Figure B.4. Ideal (I_{ideal}) v/s Non-ideal ($I_{non-ideal}$) current plots for a) V_{supply} = 0.25V and b) 0.5V showing that the case with both linear and non-linear non-idealities have higher errors than solely linear non-idealities, particularly for higher supply voltages. Inset shows the relative error (R.E) for $V_{supply} = 0.25V/0.5V$ in $I_{non-ideal}$ between the two cases (blue and red).

non-linearity become more prominent at higher V_{supply}, resulting in a higher difference in NF between the red and blue plots.

We study the impact of individual contributions of different non-idealities mentioned in Section B-A. The simulations based on the parameters listed in Table. B.1 show that source resistance has the highest mean NF among linear non-idealities in Figure B.5. This is due to the high source resistance. The non-linear device and transistor non-idealities have a negative NF, i.e, it causes $I_{non-ideal}$ to be higher than I_{ideal} unlike the linear non-idealities. This effect is pronounced for higher supply voltages (0.5V). The standard deviation of NF is also primarily contributed by device non-linearity which gets significantly magnified for

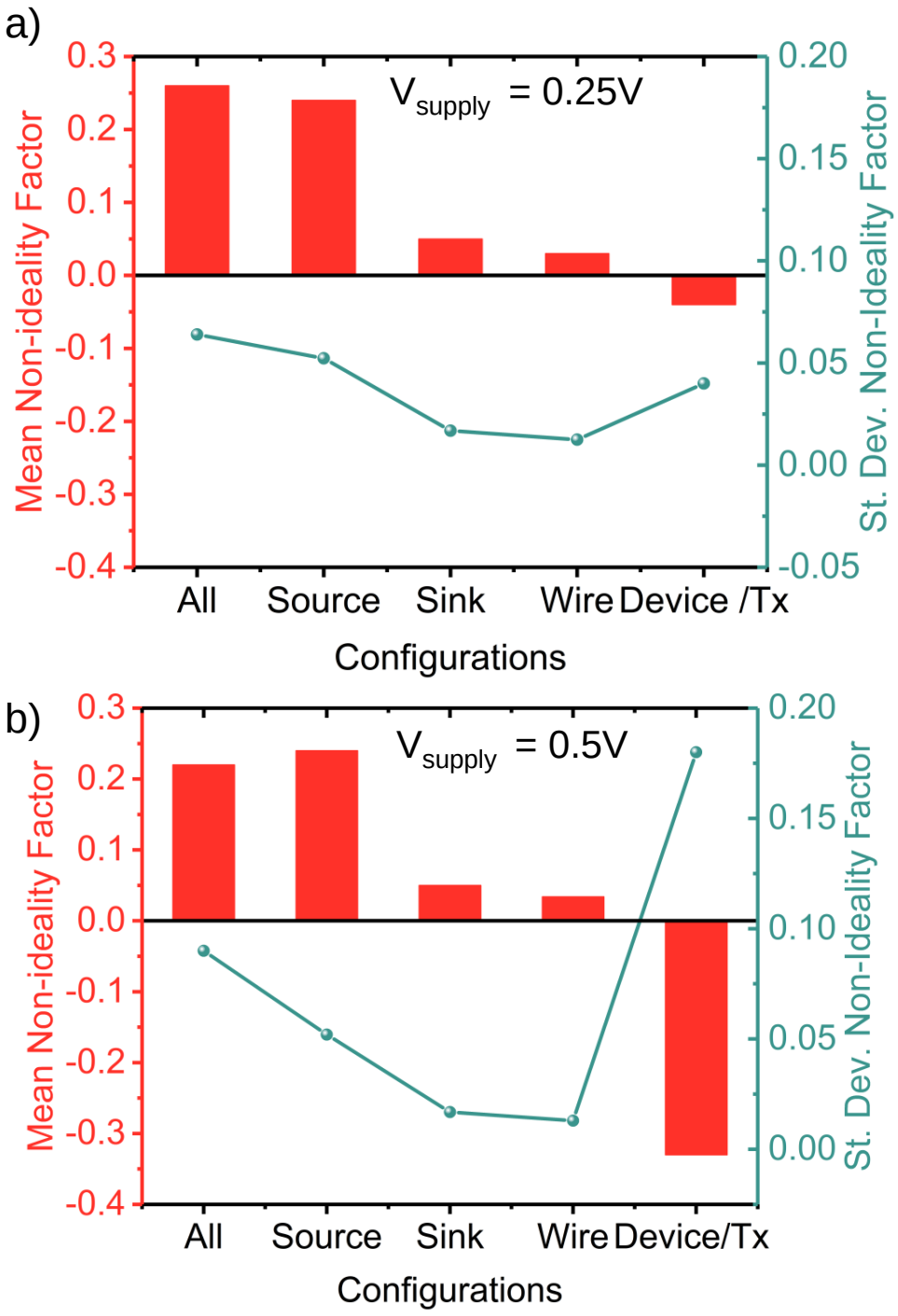


Figure B.5. Mean and standard deviation of non-ideality factor (NF) highlighting the individual contributions of the different sources of read non-idealities mentioned in Section B-A. The configurations 'Source', 'Sink', 'Wire' and 'Device/Tx' refer to cases where only the impact of source, sink, wire resistance and device/transistor non-linearities were considered respectively. The configuration 'All' refer to the case when all non-idealities are considered.

higher supply voltages. Thus, the device and transistor non-linearities act in the opposite direction to the linear non-idealities. This necessitates accurate modeling of these non-linear non-idealities.

Read non-idealities have a significant impact on crossbar design parameters both from the device and circuit point of view. These design parameters are namely, a) Crossbar size, b) ON resistance, c) ON/OFF ratio and d) Bits per device.

a. Crossbar size: The size of the crossbar has considerable effect on the impact of parasitics on MVM functionality. The reason is two-fold; first, larger crossbars consist of longer metal lines, thereby, increasing the impact of wire resistance. Second, higher crossbar size also amounts to lowering the effective resistance, as seen, from the input WL drivers. A lower effective resistance results in increasing impact of parasitic resistances on the MVM functionality. Figure B.6 (a) shows the trend of non-ideality factor for varying crossbar sizes from 16×16 to 64×64. It can be observed that mean non-ideality factor, NF, increases by $\sim 5\times$ as crossbar size increases from 16×16 to 64×64. Although, using lower crossbar sizes would reduce functional errors in large-scale DNNs, it would also lead to more crossbars for representing a particular network. This results in an energy/latency-accuracy tradeoff from a system design point of view.

b. ON resistance: The ON resistance is referred to as the resistance of the Low Resistance State (LRS) in the NVM device. For a given crossbar size and resistance ON/OFF ratio, the effective resistance of the crossbar seen from each input is also determined by the ON resistance. A lower ON resistance reduces the effective resistance, which results in an increased detrimental effect of parasitic resistances. Figure B.6 (b) shows that mean non-ideality factor, NF, is $\sim 5\times$ higher for $R_{ON} = 50k\Omega$ than for $R_{ON} = 300k\Omega$. It is intuitive to use higher ON resistance to reduce the impact of parasitics on the MVM functionality. From an energy point of view, using higher ON resistance can also reduce the array energy due to lower passive power ($P = V/R$) dissipation. One disadvantage of using higher ON resistance, however, is need for higher sense margin that can lead to higher peripheral overhead.

c: ON/OFF ratio: For a fixed ON resistance and Crossbar Size, the ON/OFF ratio of the NVM device resistance also affects the impact of non-idealities on the output current of the crossbar. This is because, for a given ON resistance, a lower ON/OFF ratio reduces the

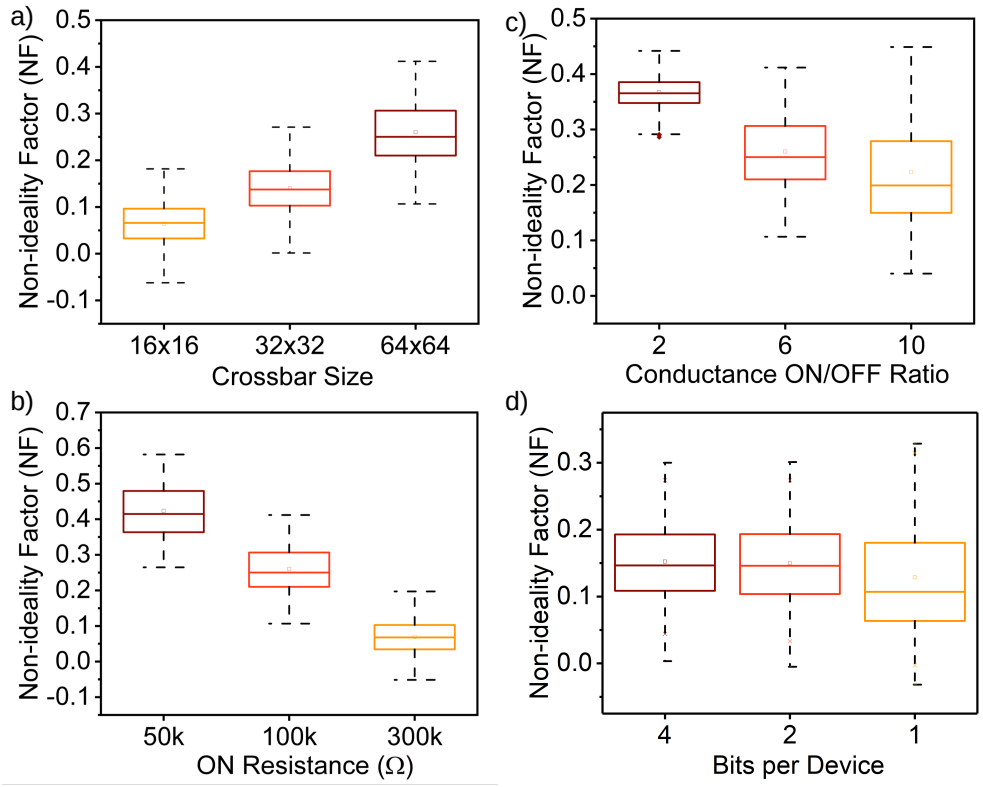


Figure B.6. Impact of read non-idealities on resistive crossbar output, expressed as NF (Equation (11)), on crossbar design parameters such as a) Crossbar Size, b) Conductance ON/OFF ratio, c) ON resistance and d) Bits per device.

