2021

Energy-Efficient In-Memory Architectures Leveraging Intrinsic Behaviors of Embedded MRAM Devices

Shadi Sheikhfaal
University of Central Florida

ENERGY-EFFICIENT IN-MEMORY ARCHITECTURES LEVERAGING INTRINSIC BEHAVIORS OF EMBEDDED MRAM DEVICES

by

SHADI SHEIKHFAAL
B.S. Azad University, Ardebil Branch, 2012
M.S. Azad University, Science and Research Branch, 2014

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2021

Major Professor: Ronald F. DeMara

ABSTRACT

For decades, innovations to surmount the processor versus memory gap and move beyond conventional von Neumann architectures continue to be sought and explored. Recent machine learning models still expend orders of magnitude more time and energy to access data in memory in addition to merely performing the computation itself. This phenomenon referred to as a memory-wall bottleneck, is addressed herein via a completely fresh perspective on logic and memory technology design. The specific solutions developed in this dissertation focus on utilizing intrinsic switching behaviors of embedded MRAM devices to design cross-layer and energy-efficient Compute-in-Memory (CiM) architectures, accelerate the computationally-intensive operations in various Artificial Neural Networks (ANNs), achieve higher density and reduce the power consumption as crucial requirements in future Internet of Things (IoT) devices.

The first cross-layer platform developed herein is an Approximate Generative Adversarial Network (ApGAN) designed to accelerate the Generative Adversarial Networks from both algorithm and hardware implementation perspectives. In addition to binarizing the weights, further reduction in storage and computation resources is achieved by leveraging an in-memory addition scheme. Moreover, a memristor-based CiM accelerator for ApGAN is developed. The second design is a biologically-inspired memory architecture. The Short-Term Memory and Long-Term Memory features in biology are realized in hardware via a beyond-CMOS-based learning approach derived from the repeated input information and retrieval of the encoded data. The third cross-layer architecture is a programmable energy-efficient hardware implementation for Recurrent Neural Network with ultra-low power, area-efficient spin-based activation functions. A novel CiM

architecture is proposed to leverage data-level parallelism during the evaluation phase. Specifically, we employ an MRAM-based Adjustable Probabilistic Activation Function (APAF) via a low-power tunable activation mechanism, providing adjustable accuracy levels to mimic ideal sigmoid and tanh thresholding along with a matching algorithm to regulate neuronal properties. Finally, the APAF design is utilized in the Long Short-Term Memory (LSTM) network to evaluate the network performance using binary and non-binary activation functions. The simulation results indicate up to 74.5× energy-efficiency, 35-fold speedup and ~11× area reduction compared with the similar baseline designs. These can form basis for future post-CMOS based non-Von Neumann architectures suitable for intermittently powered energy harvesting devices capable of pushing intelligence towards the edge of computing network.

Dedicated to my dear family for their selfless support.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. DeMara who provided me an opportunity to join the Computer Architecture Laboratory (CAL) research team and guided me through my research under his supervision. He has kindly supported me during this project, and his insightful comments helped me to proceed my research along the right direction leading to several publications in prestigious journals. I would also like to thank my committee members, Dr. Kalpathy Sundaram, Dr. Azadeh Vosoughi, Dr. Jun Wang, and Dr. Vikram Kapoor for their support.

Furthermore, this work was supported in part by the Center for Probabilistic Spin Logic for Low-Energy Boolean and Non-Boolean Computing (CAPSL), one of the Nanoelectronic Computing Research (nCORE) Centers as task 2759.006, a Semiconductor Research Corporation (SRC) program sponsored by the NSF through CCF 1739635. This work was also partly sponsored by the NSF through HRD 1953606.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 : INTRODUCTION

1.1 Research Motivation

With notable advancements in complementary metal-oxide-semiconductor (CMOS) technology, the feature size of these charge-based Field-Effect Transistors (FETs) has scaled down and the number of transistors on integrated circuits has doubled nearly every two years as predicted by the Moor's law [1]. The scaling (miniaturization) of CMOS technology has provided enhanced chip performance at a reduced cost through the increase of transistor density and switching speed, as the main objective of silicon technology for decades [2]. However, the scale down of this technology is reaching to nano ranges, where it exceeds the required spacing for the quantum mechanical tunneling of electrons leading to the well-known leakage challenge. Additionally, there are other critical challenges in the CMOS technology scaling such as high leakage currents, high power density, limited gate control, higher circuit noise sensitivity and increased lithography costs. On the other hand, with the convergence of multiple technologies such as embedded systems, machine learning, and cloud computing, Internet-of-Things (IoT) devices have evolved into the most popular and growing technology in the recent decade. IoTs employ self-sufficient ambient-powered circuits with small area overhead, which provide intermittent operations, low-power data acquisition and processing capabilities while maintaining a low cost [3, 4]. Moreover, due to the limited energy budget and challenges caused by the device scaling, achieving energy-efficient and high-performance computing is one of the main objectives within IoT applications. These challenges have motivated the research towards designing hybrid and novel energy-efficient circuits by combining the mature CMOS technology with emerging technologies such as

Spintronics [5, 6]. Spintronic technology is specifically compatible with the CMOS technology as a result of the possibility of 3D integration at the back-end process, which is able to merge the logic and memory and reduce the dynamic power. The main features of Spintronic devices that make them a suitable candidate for the next-generation hybrid technologies are non-volatility, reduced area overhead or high integration density, and near-zero static power. Moreover, the non-volatility feature significantly reduces the standby power as it can maintain the data while the power is off. These features can be leveraged to develop area-efficient digital circuits with instant store/restore functionality for power gating purposes in intermittent computations, designing arrays of non-volatile memory and novel activation function units for neuromorphic computational architectures, and most importantly designing energy-efficient circuits and architectures for IoT devices [7].