average resistance in each column of the crossbar. Thus, a lower ON/OFF ratio results in a higher impact of parasitics. Figure B.6 (c) shows that mean non-ideality factor, NF, is $\sim 1.75\times$ higher for ON/OFF Ratio $= 2$ than for ON/OFF Ratio $= 10$. However, a ON/OFF Ratio $= 10$ has a higher deviation in NF, implying a strong dependence of NF on the varying resistance range in the crossbar. Despite that variation, a higher ON/OFF resistance ratio is desirable in NVM devices for proper MVM functionality in resistive crossbars.

d: Bits per device: Finally, the bits per device can have some effect on NF as shown in Figure B.6 (d). It can be observed that although there is not a significant difference between the NF distribution when the number of bits per device is 2 or 4, but storing 1 bit per device certainly results in a lower mean NF but with higher deviation. This is because storing 1 bit per device, skews the resistance distribution in the crossbar toward either R_{ON}

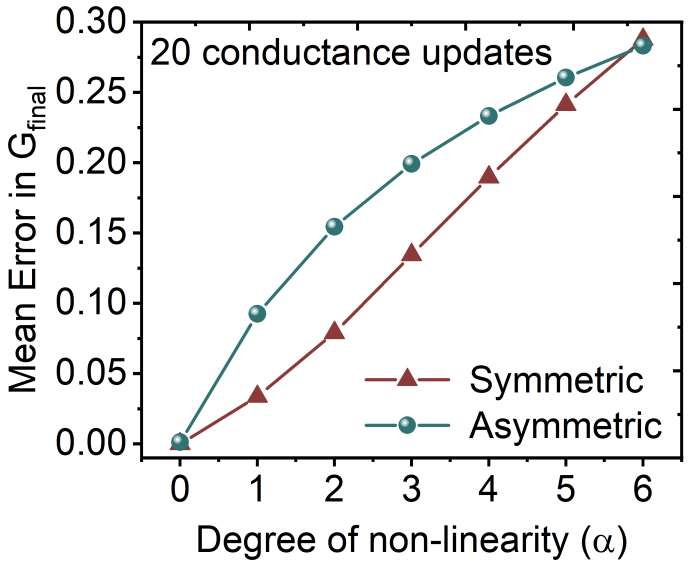


Figure B.7. Impact of device write non-linearity and asymmetry on conductance on mean error in final conductance, G_{final}, calculated by the mean relative error between the desired (linear) final conductance value after 20 updates and the achieved final conductance. Here, symmetric (asymmetric) update means that the conductance trajectories, shown in Fig. B.3 (b), are identical (different) for negative and positive weight updates.

or R_{OFF}, thus increasing the effective ON/OFF ratio of the crossbar. Thus, this impact can be correlated to higher ON/OFF ratio when a similar trend is observed. Packing lower number of bits per device might reduce the average NF but it harms the storage density of the crossbars, which will their reduce area efficiency.

The variation in NF with various design parameters provides us with insights on how to design crossbars with lower NF. Lower crossbar sizes with high ON resistances, high ON/OFF ratio and low bits per devices might seem preferable, but as we have described before, each of these design choices have their own negative implications. Thus, careful design space exploration needs to be performed to achieve optimal energy-latency-accuracy trade-off.

B.3.2 Write Non-idealities

The impact of write non-idealities becomes prominent during on-line training on crossbars. To quantify the impact of the write non-idealities, we study the deviation of the conductance matrix under the influence of write non-linearity and asymmetry explained above from the desired values. Increasing degrees of non-linearity, α, in Equation (8), can affect the iterative conductance update quite severely. Figure B.7 depicts the impact of device write non-linearity and asymmetry on the final conductance achieved after 20 weight update iterations. The mean error in conductance, G, is calculated as the mean relative error between the desired (linear) final conductance value after 20 updates and the achieved final conductance. The symmetric updates exhibits lower error than the asymmetric updates, although, for high non-linearity, both can have a significant deviation from ideal. In Section **??**-C, we will delineate the impact of device write non-linearity and asymmetry on application accuracy for systems with on-line learning.

B.3.3 Process Variations

Thus far, we have discussed non-ideal device and circuit behavior leading to errors in MVM computations and weight update operations. In addition, a major challenge in realizing large-scale resistive crossbars is process variations, which can add undesirable randomness to the device behavior. Broadly, process variations in resistive crossbars can be categorized into:

1. Device to Device Write Variations
2. Cycle to Cycle Write Variations
3. Device to Device Read Variations

B.3.4 Device to Device Write Variations

During the write process, NVM devices can exhibit considerable variation from one device to another in the same crossbar. This can be modelled by adding a variation term in

the non-linearity programming equation (Equation (5-7)). The degree of non-linearity in programming can vary device to device with a typical $3\sigma = \pm 1.5$. Figure B.8 (a) illustrates how device to device variations manifest in the form of programming characteristics.

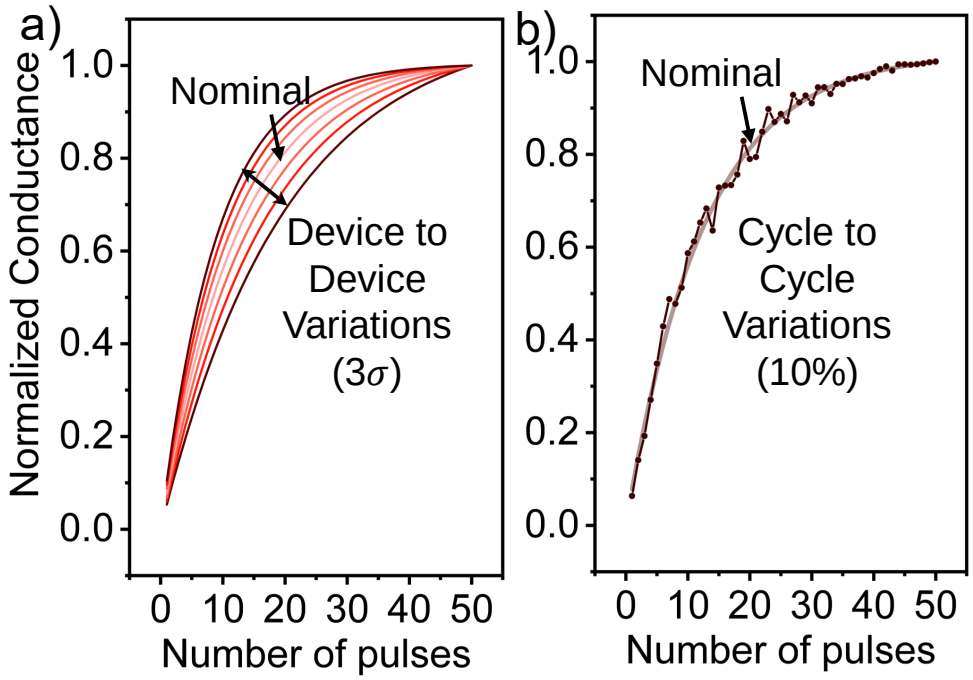


Figure B.8. Impact of a) Device to device variations and b) cycle to cycle variations on Normalized Conductance v/s Number of pulses characteristics.

B.3.5 Cycle to Cycle Write Variations

Cycle to cycle write variations can be described as the variations in conductance characteristics at every programming pulse. It can be modeled by considering a distribution in the non-linearity factor for different programming pulses. Figure B.8 (b) illustrates how device to device variations manifest in the form of programming characteristics.

The write variations primarily affect the on-line learning schemes on the NVM devices. We will study the impact of such variations on application accuracy in Section **??**.

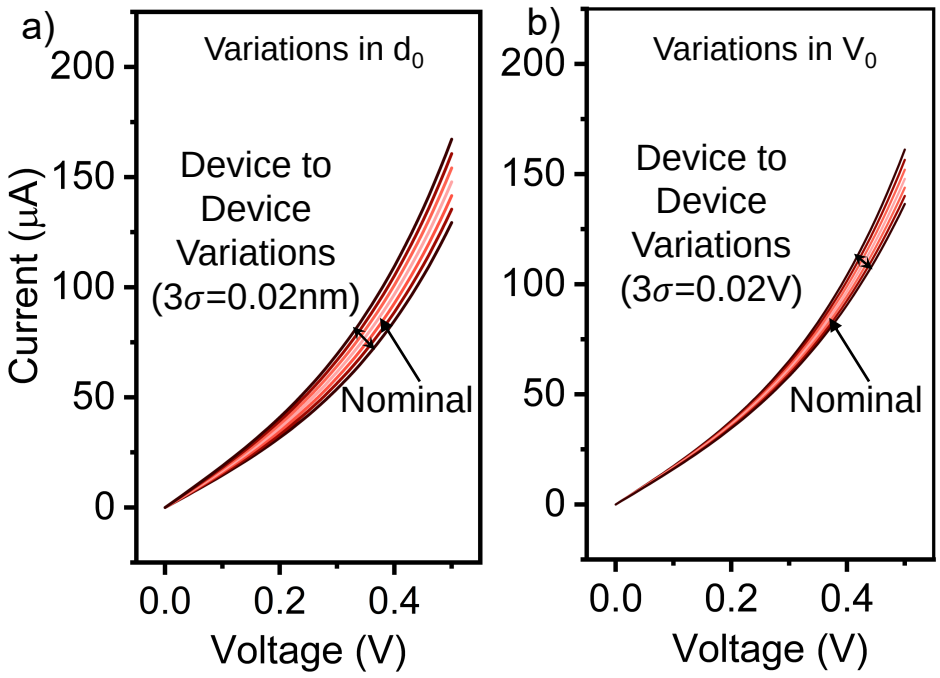


Figure B.9. Device to device variations manifesting in the read operation of the NVM device. Variations in the fitting parameters d_0 and V_0 in Equation (7) results in a variation in I-V characteristics.

B.3.6 Device to Device Read Variations

The device to device variations can also reflect in the IV characteristics of the device which can affect the MVM functionality of resistive crossbars. To analyze such an effect, the variations can be included in the fitting parameters of the compact model of such devices. For example, in RRAM devices, Equation (4) can be modified to include the effect of variations through the parameters d_0 and V_0. Figure B.9 illustrates how device to device variations manifest in the form of IV characteristics.

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," in *Parsing the Turing Test*, Springer, 2009, pp. 23–65.

[2] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[3] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual ISCA*, IEEE, 2017, pp. 1–12.

[4] Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[6] He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] Girshick *et al.*, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[8] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[9] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2017, pp. 1–12.

[10] E. Chung *et al.*, "Serving dnns in real time at datacenter scale with project brainwave," *IEEE Micro*, vol. 38, no. 2, pp. 8–20, 2018.

[11] A. Biswas and A. P. Chandrakasan, "Conv-ram: An energy-efficient sram with embedded convolution computation for low-power cnn-based machine learning applications," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 488–490.

[12] H. Valavi *et al.*, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE JSSC*, 2019. DOI: 10.1109/JSSC.2019.2899730.

[13] Q. Dong *et al.*, "A 351tops/w and 372.4gops compute-in-memory sram macro in 7nm finfet cmos for machine-learning applications," in *ISSCC*, 2020. DOI: 10.1109/ISSCC19947.2020.9062985.

[14] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu, *et al.*, "24.5 a twin-8t sram computation-in-memory macro for multiple-bit cnn-based machine learning," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 396–398.

[15] A. Ankit *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, 2019, pp. 715–731.

[16] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[17] I. Chakraborty, D. Roy, I. Garg, A. Ankit, and K. Roy, "Constructing energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 43–55, 2020.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[19] I. Chakraborty, M. Ali, A. Ankit, S. Jain, S. Roy, S. Sridharan, A. Agrawal, A. Raghunathan, and K. Roy, "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2276–2310, 2020.

[20] A. Ankit, I. Chakraborty, A. Agrawal, M. Ali, and K. Roy, "Circuits and architectures for in-memory computing-based machine learning accelerators," *IEEE Micro*, vol. 40, no. 6, pp. 8–22, 2020.

[21] I. Chakraborty, A. Jaiswal, A. Saha, S. Gupta, and K. Roy, "Pathways to efficient neuromorphic computing with non-volatile memory technologies," *Applied Physics Reviews*, vol. 7, no. 2, p. 021 308, 2020.

[22] I. Chakraborty *et al.*, "Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 5, pp. 335–344, 2018.

[23] I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2020, pp. 1–6.

[24] A. Jaiswal *et al.*, "8t sram cell as a multibit dot-product engine for beyond von neumann computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[25] M. Ali, I. Chakraborty, U. Saxena, A. Agrawal, A. Ankit, and K. Roy, "A 35.5-127.2 tops/w dynamic sparsity-aware reconfigurable-precision compute-in-memory sram macro for machine learning," *IEEE Solid-State Circuits Letters*, 2021.

[26] I. Chakraborty, G. Saha, A. Sengupta, and K. Roy, "Toward fast neural computing using all-photonic phase change spiking neurons," *Scientific reports*, vol. 8, no. 1, pp. 1–9, 2018.

[27] I. Chakraborty, G. Saha, and K. Roy, "Photonic in-memory computing primitive for spiking neural networks using phase-change materials," *Physical Review Applied*, vol. 11, no. 1, p. 014 063, 2019.

[28] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013. DOI: 10.1016/j.future.2013.01.010. [Online]. Available: https://doi.org/10.1016%5C%2Fj.future.2013.01.010.

[29] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense," in *Proceedings of the 26th International Conference on World Wide Web - WWW'17*, ACM Press, 2017. DOI: 10.1145/3038912.3052577. [Online]. Available: https://doi.org/10.1145%5C%2F3038912.3052577.

[30] L. M. Kaufman, "Data security in the world of cloud computing," *IEEE Security & Privacy*, vol. 7, no. 4, pp. 61–64, 2009.

[31] N. Gonzalez, C. Miers, F. Redigolo, M. Simplicio, T. Carvalho, M. Näslund, and M. Pourzandi, "A quantitative analysis of current security concerns and solutions for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 11, 2012.

[32] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah, "Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition," in *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, IEEE, 2016, pp. 64–76.

[33] Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[34] Alvarez and Salzmann, "Compression-aware training of deep networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 856–867.

[35] Weigend *et al.*, "Generalization by weight-elimination with application to forecasting," in *Advances in neural information processing systems*, 1991, pp. 875–882.

[36] Han *et al.*, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[37] Ullrich *et al.*, "Soft weight-sharing for neural network compression," *arXiv preprint arXiv:1702.04008*, 2017.

[38] Hubara *et al.*, "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.

[39] Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, Springer, 2016, pp. 525–542.

[40] I. Garg, P. Panda, and K. Roy, "A low effort approach to structured cnn design using pca," *arXiv preprint arXiv:1812.06224*, 2018.

[41] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," *arXiv preprint arXiv:1709.01134*, 2017.

[42] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 722–737.

[43] 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.

[44] S.-C. Zhou, Y.-Z. Wang, H. Wen, Q.-Y. He, and Y.-H. Zou, "Balanced quantization: An effective and efficient approach to quantized neural networks," *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 667–682, 2017.

[45] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 365–382.

[46] S. Jung, C. Son, S. Lee, J. Son, Y. Kwak, J.-J. Han, and C. Choi, "Joint training of low-precision neural network with quantization interval parameters," *arXiv preprint arXiv:1808.05779*, 2018.

[47] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[48] B. Graham, "Low-precision batch-normalized activations," *arXiv preprint arXiv:1702.08231*, 2017.

[49] Prabhu *et al.*, "Hybrid binary networks: Optimizing for accuracy, efficiency and memory," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 821–829.

[50] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of convnets via differentiable neural architecture search," *arXiv preprint arXiv:1812.00090*, 2018.

[51] C. Sakr and N. Shanbhag, "Per-tensor fixed-point quantization of the back-propagation algorithm," *arXiv preprint arXiv:1812.11732*, 2018.

[52] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[53] Krizhevsky and Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[54] 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.

[55] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[56] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422–1432.

[57] X. Xu *et al.*, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, pp. 216–222, 2018.

[58] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[59] S. Ambrogio *et al.*, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, p. 60, 2018.

[60] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Advanced Materials*, vol. 30, no. 9, p. 1 705 914, 2018.

[61] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–cmos system for efficient multiply–accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290–299, 2019.

[62] A. Sengupta and K. Roy, "A vision for all-spin neural networks: A device to system perspective," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2267–2277, 2016.

[63] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333–343, 2018.

[64] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electronics*, vol. 1, no. 1, p. 22, 2018.

[65] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[66] S. Jain, A. Ankit, I. Chakraborty, T. Gokmen, M. Rasch, W. Haensch, K. Roy, and A. Raghunathan, "Neural network accelerator design with resistive crossbars: Opportunities and challenges," *IBM Journal of Research and Development*, vol. 63, no. 6, pp. 10–1, 2019.

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

[68] S. G. Ramasubramanian, R. Venkatesan, M. Sharad, K. Roy, and A. Raghunathan, "Spindle: Spintronic deep learning engine for large-scale neuromorphic computing," in *Proceedings of the 2014 international symposium on Low power electronics and design*, ACM, 2014, pp. 15–20.

[69] S. Jain and A. Raghunathan, "Cxdnn: Hardware-software compensation methods for deep neural networks on resistive crossbar systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, p. 113, 2019.

[70] Leibin Ni, Yuhao Wang, H. Yu, Wei Yang, Chuliang Weng, and Junfeng Zhao, "An energy-efficient matrix multiplication accelerator by distributed in-memory computing on binary rram crossbar," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2016, pp. 280–285. DOI: 10.1109/ASPDAC.2016.7428024.

[71] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, 2010.

[72] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[73] X. Fong, Y. Kim, K. Yogendra, D. Fan, A. Sengupta, A. Raghunathan, and K. Roy, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 1–22, 2015.

[74] A. Sengupta and K. Roy, "Encoding neural and synaptic functionalities in electron spin: A pathway to efficient neuromorphic computing," *Applied Physics Reviews*, vol. 4, no. 4, p. 041 105, 2017.

[75] N. Yamada, E. Ohno, K. Nishiuchi, N. Akahira, and M. Takao, "Rapid-phase transitions of gete-sb2te3 pseudobinary amorphous thin films for an optical disk memory," *Journal of Applied Physics*, vol. 69, no. 5, pp. 2849–2856, 1991.

[76] H. F. Hamann, M. O'Boyle, Y. C. Martin, M. Rooks, and H. K. Wickramasinghe, "Ultra-high-density phase-change storage and memory," *Nature materials*, vol. 5, no. 5, p. 383, 2006.

[77] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *ACM SIGARCH computer architecture news*, ACM, vol. 37, 2009, pp. 14–23.

[78] Y. Watanabe, J. Bednorz, A. Bietsch, C. Gerber, D. Widmer, A. Beck, and S. Wind, "Current-driven insulator–conductor transition and nonvolatile memory in chromium-doped srtio 3 single crystals," *Applied Physics Letters*, vol. 78, no. 23, pp. 3738–3740, 2001.