1.2 Need for Energy-Efficient Machine Learning Architectures

In the last few years, with advancements in technology and increasing production rates of electronic companies, the number of the edge devices such as smartphones, laptops, and other IoT devices are increasing significantly and it is expected to have billions of connected devices generating vast amount of raw data [8]. Drastically-reduced energy consumption is one of the main objectives in designing next generation IoT devices such that these devices are able to operate using only ambient sources of light, kinetic, thermal, and electromagnetic energy and achieve battery-free computing [9]. On the other hand, machine learning methods have drawn great

Table 1.1: Energy consumption of various operations in 45nm CMOS processor [13].

Operations	Energy (pJ)	Relative Energy Cost
32-bit integer addition	0.1	1
32-bit floating-point addition	0.9	9
32-bit integer multiplication	3.1	31
32-bit floating-point multiplication	3.7	37
32-bit SRAM Access	5	50
32-bit DRAM Access	640	6400

attention and have achieved notable advancements in various domains such as computer vision, image recognition, speech recognition, machine translation and etc. [10]. To achieve higher accuracy levels in various Deep Neural Networks (DNNs) applications such as image classification as a subset of big data, larger model sizes and higher computing workload are required. Typically, for running a DNN on an IoT device, the process of inference is performed on the cloud. However, executing the inference on the edge device itself is gaining more attention as it reduces the latency, enhances privacy, and moderates the execution time [11, 12]. On the other hand, edge devices have limited on-chip cache memory capacity (typically <10 Mb) and high-performance models must be located in the off-chip main memory [8]. There are several algorithms for pre-processing big data, which typically run on general purpose conventional processors. However, von Neumann processing architectures cannot process big data efficiently due to the high demand of data movement between separated processing and memory units, referred to as memory bottleneck. Studies show that Dynamic Random-Access Memory (DRAM) read operation in a 32-bit system consumes orders higher energy than a 32-bit floating point multiplication compared to the on-chip operations as depicted in Table 1.1 [13].

In order to overcome the aforementioned constraints and challenges as shown in Figure 1.1, Compute-in-Memory (CiM) architectures have been proposed to eliminate the high energy

consumption and memory access latency by regulating data movement [14-17]. CiM architectures employ the analog characteristics of emerging non-volatile memory devices to provide in-place computations. This method is especially influential in designing DNN accelerators, which demand computationally expensive operations such as multiplication. On the other hand, non-volatile memory devices such as Magnetic Tunnel Junction (MTJ) and Resistive Random-Access Memory (ReRAM) can provide the required characteristics for future low-power computational IoT edge devices. Hence, this dissertation focuses on designing energy efficient cross-layer CiM architectures leveraging customized in-memory algorithms for various DNNs and exploits the intrinsic behaviors of high/low energy barrier Magnetic Random-Access Memories (MRAMs) to achieve yet more efficiency. Figure 1.1 shows the research motivations in this dissertation. Considering the challenges in this field of research and with focusing on application characteristics of the IoT devices, this dissertation aims to design energy-efficient, high performance, low area overhead acceleration designs utilizing neuromorphic computing, cross-layer evaluations and digital CiM frameworks.

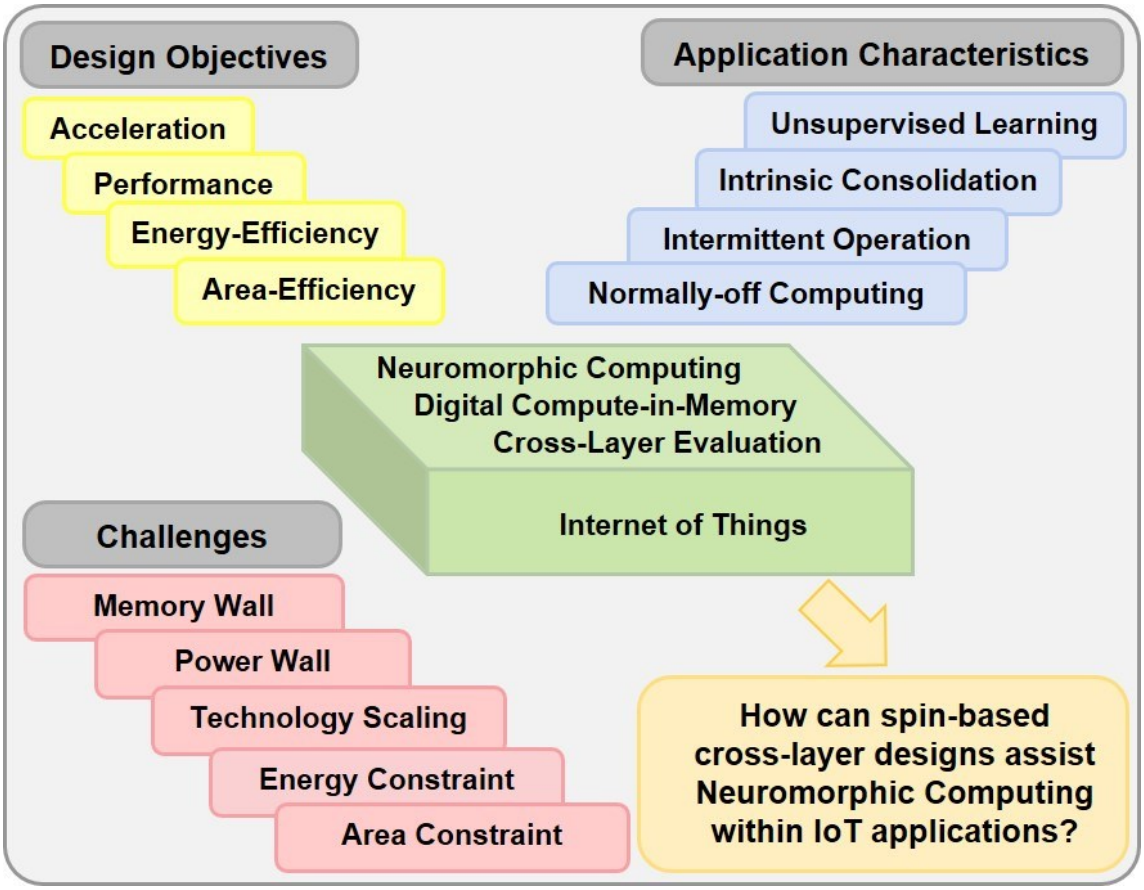


Figure 1.1: Research motivation and objective.