[79] L. Goux, P. Czarnecki, Y. Y. Chen, L. Pantisano, X. Wang, R. Degraeve, B. Govoreanu, M. Jurczak, D. Wouters, and L. Altimime, "Evidences of oxygen-mediated resistive-switching mechanism in tin\hfo 2\pt cells," *Applied Physics Letters*, vol. 97, no. 24, p. 243 509, 2010.

[80] C. Rohde, B. J. Choi, D. S. Jeong, S. Choi, J.-S. Zhao, and C. S. Hwang, "Identification of a determining parameter for resistive switching of ti o 2 thin films," *Applied Physics Letters*, vol. 86, no. 26, p. 262 907, 2005.

[81] Z. Wei, Y. Kanzawa, K. Arita, Y. Katoh, K. Kawai, S. Muraoka, S. Mitani, S. Fujii, K. Katayama, M. Iijima, *et al.*, "Highly reliable taox reram and direct evidence of redox reaction mechanism," in *2008 IEEE International Electron Devices Meeting*, IEEE, 2008, pp. 1–4.

[82] D. Jana, S. Roy, R. Panja, M. Dutta, S. Z. Rahaman, R. Mahapatra, and S. Maikap, "Conductive-bridging random access memory: Challenges and opportunity for 3d architecture," *Nanoscale research letters*, vol. 10, no. 1, p. 188, 2015.

[83] S.-Y. Wang, C.-W. Huang, D.-Y. Lee, T.-Y. Tseng, and T.-C. Chang, "Multilevel resistive switching in ti/cu x o/pt memory devices," *Journal of Applied Physics*, vol. 108, no. 11, p. 114 110, 2010.

[84] S. Park, S. Jung, M. Siddik, M. Jo, J. Park, S. Kim, W. Lee, J. Shin, D. Lee, G. Choi, *et al.*, "Self-formed schottky barrier induced selector-less rram for cross-point memory applications," *physica status solidi (RRL)–Rapid Research Letters*, vol. 6, no. 11, pp. 454–456, 2012.

[85] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[86] K. Wang, J. Alzate, and P. K. Amiri, "Low-power non-volatile spintronic memory: Stt-ram and beyond," *Journal of Physics D: Applied Physics*, vol. 46, no. 7, p. 074 003, 2013.

[87] P. Khalili Amiri, Z. Zeng, J. Langer, H. Zhao, G. Rowlands, Y.-J. Chen, I. Krivorotov, J.-P. Wang, H. Jiang, J. Katine, *et al.*, "Switching current reduction using perpendicular anisotropy in cofeb–mgo magnetic tunnel junctions," *Applied Physics Letters*, vol. 98, no. 11, p. 112 507, 2011.

[88] J. C. Slonczewski, "Current-driven excitation of magnetic multilayers," *Journal of Magnetism and Magnetic Materials*, vol. 159, no. 1-2, pp. L1–L7, 1996.

[89] T. Kawahara, R. Takemura, K. Miura, J. Hayakawa, S. Ikeda, Y. Lee, R. Sasaki, Y. Goto, K. Ito, T. Meguro, *et al.*, "2mb spin-transfer torque ram (spram) with bit-by-bit bidirectional current write and parallelizing-direction current read," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, IEEE, 2007, pp. 480–617.

[90] S. Ikeda, J. Hayakawa, Y. Ashizawa, Y. Lee, K. Miura, H. Hasegawa, M. Tsunoda, F. Matsukura, and H. Ohno, "Tunnel magnetoresistance of 604% at 300 k by suppression of ta diffusion in co fe b/ mg o/ co fe b pseudo-spin-valves annealed at high temperature," *Applied Physics Letters*, vol. 93, no. 8, p. 082 508, 2008.

[91] J. Mathon and A. Umerski, "Theory of tunneling magnetoresistance of an epitaxial fe/mgo/fe (001) junction," *Physical Review B*, vol. 63, no. 22, p. 220 403, 2001.

[92] A. Hirohata, H. Sukegawa, H. Yanagihara, I. Žutić, T. Seki, S. Mizukami, and R. Swaminathan, "Roadmap for emerging materials for spintronic device applications," *IEEE Transactions on Magnetics*, vol. 51, no. 10, pp. 1–11, 2015.

[93] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *2013 IEEE International Symposium on Information Theory*, IEEE, 2013, pp. 156–160.

[94] L. Liu, C.-F. Pai, Y. Li, H. Tseng, D. Ralph, and R. Buhrman, "Spin-torque switching with the giant spin hall effect of tantalum," *Science*, vol. 336, no. 6081, pp. 555–558, 2012.

[95] J. Woo and S. Yu, "Impact of selector devices in analog rram-based crossbar arrays for inference and training of neuromorphic system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2205–2212, Sep. 2019, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2019.2917764.

[96] K. Gopalakrishnan, R. S. Shenoy, C. T. Rettner, K. Virwani, D. S. Bethune, R. M. Shelby, G. W. Burr, A. Kellock, R. S. King, K. Nguyen, A. N. Bowers, M. Jurich, B. Jackson, A. M. Friz, T. Topuria, P. M. Rice, and B. N. Kurdi, "Highly-scalable novel access device based on mixed ionic electronic conduction (miec) materials for high density phase change memory (pcm) arrays," in *2010 Symposium on VLSI Technology*, Jun. 2010, pp. 205–206. DOI: 10.1109/VLSIT.2010.5556229.

[97] J. Woo, D. Lee, E. Cha, S. Lee, S. Park, and H. Hwang, "Vertically stacked reram composed of a bidirectional selector and cb-ram for cross-point array applications," *IEEE Electron Device Letters*, vol. 34, no. 12, pp. 1512–1514, Dec. 2013, ISSN: 1558-0563. DOI: 10.1109/LED.2013.2285583.

[98] Wan Gee Kim, Hyun Min Lee, Beom Yong Kim, Kyoo Ho Jung, Tae Geun Seong, Seonghyun Kim, Ha Chang Jung, Hyo June Kim, Jong Hee Yoo, Hyung Dong Lee, Soo Gil Kim, Suock Chung, Kee Jeung Lee, Jung Hoon Lee, Hyeong Soo Kim, Seok Hee Lee, Jianhua Yang, Yoocharn Jeon, and R. S. Williams, "Nbo2-based low power and cost effective 1s1r switching for high density cross point reram application," in *2014 Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers*, Jun. 2014, pp. 1–2. DOI: 10.1109/VLSIT.2014.6894405.

[99] R. Midya, Z. Wang, J. Zhang, S. E. Savel'ev, C. Li, M. Rao, M. H. Jang, S. Joshi, H. Jiang, P. Lin, K. Norris, N. Ge, Q. Wu, M. Barnell, Z. Li, H. L. Xin, R. S. Williams, Q. Xia, and J. J. Yang, "Anatomy of ag/hafnia-based selectors with 1010 nonlinearity," *Advanced Materials*, vol. 29, no. 12, p. 1 604 457, 2017. DOI: 10.1002/adma.201604457. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.201604457. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201604457.

[100] Sung Hyun Jo, T. Kumar, S. Narayanan, W. D. Lu, and H. Nazarian, "3d-stackable crossbar resistive memory based on field assisted superlinear threshold (fast) selector," in *2014 IEEE International Electron Devices Meeting*, Dec. 2014, pp. 6.7.1–6.7.4. DOI: 10.1109/IEDM.2014.7046999.

[101] S. Yasuda, K. Ohba, T. Mizuguchi, H. Sei, M. Shimuta, K. Aratani, T. Shiimoto, T. Yamamoto, T. Sone, S. Nonoguchi, J. Okuno, A. Kouchiyama, W. Otsuka, and K. Tsutsui, "A cross point cu-reram with a novel ots selector for storage class memory applications," in *2017 Symposium on VLSI Technology*, Jun. 2017, T30–T31. DOI: 10.23919/VLSIT.2017.7998189.

[102] B. Razavi, "The r-2r and c-2c ladders [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 10–15, 2019, ISSN: 1943-0590. DOI: 10.1109/MSSC.2019.2922886.

[103] C. Xue, W. Chen, J. Liu, J. Li, W. Lin, W. Lin, J. Wang, W. Wei, T. Huang, T. Chang, T. Chang, H. Kao, Y. Chiu, C. Lee, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang, "Embedded 1-mb reram-based computing-in-memory macro with multibit input and weight for cnn-based ai edge processors," *IEEE Journal of Solid-State Circuits*, pp. 1–13, 2019, ISSN: 1558-173X. DOI: 10.1109/JSSC.2019.2951363.

[104] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, IEEE, 2016, pp. 1–6.

[105] B. Razavi, "The current-steering dac [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 10, no. 1, pp. 11–15, 2018, ISSN: 1943-0590. DOI: 10.1109/MSSC.2017.2771102.

[106] B. Razavi, "The transimpedance amplifier [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 1, pp. 10–97, 2019, ISSN: 1943-0590. DOI: 10.1109/MSSC.2018.2881860.

[107] B. Razavi, "A tale of two adcs: Pipelined versus sar," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 3, pp. 38–46, 2015.

[108] B. Razavi, "The flash adc [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 3, pp. 9–13, 2017.

[109] B. Murmann, "The successive approximation register adc: A versatile building block for ultra-low- power to ultra-high-speed applications," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 78–83, Apr. 2016, ISSN: 1558-1896. DOI: 10.1109/MCOM.2016.7452270.

[110] X. Sun, S. Yin, X. Peng, R. Liu, J. Seo, and S. Yu, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2018, pp. 1423–1428. DOI: 10.23919/DATE.2018.8342235.

[111] R. Mochida *et al.*, "A 4m synapses integrated analog reram based 66.5 tops/w neural-network processor with cell current controlled writing and flexible network architecture," in *2018 IEEE Symposium on VLSI Technology*, 2018, pp. 175–176.

[112] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020. DOI: 10.1038/s41586-020-1942-4.

[113] Q. Liu *et al.*, "33.2 a fully integrated analog reram based 78.4tops/w compute-in-memory chip with fully parallel mac computing," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 500–502.