1.3 Contribution of the Dissertation

In consequence of the motivations, this dissertation focuses on designing energy efficient cross layer CiM architectures leveraging customized in-memory algorithms for various DNNs and exploits the intrinsic behaviors of high/low energy barrier Magnetic Random-Access Memories (MRAMs) to achieve yet more efficiency as shown in Figure 1.2. The main focus of this dissertation is to develop a cross-layer framework, starting from device/circuit to architecture and

application, which provides customized algorithms to guarantee the high performance. In summary, the major contributions in this dissertation can be listed as follows:

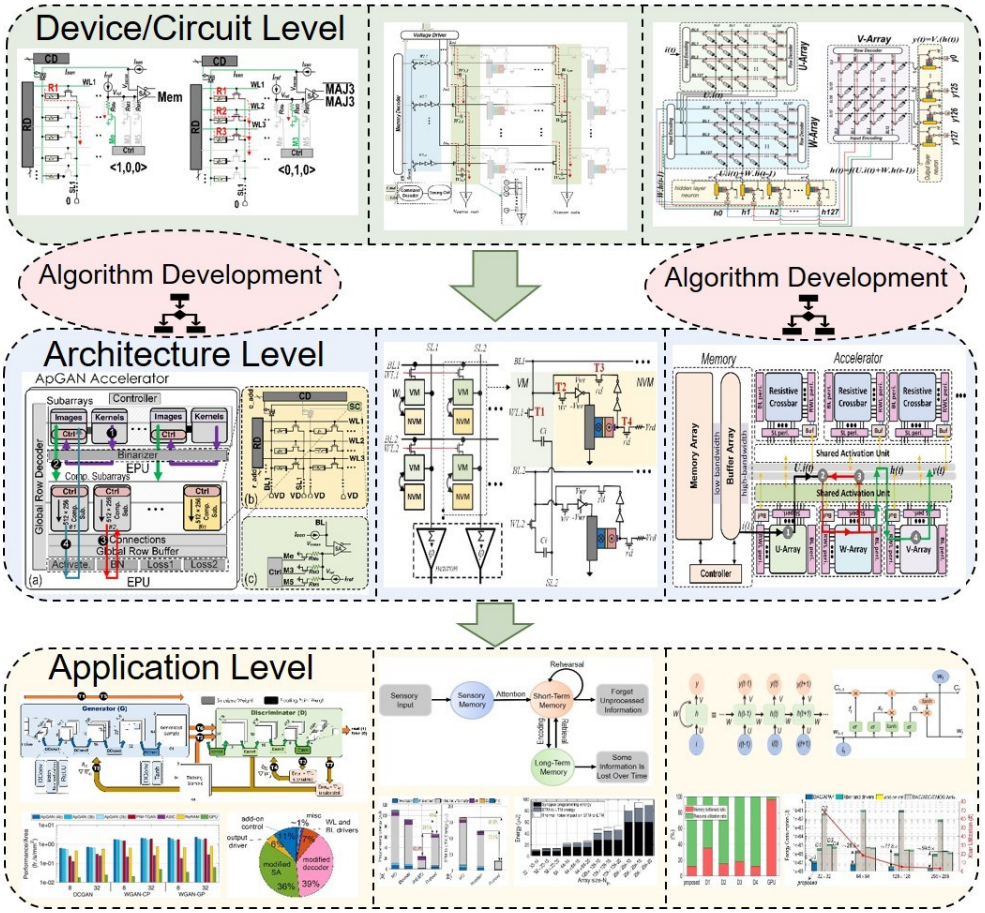


Figure 1.2: Cross-layer research flow.

- In the first cross-layer design, an Approximate Generative Adversarial Network (ApGAN) is developed. A GAN is an adversarial learning approach which empowers conventional deep learning methods by alleviating the demands of massive labeled datasets. However, GAN training can be computationally-intensive limiting its feasibility in resource-limited edge devices. In this chapter, we propose an approximate GAN (ApGAN) for accelerating

GANs from both algorithm and hardware implementation perspectives. First, inspired by the binary pattern feature extraction method along with binarized representation entropy, the existing Deep Convolutional GAN (DCGAN) algorithm is modified by binarizing the weights for a specific portion of layers within both the generator and discriminator models. Further reduction in storage and computation resources is achieved by leveraging a novel hardware-configurable in-memory addition scheme, which can operate in the accurate and approximate modes. Finally, a memristor-based processing-in-memory accelerator for ApGAN is developed. The performance of the ApGAN accelerator on different data-sets such as Fashion-MNIST, CIFAR-10, STL-10, and celeb-A is evaluated and compared with recent GAN accelerator designs. With almost the same Inception Score (IS) to the baseline GAN, the ApGAN accelerator can increase the energy-efficiency by ~28.6× achieving 35-fold speedup compared with a baseline GPU platform. Additionally, it shows 2.5× and 5.8× higher energy-efficiency and speedup over CMOS-ASIC accelerator subject to an 11 percent reduction in IS [18].

- The second cross-layer design is a biologically inspired Short-Term Long-Term Memory architecture. Biological memory structures impart enormous retention capacity while automatically pro-viding vital functions for chronological information management and update the resolution of the domain and episodic knowledge. A crucial requirement for hardware realization of such cortical operations found in biology is to first design both short-term memory (STM) and long-term memory (LTM). Herein, these memory features are realized via a beyond-CMOS-based learning approach derived from the repeated input information and retrieval of the encoded data. We first propose a new binary STM-LTM

architecture with composite synapse of the Spin Hall Effect-driven Magnetic Tunnel Junction (SHE-MTJ) and capacitive memory bit cell to mimic the behavior of biological synapses. This STM-LTM platform realizes the memory potentiation through a continual update process using STM-to-LTM transfer, which is applied to neural networks based on the established capacitive crossbar. We then propose a hardware-enabled and customized STM-LTM transition algorithm for the platform considering the real hardware parameters. We validate the functionality of the design using SPICE simulations that show the proposed synapse has the potential of reaching ~30.2 pJ energy consumption for STM-to-LTM transfer and 65 pJ during STM programming. We further analyze the correlation between energy, array size, and STM-to-LTM threshold utilizing the MNIST data set [8].

- The third cross-layer CiM architecture is a customized design for Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. As research in RNNs continue preeminent algorithmic refinements, the use of conventional hardware structures requires higher energy and latency to process the sophisticated computations. Herein, we develop a programmable energy-efficient hardware implementation for RNNs and LSTMs with Resistive Random-Access Memory (ReRAM) synapses and ultra-low power, area-efficient spin-based activation functions. To attain high energy-efficiency while maintaining accuracy, a novel Computing-in-Memory (CiM) architecture is proposed to leverage data-level parallelism during the evaluation phase. Specifically, we employ an MRAM-based Adjustable Probabilistic Activation Function (APAF) via a low-power tunable activation mechanism, providing adjustable levels of accuracy to mimic ideal sigmoid and tanh thresholding along with a matching algorithm to regulate the neuron

properties. To evaluate the performance of the proposed design, we present a hardware/software cross-layer framework. The simulations show that our proposed design achieves up to 74.5× energy-efficiency with ~11× area reduction compared to its counterpart designs while keeping the accuracy comparable with the baseline designs. We also have examined the performance of an LSTM network for name prediction purposes utilizing ideal, binary, and the proposed non-binary APAF based neuron. The comparison of the results shows that our proposed neuron can achieve up to 85% accuracy and perplexity of 1.56, which attains performance similar to algorithmic expectations of near-ideal neurons. The simulations show that our proposed neuron achieves up to 34-fold improvement in energy efficiency and 2-fold area reduction compared to the CMOS-based non-binary designs.

CHAPTER 2 : BACKGROUND

Magnetic tunnel junction (MTJ) devices are the building block of any spin-based structure, which can be configured into two different resistant levels as parallel (P) and anti-parallel (AP) states. As a result of the tunnel magnetoresistance (TMR) effect, the P and AP states show high and low resistance, denoting “1” and “0” in binary, respectively. There are two different switching approaches originated for MTJs as Spin-Transfer Torque (STT) [19] and Spin-Orbit Torque (SOT) [20], in which only one bidirectional ultra-small current is required. Recent fabrications and experiments of nano-magnets show that switching the magnetization can be achieved with high speed (sub-nanosecond), below fJ/bit memory write energy, and long endurance (10 years). In the STT switching approach, the bidirectional current passes through an MTJ resulting in either AP or P state. STT provides several improvements compared to the previous switching methods such as field-induced magnetic switching (FIMS) [21] and thermally assisted switching (TAS) [22]. However, this method is affected by some challenges in its overall functionality such as switching asymmetry, high write current [23, 24] and a shared read and write path. Accordingly, during read operation, malfunctions such as unwanted switching may appear, that can flip the stored data unintentionally. However, SHE-MTJ as a potential alternative to STT-MTJ has been investigated, which is a 3-terminal device, offering advantages including separated read and write paths, higher energy efficiency and higher write speed [25-27].

2.1 Spintronic Devices

Spintronics is a relatively novel computing paradigm that utilizes the spin of electrons as the state variable for computation by means of spin-polarized current [28]. There are two stable polarizations for the spin-based devices as 0° denoting up-spin, and 180° denoting down-spin magnet spin momentum. The state of the device is retained in a magnet with no constant electrical power requirement due to its non-volatility coming from the energy barrier (E_B). The correlation between the energy barrier and the information retention time is expressed by Equation 2.1:

$$T_{retention} = T_0 \, e^{(\frac{E_B}{K_B T})}, \; E_B = K_u V \tag{2.1}$$

where K_B is the Boltzmann's constant, T is the temperature, T_0 is the characteristic time, K_u is the magnetic anisotropy, and V is the magnet volume. The energy barrier (E_B) in majority of the spin-based memory and logic realizations, is set to 40 resulting in ten years of retention time ($T_{retention}$). Figure 2.1 shows the two stable states (0° and 180°), and the unstable (90°) state referred to in the following sections as probabilistic state, with regards to the E_B [29].

The spin polarization and magnetoresistance are the main characteristics of the Spintronics which are employed to perform read and write operations, respectively. The spin population can be defined as the imbalance of up-spin (n↑) and down-spin (n↓) numbers in ferromagnetic (FM) devices, which is defined as:

$$P = \frac{|n\uparrow - n\downarrow|}{n\uparrow + n\downarrow} \tag{2.2}$$

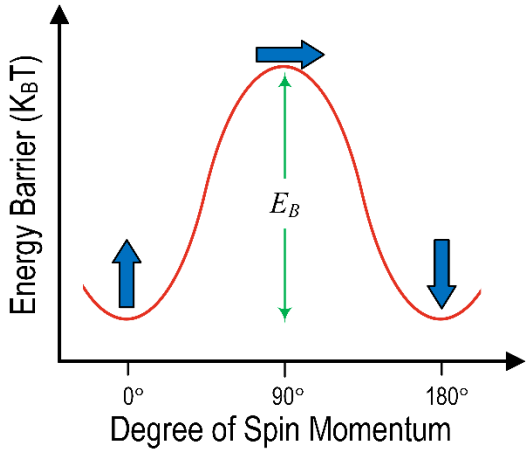


Figure 2.1: Spin momentum and the energy barrier of a nanomagnet.