[114] Q. Wang, X. Wang, S. H. Lee, F. Meng, and W. D. Lu, "A deep neural network accelerator based on tiled rram architecture," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 14.4.1–14.4.4.

[115] C. Xue *et al.*, "15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 244–246.

[116] S. Agarwal, T.-T. Quach, O. Parekh, A. H. Hsia, E. P. DeBenedictis, C. D. James, M. J. Marinella, and J. B. Aimone, "Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding," *Frontiers in neuroscience*, vol. 9, p. 484, 2016.

[117] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, S. Agarwal, M. Marinella, M. Foltin, J. P. Strachan, D. Milojicic, W.-m. Hwu, and K. Roy, "Panther: A programmable architecture for neural network training harnessing energy-efficient reram," *arXiv preprint arXiv:1912.11516*, 2019.

[118] M. J. Marinella, S. Agarwal, A. Hsia, I. Richter, R. Jacobs-Gedrim, J. Niroula, S. J. Plimpton, E. Ipek, and C. D. James, "Multiscale co-design analysis of energy, latency, area, and accuracy of a reram analog neural training accelerator," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 86–101, 2018.

[119] D. Kadetotad, Z. Xu, A. Mohanty, P.-Y. Chen, B. Lin, J. Ye, S. Vrudhula, S. Yu, Y. Cao, and J.-s. Seo, "Parallel architecture with resistive crosspoint array for dictionary learning acceleration," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 194–204, 2015.

[120] E. J. Fuller, S. T. Keene, A. Melianas, Z. Wang, S. Agarwal, Y. Li, Y. Tuchman, C. D. James, M. J. Marinella, J. J. Yang, *et al.*, "Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing," *Science*, vol. 364, no. 6440, pp. 570–574, 2019.

[121] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

[122] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, p. 52, 2018.

[123] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, pp. 1423–1428.

[124] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *2018 IEEE Symposium on VLSI Circuits*, IEEE, 2018, pp. 141–142.

[125] W. A. Wulf and S. A. McKee, "Hitting the memory wall," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, Mar. 1995. DOI: 10.1145/216585.216588. [Online]. Available: https://doi.org/10.1145%5C%2F216585.216588.

[126] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Ballcels, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "CAVIAR: A 45k neuron, 5m synapse, 12g connects/s AER hardware sensory–processing– learning–actuating system for high-speed visual object recognition and tracking," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009. DOI: 10.1109/tnn.2009.2023653. [Online]. Available: https://doi.org/10.1109%5C%2Ftnn.2009.2023653.

[127] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, Sep. 2011. DOI: 10.1109/cicc.2011.6055294. [Online]. Available: https://doi.org/10.1109%5C%2Fcicc.2011.6055294.

[128] X. Jin, M. Lujan, L. A. Plana, S. Davies, S. Temple, and S. B. Furber, "Modeling spiking neural networks on SpiNNaker," *Computing in Science & Engineering*, vol. 12, no. 5, pp. 91–97, Sep. 2010. DOI: 10.1109/mcse.2010.112. [Online]. Available: https://doi.org/10.1109%5C%2Fmcse.2010.112.

[129] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[130] H. Shiga, D. Takashima, S.-i. Shiratake, K. Hoya, T. Miyakawa, R. Ogiwara, R. Fukuda, R. Takizawa, K. Hatsuda, F. Matsuoka, *et al.*, "A 1.6 gb/s ddr2 128 mb chain feram with scalable octal bitline and sensing schemes," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 1, pp. 142–152, 2010.

[131] K. Osada, T. Kawahara, R. Takemura, N. Kitai, N. Takaura, N. Matsuzaki, K. Kurotsuchi, H. Moriya, and M. Moniwa, "Phase change ram operated with 1.5-v cmos as low cost embedded memory," in *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, IEEE, 2005, pp. 431–434.

[132] S.-S. Sheu, M.-F. Chang, K.-F. Lin, C.-W. Wu, Y.-S. Chen, P.-F. Chiu, C.-C. Kuo, Y.-S. Yang, P.-C. Chiang, W.-P. Lin, *et al.*, "A 4mb embedded slc resistive-ram macro with 7.2 ns read-write random-access time and 160ns mlc-access capability," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, IEEE, 2011, pp. 200–202.

[133] S. Chung, K.-M. Rho, S.-D. Kim, H.-J. Suh, D.-J. Kim, H.-J. Kim, S.-H. Lee, J.-H. Park, H.-M. Hwang, S.-M. Hwang, *et al.*, "Fully integrated 54nm stt-ram with the smallest bit cell dimension for high density memory application," in *Electron Devices Meeting (IEDM), 2010 IEEE International*, IEEE, 2010, pp. 12–7.

[134] A. Sengupta, P. Panda, P. Wijesinghe, Y. Kim, and K. Roy, "Magnetic tunnel junction mimics stochastic cortical spiking neurons," *Scientific Reports*, vol. 6, no. 1, Jul. 2016. DOI: 10.1038/srep30039. [Online]. Available: https://doi.org/10.1038%5C%2Fsrep30039.

[135] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015. DOI: 10.1038/nature14441. [Online]. Available: https://doi.org/10.1038%5C%2Fnature14441.

[136] C. Liu, Q. Yang, B. Yan, J. Yang, X. Du, W. Zhu, H. Jiang, Q. Wu, M. Barnell, and H. Li, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, IEEE, Jul. 2016. DOI: 10.1109/isvlsi.2016.46. [Online]. Available: https://doi.org/10.1109%5C%2Fisvlsi.2016.46.

[137] S. B. Eryilmaz, D. Kuzum, R. Jeyasingh, S. Kim, M. BrightSky, C. Lam, and H.-S. P. Wong, "Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array," *Frontiers in Neuroscience*, vol. 8, Jul. 2014. DOI: 10.3389/fnins.2014.00205. [Online]. Available: https://doi.org/10.3389%5C%2Ffnins.2014.00205.

[138] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015. DOI: 10.1016/j.neunet.2014.09.003. [Online]. Available: https://doi.org/10.1016%5C%2Fj.neunet.2014.09.003.

[139] P. Gu, B. Li, T. Tang, S. Yu, Y. Cao, Y. Wang, and H. Yang, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," in *The 20th Asia and South Pacific Design Automation Conference*, IEEE, Jan. 2015. DOI: 10.1109/aspdac.2015.7058989. [Online]. Available: https://doi.org/10.1109%5C%2Faspdac.2015.7058989.

[140] A. Sengupta and K. Roy, "A vision for all-spin neural networks: A device to system perspective," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2267–2277, Dec. 2016. DOI: 10.1109/tcsi.2016.2615312. [Online]. Available: https://doi.org/10.1109%5C%2Ftcsi.2016.2615312.

[141] S. Yu, X. Guan, and H.-S. P. Wong, "On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, monte carlo simulation, and experimental characterization," in *2011 International Electron Devices Meeting*, IEEE, Dec. 2011. DOI: 10.1109/iedm.2011.6131572. [Online]. Available: https://doi.org/10.1109%5C%2Fiedm.2011.6131572.

[142] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano Letters*, vol. 12, no. 1, pp. 389–395, Jan. 2012. DOI: 10.1021/nl203687n. [Online]. Available: https://doi.org/10.1021%5C%2Fnl203687n.

[143] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Detection, diagnosis, and repair of faults in memristor-based memories," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, IEEE, Apr. 2014. DOI: 10.1109/vts.2014.6818762. [Online]. Available: https://doi.org/10.1109%5C%2Fvts.2014.6818762.

[144] P.-Y. Chen, D. Kadetotad, Z. Xu, A. Mohanty, B. Lin, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, and S. Yu, "Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, IEEE Conference Publications, 2015. DOI: 10.7873/date.2015.0620. [Online]. Available: https://doi.org/10.7873%5C%2Fdate.2015.0620.

[145] B. Liu, H. Li, Y. Chen, X. Li, T. Huang, Q. Wu, and M. Barnell, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, Nov. 2014. DOI: 10.1109/iccad.2014.7001330. [Online]. Available: https://doi.org/10.1109%5C%2Ficcad.2014.7001330.

[146] Y. Chen, X. Li, Y. Sonnefraud, A. I. Fernández-Domínguez, X. Luo, M. Hong, and S. A. Maier, "Engineering the phase front of light with phase-change material based planar lenses," *Scientific Reports*, vol. 5, no. 1, Mar. 2015. DOI: 10.1038/srep08660. [Online]. Available: https://doi.org/10.1038%5C%2Fsrep08660.

[147] C. Liu, M. Hu, J. P. Strachan, and H. (Li, "Rescuing memristor-based neuromorphic design with high defects," in *Proceedings of the 54th Annual Design Automation Conference 2017 on - DAC 2017*, ACM Press, 2017. DOI: 10.1145/3061639.3062310. [Online]. Available: https://doi.org/10.1145%5C%2F3061639.3062310.

[148] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th ICML*, 2010, pp. 807–814.

[149] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[150] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[151] K. Jarrett, K. Kavukcuoglu, Y. LeCun, *et al.*, "What is the best multi-stage architecture for object recognition?" In *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, 2009, pp. 2146–2153.

[152] D. RUMELHART, G. HINTON, and R. WILLIAMS, "Learning internal representations by error propagation," in *Readings in Cognitive Science*, Elsevier, 1988, pp. 399–421. DOI: 10.1016/b978-1-4832-1446-7.50035-2. [Online]. Available: https://doi.org/10.1016%5C%2Fb978-1-4832-1446-7.50035-2.

[153] I.-T. Wang, Y.-C. Lin, Y.-F. Wang, C.-W. Hsu, and T.-H. Hou, "3d synaptic architecture with ultralow sub-10 fJ energy per spike for neuromorphic computation," in *2014 IEEE International Electron Devices Meeting*, IEEE, Dec. 2014. DOI: 10.1109/iedm.2014.7047127. [Online]. Available: https://doi.org/10.1109%5C%2Fiedm.2014.7047127.

[154] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nature Communications*, vol. 4, Jun. 2013. DOI: 10.1038/ncomms3072. [Online]. Available: https://doi.org/10.1038%5C%2Fncomms3072.

[155] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, Apr. 2010. DOI: 10.1021/nl904092h. [Online]. Available: https://doi.org/10.1021%5C%2Fnl904092h.

[156] A. Hirohata, H. Sukegawa, H. Yanagihara, I. Zutic, T. Seki, S. Mizukami, and R. Swaminathan, "Roadmap for emerging materials for spintronic device applications," *IEEE Transactions on Magnetics*, vol. 51, no. 10, pp. 1–11, Oct. 2015. DOI: 10.1109/tmag.2015.2457393. [Online]. Available: https://doi.org/10.1109%5C%2Ftmag.2015.2457393.

[157] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, Apr. 2010. DOI: 10.1038/nature08940. [Online]. Available: https://doi.org/10.1038%5C%2Fnature08940.

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

[159] R. Berdan, E. Vasilaki, A. Khiat, G. Indiveri, A. Serb, and T. Prodromakis, "Emulating short-term synaptic dynamics with memristive devices," *Scientific Reports*, vol. 6, no. 1, Jan. 2016. DOI: 10.1038/srep18639. [Online]. Available: https://doi.org/10.1038%5C%2Fsrep18639.

[160] K. M. Kim, J. J. Yang, J. P. Strachan, E. M. Grafals, N. Ge, N. D. Melendez, Z. Li, and R. S. Williams, "Voltage divider effect for the improvement of variability and endurance of TaOx memristor," *Scientific Reports*, vol. 6, no. 1, Feb. 2016. DOI: 10.1038/srep20085. [Online]. Available: https://doi.org/10.1038%5C%2Fsrep20085.

[161] X. Fong, S. K. Gupta, N. N. Mojumder, S. H. Choday, C. Augustine, and K. Roy, "KNACK: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque MRAM bit-cells," in *2011 International Conference on Simulation of Semiconductor Processes and Devices*, IEEE, Sep. 2011. DOI: 10.1109/sispad.2011.6035047. [Online]. Available: https://doi.org/10.1109%5C%2Fsispad.2011.6035047.

[162] R. Hecht-Nielsen *et al.*, "Theory of the backpropagation neural network.," *Neural Networks*, vol. 1, no. Supplement-1, pp. 445–448, 1988.

[163] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Technical University of Denmark*, vol. 5, 2012.

[164] A. Vedaldi and K. Lenc, "MatConvNet," in *Proceedings of the 23rd ACM international conference on Multimedia 2015*, ACM Press, 2015. DOI: 10.1145/2733373.2807412. [Online]. Available: https://doi.org/10.1145%5C%2F2733373.2807412.

[165] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Advanced Materials*, 2018. DOI: 10.1002/adma.201705914.

[166] A. Agrawal *et al.*, "X-changr: Changing memristive crossbar mapping for mitigating line-resistance induced accuracy degradation in deep neural networks," *arXiv preprint arXiv:1907.00285*, 2019.

[167] Y. Jeong *et al.*, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *IEEE Transactions on Nanotechnology*, vol. 17, no. 1, pp. 184–193, 2017.

[168] B. Liu *et al.*, "Reduction and ir-drop compensations techniques for reliable neuromorphic computing systems," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, IEEE Press, 2014, pp. 63–70.

[169] C. Liu *et al.*, "Rescuing memristor-based neuromorphic design with high defects," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 2017, pp. 1–6.

[170] B. Liu *et al.*, "Vortex: Variation-aware training for memristor x-bar," in *Proceedings of the 52nd Annual Design Automation Conference*, ACM, 2015, p. 15.

[171] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 570–579, 2019.

[172] P.-Y. Chen *et al.*, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018.

[173] A. S. Rekhi *et al.*, "Analog/mixed-signal hardware error modeling for deep learning inference," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ACM, 2019, p. 81.

[174] S. Agarwal *et al.*, "Crosssim," [Online]. Available: http://cross-sim.sandia.gov.

[175] M. A. Zidan *et al.*, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2013.

[176] X. Guan *et al.*, "A spice compact model of metal oxide resistive switching memory with variations," *IEEE electron device letters*, vol. 33, no. 10, pp. 1405–1407, 2012.

[177] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, IEEE, 2018, pp. 1–6.

[178] N. Zmora *et al.*, *Neural network distiller*, Jun. 2018. DOI: 10.5281/zenodo.1297430. [Online]. Available: https://doi.org/10.5281/zenodo.1297430.

[179] C. Xu *et al.*, "Modeling and design analysis of 3d vertical resistive memory—a low cost cross-point architecture," in *2014 19th ASP-DAC*, IEEE, 2014, pp. 825–830.

[180] S. Yu *et al.*, "A neuromorphic visual system using rram synaptic devices with sub-pj energy and tolerance to variability: Experimental characterization and large-scale modeling," in *2012 IEDM*, IEEE, 2012, pp. 10–4.

[181] J. Von Neumann, *The computer and the brain*. Yale University Press, 2012.

[182] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, p. 13, 2013.

[183] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, pp. 52–59, Dec. 2017. DOI: 10.1038/s41928-017-0002-z. [Online]. Available: https://doi.org/10.1038%5C%2Fs41928-017-0002-z.

[184] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'memristive'switches enable 'stateful'logic operations via material implication," *Nature*, vol. 464, no. 7290, p. 873, 2010.

[185] Q. Dong, S. Jeloka, M. Saligane, Y. Kim, M. Kawaminami, A. Harada, S. Miyoshi, D. Blaauw, and D. Sylvester, "A 0.3 v vddmin 4+ 2t sram for searching and in-memory computing using 55nm ddc technology," in *VLSI Circuits, 2017 Symposium on*, IEEE, 2017, pp. C160–C161.

[186] A. Agrawal, A. Jaiswal, and K. Roy, "X-sram: Enabling in-memory boolean computations in cmos static random access memories," *arXiv preprint arXiv:1712.05096*, 2017.

[187] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, IEEE Press, 2018, pp. 383–396.

[188] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb. 2018, ISSN: 0018-9200. DOI: 10.1109/JSSC.2017.2782087.

[189] J. Lee, D. Shin, Y. Kim, and H. J. Yoo, "A 17.5-fj/bit energy-efficient analog sram for mixed-signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2714–2723, Oct. 2017, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2017.2664069.

[190] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.

[191] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits*, no. 99, pp. 1–11, 2018.

[192] J. Yang, Y. Kong, Z. Wang, Y. Liu, B. Wang, S. Yin, and L. Shi, "24.4 sandwich-ram: An energy-efficient in-memory bwn architecture with pulse-width modulation," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 394–396.

[193] Z. Jiang, S. Yin, M. Seok, and J.-s. Seo, "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," in *2018 IEEE Symposium on VLSI Technology*, IEEE, 2018, pp. 173–174.

[194] R. Liu, X. Peng, X. Sun, W.-S. Khwa, X. Si, J.-J. Chen, J.-F. Li, M.-F. Chang, and S. Yu, "Parallelizing sram arrays with customized bit-cell for binary neural networks," in *Proceedings of the 55th Annual Design Automation Conference*, ACM, 2018, p. 21.

[195] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, "8t sram cell as a multi-bit dot product engine for beyond von-neumann computing," *arXiv preprint arXiv:1802.08601*, 2018.

[196] L. Chang, D. M. Fried, J. Hergenrother, J. W. Sleight, R. H. Dennard, R. K. Montoye, L. Sekaric, S. J. McNab, A. W. Topol, C. D. Adams, *et al.*, "Stable sram cell design for the 32 nm node and beyond," in *VLSI Technology, 2005. Digest of Technical Papers. 2005 Symposium on*, IEEE, 2005, pp. 128–129.

[197] L. Chang, Y. Nakamura, R. K. Montoye, J. Sawada, A. K. Martin, K. Kinoshita, F. H. Gebara, K. B. Agarwal, D. J. Acharyya, W. Haensch, *et al.*, "A 5.3 ghz 8t-sram with operation down to 0.41 v in 65nm cmos," in *VLSI Circuits, 2007 IEEE Symposium on*, IEEE, 2007, pp. 252–253.

[198] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Transactions on Electron Devices*, vol. 53, no. 11, pp. 2816–2823, 2006.

[199] G. Posser, V. Mishra, R. Reis, and S. S. Sapatnekar, "Analyzing the electromigration effects on different metal layers and different wire lengths," in *Electronics, Circuits and Systems (ICECS), 2014 21st IEEE International Conference on*, IEEE, 2014, pp. 682–685.

[200] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," *arXiv preprint arXiv:1702.06257*, 2017.

[201] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Tech. Rep., Sep. 1985. DOI: 10.21236/ada164453. [Online]. Available: https://doi.org/10.21236%5C%2Fada164453.

[202] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "Cacti 5.1," Technical Report HPL-2008-20, HP Labs, Tech. Rep., 2008.

[203] K. J. Kuhn, M. D. Giles, D. Becher, P. Kolar, A. Kornfeld, R. Kotlyar, S. T. Ma, A. Maheshwari, and S. Mudanai, "Process technology variation," *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2197–2208, 2011.

[204] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 10, pp. 1864–1878, 2014.

[205] 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 *Proceedings of the 43rd International Symposium on Computer Architecture*, IEEE Press, 2016, pp. 27–39.

[206] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP laboratories*, pp. 22–31, 2009.

[207] C.-C. Liu, S.-J. Chang, G.-Y. Huang, Y.-Z. Lin, C.-M. Huang, C.-H. Huang, L. Bu, and C.-C. Tsai, "A 10b 100ms/s 1.13 mw sar adc with binary-scaled error compensation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, IEEE, 2010, pp. 386–387.