A charge current passing through a ferromagnet becomes polarized corresponding to the local magnetic momentum, which consequently, outputs a spin-polarized current. The distributing of the electrons on the ferromagnetic layers identifies the magnetoresistance. To sense the states of magnetic devices, high or low magnetoresistance (MR) are utilized for magnetic materials. In the metal multilayer films Giant Magnetoresistance (GMR) [30, 31] and Tunneling Magnetoresistance (TMR) [32, 33] are the most employed MR effects. The GMR devices are fabricated from two ferromagnetic layers, sandwiching a thin layer of metal such as copper. In the parallel configuration case with similar magnetization directions of two FM layers, the spin-down or spin-up electrons pass through the device with no scattering contributing to a lower resistance. On the other hand, for the anti-parallel configuration (AP) with opposite magnetization directions of FM layers, both spin-down and spin-up electrons will have scattering condition resulting in higher

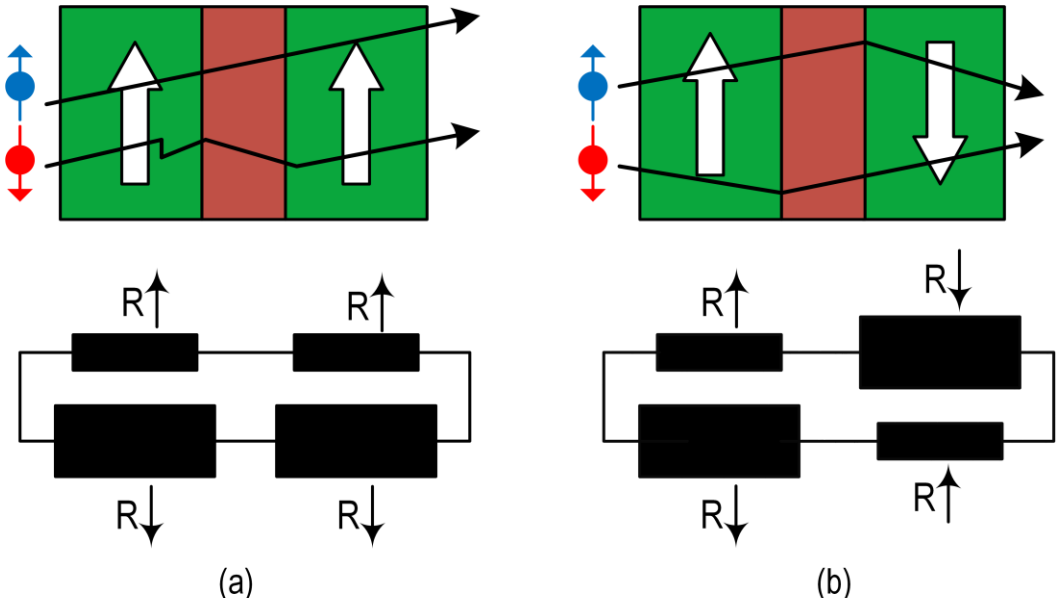


Figure 2.2: (a) GMR effect in parallel state, and (b) GMR effect in anti-parallel state [29].

resistance. One of the commonly used GMR-based applications is the spin valve model, leveraged as reading heads in conventional hard disk drives. Figure 2.2 illustrates the GMR effect in two-channel multilayer films. TMR effect can be detected, if the non-magnetic layer in a GMR structure is replaced by a thin oxide insulator such as MgO [34], and AlxOy [35]. The thickness of this spacer is designed to allow the tunneling effect for the electrons. Figure 2.3 demonstrations a TMR device with its two stable states. Similar to the GMR effect, TMR can define AP magnetization orientation by high and P magnetization orientation by low resistance. Nevertheless, there are two main differences that set the two devices apart in addition to the barrier material difference for GMR and TMR devices.

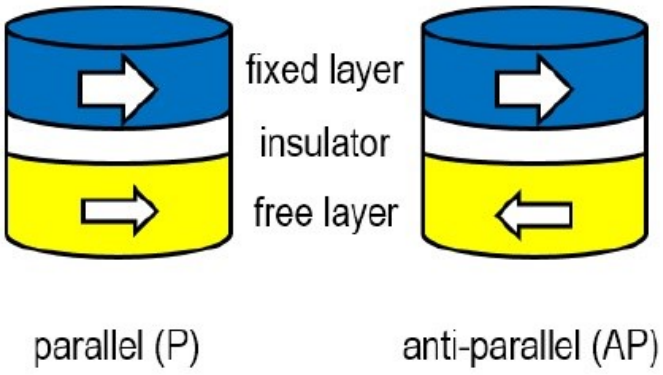


Figure 2.3: A TMR device in anti-parallel and parallel configurations.

The first difference is in the GMR structure. In this case, current flows in both "perpendicular to plane" (CPP), or "in the layer plane" (CIP) [36]. However, in TMR, current only flows in a perpendicular way. The second difference is that in GMR, all the layers are conductors, which leads to larger current transfer. On the other hand, TMR devices have insulators which is preferable in logic and non-volatile memory designs.