[208] J. Su *et al.*, "A 28nm 64kb inference-training two-way transpose multibit 6t sram compute-in-memory macro for ai edge chips," in *ISSCC*, 2020. DOI: 10.1109/ISSCC19947.2020.9062949.

[209] M. Ali, A. Jaiswal, S. Kodge, A. Agrawal, I. Chakraborty, and K. Roy, "Imac: In-memory multi-bit multiplication and accumulation in 6t sram array," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 8, pp. 2521–2531, 2020.

[210] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 27–40, 2017.

[211] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2020, pp. 58–70.

[212] H. Yang, L. Duan, Y. Chen, and H. Li, "Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization," *arXiv preprint arXiv:2102.10462*, 2021.

[213] J. Yue *et al.*, "A 65nm computing-in-memory-based cnn processor with 2.9-to-35.8tops/w system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *ISSCC*, 2020. DOI: 10.1109/ISSCC19947.2020.9062958.

[214] X. Si *et al.*, "A 28nm 64kb 6t sram computing-in-memory macro with 8b mac operation for ai edge chips," in *ISSCC*, 2020. DOI: 10.1109/ISSCC19947.2020.9062995.

[215] B. Murmann, 2021. [Online]. Available: https://web.stanford.edu/~murmann/adcsurvey.html.

[216] L. Wang *et al.*, "Efficient and robust nonvolatile computing-in-memory based on voltage division in 2t2r rram with input-dependent sensing control," *IEEE TCAS-II*, vol. 68, no. 5, 2021. DOI: 10.1109/TCSII.2021.3067385.

[217] J. Yue *et al.*, "A 2.75-to-75.9tops/w computing-in-memory nn processor supporting set-associate block-wise zero skipping and ping-pong cim with simultaneous computation and weight updating," in *ISSCC*, vol. 64, 2021. DOI: 10.1109/ISSCC42613.2021.9365958.

[218] R. Guo, Y. Liu, S. Zheng, S.-Y. Wu, P. Ouyang, W.-S. Khwa, X. Chen, J.-J. Chen, X. Li, L. Liu, *et al.*, "A 5.1 pj/neuron 127.3 us/inference rnn-based speech recognition processor using 16 computing-in-memory sram macros in 65nm cmos," in *2019 Symposium on VLSI Circuits*, IEEE, 2019, pp. C120–C121.

[219] H. Jia, M. Ozatay, Y. Tang, H. Valavi, R. Pathak, J. Lee, and N. Verma, "15.1 a programmable neural-network inference accelerator based on scalable in-memory computing," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 64, 2021, pp. 236–238.

[220] J. Yue, X. Feng, Y. He, Y. Huang, Y. Wang, Z. Yuan, M. Zhan, J. Liu, J.-W. Su, Y.-L. Chung, *et al.*, "A 2.75-to-75.9 tops/w computing-in-memory nn processor supporting set-associate block-wise zero skipping and ping-pong cim with simultaneous computation and weight updating," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 64, 2021, pp. 238–240.

[221] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. DOI: 10.1038/nature14539. [Online]. Available: https://doi.org/10.1038%5C%2Fnature14539.

[222] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. DOI: 10.1038/nature16961. [Online]. Available: https://doi.org/10.1038%5C%2Fnature16961.

[223] M. Campbell, A. Hoane, and F.-h. Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 57–83, Jan. 2002. DOI: 10.1016/s0004-3702(01)00129-1. [Online]. Available: https://doi.org/10.1016%5C%2Fs0004-3702%5C%2801%5C%2900129-1.

[224] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014. DOI: 10.1126/science.1254642. [Online]. Available: https://doi.org/10.1126%5C%2Fscience.1254642.

[225] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014. DOI: 10.1109/jproc.2014.2304638. [Online]. Available: https://doi.org/10.1109%5C%2Fjproc.2014.2304638.

[226] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, IEEE, Feb. 2018. DOI: 10.1109/isscc.2018.8310397. [Online]. Available: https://doi.org/10.1109%5C%2Fisscc.2018.8310397.

[227] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, Aug. 1952. DOI: 10.1113/jphysiol.1952.sp004764. [Online]. Available: https://doi.org/10.1113%5C%2Fjphysiol.1952.sp004764.

[228] R. Stein, "Some models of neuronal variability," *Biophysical Journal*, vol. 7, no. 1, pp. 37–68, Jan. 1967. DOI: 10.1016/s0006-3495(67)86574-3. [Online]. Available: https://doi.org/10.1016%5C%2Fs0006-3495%5C%2867%5C%2986574-3.

[229] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990. DOI: 10.1109/5.58356. [Online]. Available: https://doi.org/10.1109%5C%2F5.58356.

[230] T. Tuma, A. Pantazi, M. L. Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nature Nanotechnology*, vol. 11, no. 8, pp. 693–699, May 2016. DOI: 10.1038/nnano.2016.70. [Online]. Available: https://doi.org/10.1038%5C%2Fnnano.2016.70.

[231] K. Vandoorne, P. Mechet, T. V. Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Communications*, vol. 5, no. 1, Mar. 2014. DOI: 10.1038/ncomms4541. [Online]. Available: https://doi.org/10.1038%5C%2Fncomms4541.

[232] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, and M. Soljačić, "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, vol. 11, no. 7, pp. 441–446, Jun. 2017. DOI: 10.1038/nphoton.2017.93. [Online]. Available: https://doi.org/10.1038%5C%2Fnphoton.2017.93.

[233] A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Broadcast and weight: An integrated network for scalable photonic spike processing," *Journal of Lightwave Technology*, vol. 32, no. 21, pp. 3427–3439, 2014.

[234] A. N. Tait, T. F. de Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Neuromorphic photonic networks using silicon photonic weight banks," *Scientific Reports*, vol. 7, no. 1, Aug. 2017. DOI: 10.1038/s41598-017-07754-z. [Online]. Available: https://doi.org/10.1038%5C%2Fs41598-017-07754-z.

[235] B. Rajendran, Y. Liu, J.-s. Seo, K. Gopalakrishnan, L. Chang, D. J. Friedman, and M. B. Ritter, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *IEEE Transactions on Electron Devices*, vol. 60, no. 1, pp. 246–253, 2013.

[236] M. Stegmaier, C. Rios, H. Bhaskaran, C. D. Wright, and W. H. P. Pernice, "Nonvolatile all-optical 1×2 switch for chipscale photonic networks," *Advanced Optical Materials*, vol. 5, no. 1, p. 1 600 346, Oct. 2016. DOI: 10.1002/adom.201600346. [Online]. Available: https://doi.org/10.1002%5C%2Fadom.201600346.

[237] W. H. Pernice and H. Bhaskaran, "Photonic non-volatile memories using phase change materials," *Applied Physics Letters*, vol. 101, no. 17, p. 171 101, 2012.

[238] C. Rios, M. Stegmaier, P. Hosseini, D. Wang, T. Scherer, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, "Integrated all-photonic non-volatile multi-level memory," *Nature Photonics*, vol. 9, no. 11, pp. 725–732, Sep. 2015. DOI: 10.1038/nphoton.2015.182. [Online]. Available: https://doi.org/10.1038%5C%2Fnphoton.2015.182.

[239] G. Rodriguez-Hernandez, P. Hosseini, C. Ríos, C. D. Wright, and H. Bhaskaran, "Mixed-mode electro-optical operation of ge2 sb2 te5 nanoscale crossbar devices," *Advanced Electronic Materials*, vol. 3, no. 8, p. 1 700 079, Jun. 2017. DOI: 10.1002/aelm.201700079. [Online]. Available: https://doi.org/10.1002%5C%2Faelm.201700079.

[240] Z. Cheng, C. Ríos, W. H. Pernice, C. D. Wright, and H. Bhaskaran, "On-chip photonic synapse," *Science advances*, vol. 3, no. 9, e1700160, 2017.

[241] L. Yang, R. Ji, L. Zhang, J. Ding, and Q. Xu, "On-chip cmos-compatible optical signal processor," *Optics express*, vol. 20, no. 12, pp. 13 560–13 565, 2012.

[242] N. V. Voshchinnikov, G. Videen, and T. Henning, "Effective medium theories for irregular fluffy structures: Aggregation of small particles," *Applied Optics*, vol. 46, no. 19, p. 4065, 2007. DOI: 10.1364/ao.46.004065. [Online]. Available: https://doi.org/10.1364%5C%2Fao.46.004065.

[243] H.-K. Lyeo, D. G. Cahill, B.-S. Lee, J. R. Abelson, M.-H. Kwon, K.-B. Kim, S. G. Bishop, and B.-k. Cheong, "Thermal conductivity of phase-change material $Ge_2Sb_2Te_5$," *Applied Physics Letters*, vol. 89, no. 15, p. 151 904, 2006.

[244] A. Sebastian, M. L. Gallo, and D. Krebs, "Crystal growth within a phase change memory cell," *Nature Communications*, vol. 5, Jul. 2014. DOI: 10.1038/ncomms5314. [Online]. Available: https://doi.org/10.1038%5C%2Fncomms5314.

[245] D. E. Aspnes and A. Studna, "Dielectric functions and optical parameters of Si, Ge, GaP, GaAs, GaSb, InP, InAs, and InSb from 1.5 to 6.0 ev," *Physical review B*, vol. 27, no. 2, p. 985, 1983.

[246] I. Malitson, "Interspecimen comparison of the refractive index of fused silica," *Josa*, vol. 55, no. 10, pp. 1205–1209, 1965.

[247] S.-Y. Kim, S. J. Kim, H. Seo, and M. R. Kim, "Variation of the complex refractive indices with sb-addition in ge-sb-te alloy and their wavelength dependence," in *Optical Data Storage'98*, International Society for Optics and Photonics, vol. 3401, 1998, pp. 112–116.