2.1.1 Magnetic Tunnel Junction (MTJ)

Figure 2.4 (a) shows the vertical structure of an MTJ, where two FM layers i.e., Free Layer (FL) and Pinned Layer (PL) with distinct coercivities, sandwich a thin oxide barrier, e.g. MgO [37]. The magnetization orientation of the pinned layer is fixed magnetically and is used as the reference layer. However, the magnetization of the free layer can be switched to be anti-parallel or parallel

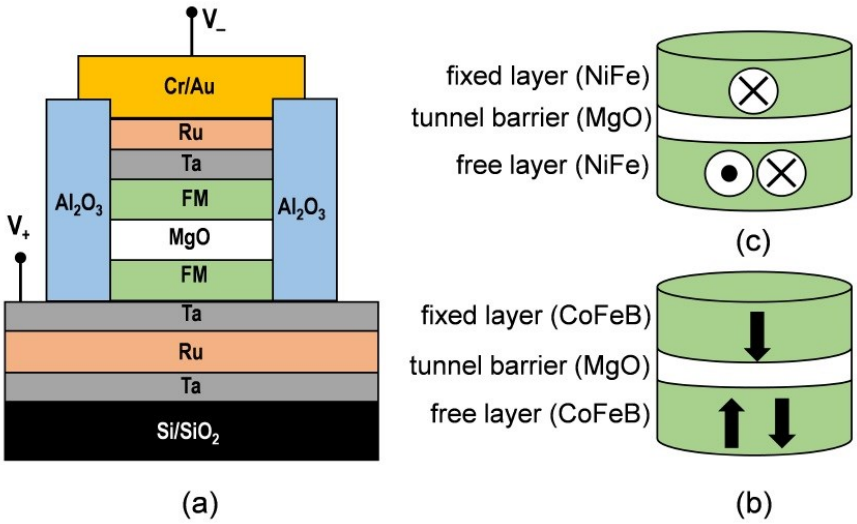


Figure 2.4: (a) vertical view of the MTJ structure [20], (b) In-plane MTJ (IMTJ), and (c) Perpendicular MTJ (PMTJ) [29].

to the pinned layer orientation as depicted in Figure 2.4 (b). The resistance of the MTJ is referred to as tunneling magnetoresistance (TMR) [37] and the TMR ration determines the performance of an MTJ as defined below:

$$TMR = \frac{\Delta R}{R_P} = \frac{R_{AP} - R_P}{R_P} = \frac{G_P - G_{AP}}{G_{AP}} \tag{2.3}$$

where G_P and G_{AP} are the conductance of anti-parallel and parallel states. The conductance expressions are given by:

$$G_P = N_{M1}N_{M2} + N_{m1}N_{m2}$$

$$G_{AP} = N_{M1}N_{m2} + N_{m1}N_{M2} \tag{2.4}$$

where N_{M1} and N_{m1} are the effective densities of states of majority and minority electrons at the Fermi energy in both magnetic layers. As a result, the TMR ratio can be calculated using Equations

2.2, 2.3 and 2.4, which is expressed in terms of the spin polarization by:

$$TMR = \frac{2P_1P_2}{1-P_1P_2} = \begin{cases} R_P = \frac{2}{1+P_1P_2} \\ R_{AP} = \frac{2}{1-P_1P_2} \end{cases} \quad (2.5)$$

where P_1 and P_2 are spin-polarizations of each layer. For the tunneling barrier design, amorphous Al_2O_3 was first utilized in 1994 to achieve a room temperature magnetic tunneling transport [32, 38]. The TMR ratio of such design can reach up to 70% by enhancing the fabrication and material conditions [35]. However, spintronic applications such as MRAMs still require a minimum of %150 TMR at the room temperature, regardless of the fact that 70% TMR is a huge improvement compared to the spin valve GMR. One of the other improvements in MTJ is utilizing a single-crystal MgO tunnel barrier providing larger TMR, referred to as the giant TMR effect [39, 40]. The recent experiments on the TMR ratio have reached to a 600% at room temperature [41].

As shown in Figure 2.4 (b) and 2.4(c), the magnetic direction of MTJ layers can be out of the film plane or in the film plane indicated as perpendicular MTJ (PMA) and in-plane MTJ (IMA) structure, respectively. Nevertheless, PMAs are more preferable due to their improvements over IMAs including higher thermal stability and lower switching critical current [9].

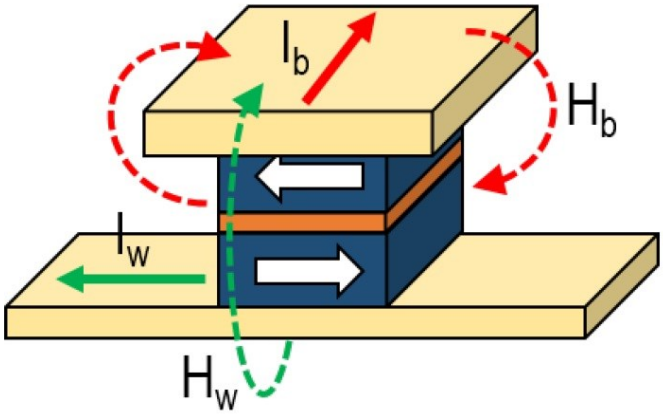


Figure 2.5: Magnetic field switching approach for MTJ [29].

2.1.2 Magnetic Field Switching

The write operation of an MTJ is achieved by switching the FL magnetization orientation. In the magnetic field switching approach, an external magnetic field which is generated by two orthogonal current lines, the word line (WL) and bit line (BL), is applied to switch the free layer magnetization orientation as shown in Figure 2.5. For performing the write operation, I_w and I_b currents, are applied to BL and WL, generating the easy-axis H_b, and the hard-axis H_w switching fields, respectively. Here, H_w corresponds to $2K_u/M_s$ where M_s is the saturation magnetization and is applied to the easy axis perpendicularly. Next, this field is replaced by a smaller bias field applied along the easy axis to finish the switching process. By passing the current through BL, we can accomplish the read operation. One of the advantages of this approach is the separate read and write paths. Nevertheless, in the write operation, the narrow write margin and half-selectivity issues are a result of the combination of two perpendicular currents. Additionally, to perform an accurate write operation in magnetic field switching approach, generating the needed magnetic fields involves high currents of ~10 mA, which limits the scalability of this approach as a result of

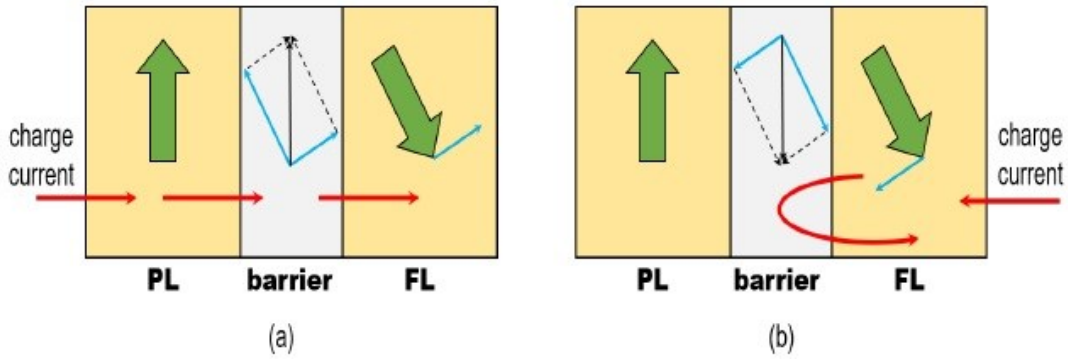


Figure 2.6: Spin filtering effect in STT , (a) electrons flowing from the pinned layer to free layer, switch the nanomagnet to parallel state, and (b) electrons flowing from the free layer to pinned layer, switch the nanomagnet to anti-parallel state [29].

the electromigration effect. To solve this problem, several solutions have been proposed [42]. Though, the magnetic field switching still endures large area overhead, high power consumption, and low speed.

2.1.3 Spin Transfer Torque (STT) Switching

In the Spin Transfer Torque (STT) switching approach, a bidirectional spin-polarized current is required for switching MTJ nanomagnet configuration. The spin-polarized current is generated by a spin-polarizer. Electrons flowing from the pinned layer to the free layer are spin-polarized by the pinned layer and obtain a spin angular momentum that is approximately aligned to the magnetization orientation of the pinned layer. This process is referred to as the *filtering effect* as shown in Figure 2.6 Next, the spin-polarized electrons proceed into the free layer, where their opposite sign torque with equal magnitude must be transferred to the free layer magnetization as a result of the conservation of angular momentum. When the number of electrons surpasses the critical current as the threshold value, the spin-transfer torque (STT) employed by the current will switch the magnetization of the free layer regarding the pinned layer. When the charge current is

applied through the opposite direction, the obtained spin-polarization will be opposite the pinned layer magnetization by the reflection from the free layer, which in turn switches the nano magnet to anti-parallel state.

In the STT switching approach, the free layer magnetization is theorized by a unit vector named magnetic moment $\vec{m}$ under the macrospin approximation. The magnetization switching dynamics are described by a Landau-Lifshitz-Gilbert (LLG) equation [43], as below:

$$\frac{\partial \vec{m}}{\partial t} = -\gamma \mu_0 \vec{m} \times \overrightarrow{H_{eff}} + a\left(\vec{m} \times \frac{d\vec{m}}{dt}\right) - \frac{\gamma \hbar JP}{2et_{ox}M_s}\vec{m} \times (\vec{m} \times \vec{m}_r) \tag{2.6}$$

where $\overrightarrow{H_{eff}}$ is the effective magnetic field, which is the summation of various magnetic fields such as the external magnetic field, the anisotropy field, and the demagnetization field. γ is the gyromagnetic ratio. μ_0 is the permeability in the free space. a is the Gilbert damping constant. $\hbar$ is the reduced Planck constant, P is the spin-polarization, e is the elementary charge, t_{ox} is the FL thickness, M_s is the saturation magnetization, $\vec{m}$ is the unit vector along the pinned layer magnetization, and J is the write current density. Figure 2.7 shows the three torques presented in Equation 2.6. [44, 45]. The field-induced torque is the first torque that initiates the magnetic moment to process in presence of the effective magnetic field. The second torque is the Gilbert damping torque which eases the precession. Finally, the third torque is the STT, which is proportionate to the density of the charge current and because of the polarity of applied current, it can help or resist the Gilbert damping torque. For example, for the injected current densities larger

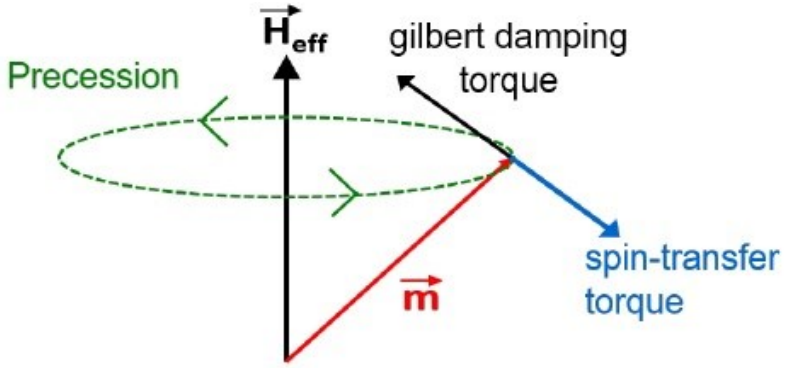


Figure 2.7: The dynamics of a nanomagnet under the spin transfer torque impact.

than the critical current density, the employed STT can make up for the Gilbert damping torque and switches the free layer magnetization orientation. As a result of the straightforwardness of STT implementation, scalability, lower read energy, and higher read speed compared to the magnetic field switching and thermally-assisted switching approaches, it has developed into the principal switching approach for the two-terminal Spintronics devices including GMR [36, 46] and TMR devices [47, 48]. In this method, a single shared path is used for both write and read operations. This can result in the unintentional write operation during the read operation. Additionally, as a result of the pre-switching oscillation [23, 49] a substantial incubation delay imposes high switching energy. Consequently, as an alternative method, the Spin-Hall Effect (SHE) method has been proposed for 3-terminal spin-based TMR devices, which offers separate read and write paths, while spending a notably less switching energy [26, 27, 50].