[248] M. L. Gallo, A. Athmanathan, D. Krebs, and A. Sebastian, "Evidence for thermally assisted threshold switching behavior in nanoscale phase-change memory cells," *Journal of Applied Physics*, vol. 119, no. 2, p. 025 704, 2016. DOI: 10.1063/1.4938532. [Online]. Available: https://doi.org/10.1063/1.4938532.

[249] W. K. Njoroge, H.-W. Wöltgens, and M. Wuttig, "Density changes upon crystallization of Ge_2 $Sb_{2.04}Te_{4.74}$ films," *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 20, no. 1, pp. 230–233, 2002. DOI: 10.1116/1.1430249. [Online]. Available: https://doi.org/10.1116/1.1430249.

[250] Comsol, *Multiphysics reference guide for comsol 4.2*, 2011. [Online]. Available: www.comsol.com.

[251] Lumerical, *Lumerical inc.* 2017. [Online]. Available: http://www.lumerical.com/tcad-products/fdtd/.

[252] *MNIST handwritten digit database*, http://yann.lecun.com/exdb/mnist/.

[253] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, Elsevier, 1992, pp. 65–93.

[254] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, IEEE, 2015, pp. 1–8.

[255] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *ACM SIGARCH Computer Architecture News*, ACM, vol. 37, 2009, pp. 2–13.

[256] J. Zheng, A. Khanolkar, P. Xu, S. Colburn, S. Deshmukh, J. Myers, J. Frantz, E. Pop, J. Hendrickson, J. Doylend, N. Boechler, and A. Majumdar, "GST-on-silicon hybrid nanophotonic integrated circuits: A non-volatile quasi-continuously reprogrammable platform," *Optical Materials Express*, vol. 8, no. 6, p. 1551, May 2018. DOI: 10.1364/ome.8.001551. [Online]. Available: https://doi.org/10.1364%5C%2Fome.8.001551.

[257] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[258] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: ImageNet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, pp. 525–542. DOI: 10.1007/978-3-319-46493-0_32.

[259] W. Bogaerts, P. De Heyn, T. Van Vaerenbergh, K. De Vos, S. Kumar Selvaraja, T. Claes, P. Dumon, P. Bienstman, D. Van Thourhout, and R. Baets, "Silicon microring resonators," *Laser & Photonics Reviews*, vol. 6, no. 1, pp. 47–73, 2012.

[260] Q. Xu, D. Fattal, and R. G. Beausoleil, "Silicon microring resonators with 1.5-μm radius," *Optics Express*, vol. 16, no. 6, p. 4309, Mar. 2008. DOI: 10.1364/oe.16.004309. [Online]. Available: https://doi.org/10.1364%5C%2Foe.16.004309.

[261] A. Ankit, A. Sengupta, P. Panda, and K. Roy, "Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ACM, 2017, p. 27.

[262] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, 2019.

[263] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[264] A. Sengupta, M. Parsa, B. Han, and K. Roy, "Probabilistic deep spiking neural systems enabled by magnetic tunnel junction," *IEEE Transactions on Electron Devices*, vol. 63, no. 7, pp. 2963–2970, 2016.

[265] J. M. Shainline, A. N. McCaughan, S. M. Buckley, C. A. Donnelly, M. Castellanos-Beltran, M. L. Schneider, R. P. Mirin, and S. W. Nam, "Superconducting optoelectronic neurons iii: Synaptic plasticity," *arXiv preprint arXiv:1805.01937*, 2018.

[266] J. M. Shainline, J. Chiles, S. M. Buckley, A. N. McCaughan, R. P. Mirin, and S. W. Nam, "Superconducting optoelectronic neurons v: Networks and scaling," *arXiv preprint arXiv:1805.01942*, 2018.

[267] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[268] D. Roy, I. Chakraborty, T. Ibrayev, and K. Roy, "Robustness hidden in plain sight: Can analog computing defend against adversarial attacks?" *arXiv e-prints*, arXiv–2008, 2020.

[269] S. Negi, I. Chakraborty, A. Ankit, and K. Roy, "Nax: Co-designing neural network and hardware architecture for memristive xbar based computing systems," *arXiv preprint arXiv:2106.12125*, 2021.

[270] Keckler *et al.*, "Gpus and the future of parallel computing," *IEEE Micro*, no. 5, pp. 7–17, 2011.

[271] *2015 ITRS 2.0 OFFICIAL PUBLICATION.zip*, 2015.

[272] D. Ielmini and Y. Zhang, "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices," *Journal of Applied Physics*, vol. 102, no. 5, p. 054 517, 2007.

[273] D. Ventrice, P. Fantini, A. Redaelli, A. Pirovano, A. Benvenuti, and F. Pellizzer, "A phase change memory compact model for multilevel applications," *IEEE Electron Device Letters*, vol. 28, no. 11, pp. 973–975, 2007.

[274] S. R. Kulkarni, D. V. Kadetotad, J.-s. Seo, and B. Rajendran, "Well-posed verilog-a compact model for phase change memory," in *2018 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, IEEE, 2018, pp. 369–373.

[275] X. Fong, S. K. Gupta, N. N. Mojumder, S. H. Choday, C. Augustine, and K. Roy, "Knack: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque mram bit-cells," in *2011 International Conference on Simulation of Semiconductor Processes and Devices*, IEEE, 2011, pp. 51–54.

[276] P.-Y. Chen, B. Lin, I. Wang, T.-H. Hou, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, S. Yu, *et al.*, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, IEEE Press, 2015, pp. 194–199.

[277] L. Gao, P.-Y. Chen, and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," *IEEE Electron Device Letters*, vol. 36, no. 11, pp. 1157–1159, 2015.

VITA

Indranil Chakraborty received his B.E. degree from Jadavpur University, India, in 2013 and the Master's degree from Indian Institute of Technology, Bombay, in 2016. His master's thesis was on physics-based modelling of PCMO-based devices. He was the recipient of best M. Tech thesis award and academic excellence award during his time at IIT Bombay for his academic performance. Currently, he is pursuing Ph.D. degree under the guidance of Prof. Kaushik Roy. His primary research interests include in-memory computing platforms based on CMOS, beyond-CMOS technologies and Si Photonics as well as hardware-software co-design for enabling edge computing systems under the broad umbrella of Artificial Intelligence. He was an intern with Intel Labs, Hillsboro, in summers of 2019 and 2020.

PUBLICATIONS

1. **I. Chakraborty** et al., "Resistive Crossbars as Approximate Hardware Building Blocks for Machine Learning: Opportunities and Challenges," in Proceedings of the IEEE, vol. 108, no. 12, pp. 2276-2310, Dec. 2020, doi: 10.1109/JPROC.2020.3003007.

2. **I. Chakraborty**, M. Fayez Ali, D. Eun Kim, A. Ankit and K. Roy, "GENIEx: A Generalized Approach to Emulating Non-Ideality in Memristive Xbars using Neural Networks," 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020, pp. 1-6, doi: 10.1109/DAC18072.2020.9218688.

3. Mustafa Ali, **I. Chakraborty**, Utkarsh Saxena, Amogh Agrawal, "A 35.5-127.2 TOP-S/W Dynamic Sparsity-Aware Reconfigurable-Precision Compute-in-Memory SRAM Macro for Machine Learning" under review in Solid State Circuit Letters.

4. A. Jaiswal*, **I.Chakraborty***, A. Agrawal, K. Roy. "8T SRAM Cell as a Multi-bit Dot Product Engine for Beyond von-Neumann Computing", in *IEEE Transactions on Very Large Scale Integrated Circuits*, 2020. (* Equal contributors)

5. **I.Chakraborty**, D. Roy, I. Garg, A. Ankit and K. Roy. "Constructing energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence", in *Nature Machine Intelligence*, 2020.

6. **I.Chakraborty**, D. Roy, and K. Roy. "Technology Aware Training in Memristive Neuromorphic Systems based on non-ideal Synaptic Crossbars", in *IEEE Transactions on Emerging Topics in Computational Intelligence* 2018.

7. **I.Chakraborty***, A. Jaiswal*, A. K. Saha, S. K. Gupta and K. Roy. "Pathways to Efficient Neuromorphic Computing with Non-Volatile Memory Technologies", in *Applied Physics Reviews*, 2020. (* Equal contributors.)

8. **I.Chakraborty**, G. Saha, A. Sengupta, and K. Roy. "Toward Fast Neural Computing using All-Photonic Phase Change Spiking Neurons", in *Scientific Reports, Nature*, 2018.

9. **I.Chakraborty**, G. Saha, and K. Roy. "A Photonic In-Memory Computing primitive for Spiking Neural Networks using Phase-Change Materials", in *Physical Review Applied* 2019.

10. **I. Chakraborty**, A. Agrawal, A. Jaiswal, G. Srinivasan and K. Roy, "In-Situ UnsupervisedLearning using StochasticSwitching in Magneto-Electric Magnetic Tunnel Junctions", in *Proceedings of the Royal Society A, IEEE*, 2020.

11. **I. Chakraborty**, A. Agarwal and K. Roy. "Design of a Low Voltage Analog-to-Digital Converter using Voltage Controlled Stochastic Switching of Low Barrier Nanomagnets", in *IEEE Magnetics Letters*, Vol. 9, pp.1-5, 2018.

12. S. Jain, A. Ankit, **I. Chakraborty**, K.Roy, A. Raghunathan et al, "Neural network accelerator design with resistive crossbars: Opportunities and challenges", in *IBM Journal of Research and Development*, vol. 63, no. 6, pp. 10:1-10:13, 1 Nov.-Dec. 2019.

13. A. Jaiswal, **I. Chakraborty**, K. Roy. "Energy-Efficient Memories using Magneto-Electric Switching of Ferromagnets", in *IEEE Magnetics Letters*, 8, pp.1-5, 2017.

14. P. Panda, **I. Chakraborty**, K. Roy, "Discretization based Solutions for Secure Machine Learning against Adversarial Attacks", in *IEEE Access* 2019.