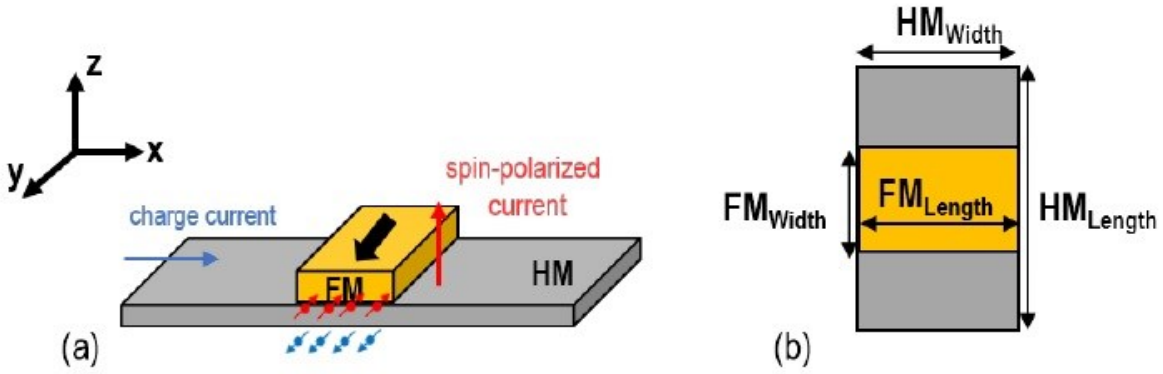


Figure 2.8: (a) A positive current in the +x direction generates a spin current in the +z direction.The applied spin current generates the needed spin torque for adjusting the magnetic direction of the FM in +y direction, (b) Top view [29].

2.1.4. Spin Hall Effect (SHE) Switching

The research in [51] confirms that the Spin-Hall Effect (SHE) in nanomagnetic devices, can generate a spin-polarized current used to create torque, instead of passing charge current through a ferromagnet in spin polarizer approach. The Spin-Hall Effect method is shown in Figure 2.8.

A SHE-MTJ is a 3-terminal device, with isolated paths for write and read operations with lower switching energy compared to STT-MTJs. It consists of a Heavy Metal (HM) nanowire beneath an MTJ with two ferromagnetic layers, called the pinned and free layers, separated by a thin oxide barrier [52]. The MTJ free layer has two different magnetization orientations, called parallel (P) and antiparallel (AP), that provide two different levels of resistance for this device. The HM can be made of β-tungsten (β-W) or β-tantalum (β-Ta) [53] with different electrical characteristics. Due to the higher positive Spin Hall angle achieved with tungsten [53], we modeled our device with this material. In order to store the data in the SHE-MTJ, the free-layer magnetization should be manipulated. This is accomplished by injecting a charge current (Ic) to HM in the +x (/ −x)

direction as shown in Figure 2.9 (a). Due to spin Hall effect, Ic will cause an accumulation of oppositely-directed spin vectors on both surfaces of the HM that then generate a spin current (Is) and further a Spin-Orbit Torque (SOT) in +y (/ −y) direction. The spin current will change the magnetization configuration of the free layer in the ±z direction according to the direction of the charge current [54]. The Spin Hall injection efficiency (P_{SHE}) can be expressed as:

$$P_{SHE} = \frac{I_s}{I_C} = \theta_{SH} \frac{A_{FM}}{A_{HM}} \left(1 - \operatorname{sech}\left(\frac{t_{HM}}{\lambda_{sf}}\right)\right) \tag{2.7}$$

where A_{FM} and A_{HM} denote the adjacent free layer area and the cross-sectional area of HM, respectively. In Equation (2.7), θ_{SH} represents the spin Hall angle, as the ratio of generated spin current density to the charge current density. Also, t_{HM} and λ_{sf} denote the thickness of HM substrate and the spin flip length, respectively [27]. If the right portion of the Equation 2.7 is greater than 1, then the spin-polarized current is larger than the charge current. As a result of the difference in scattering ratio of electrons at the heavy metal and ferromagnet interface, the spin-transfer efficiency in ferromagnet is lower than heavy metal. Thus, the P_{SHE} is larger than 1, which shows high efficiency [8].

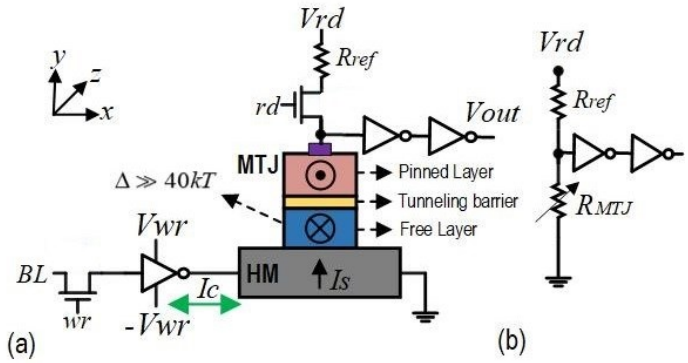


Figure 2.9: (a) Structure of a SHE-MTJ, (b) Resistive equivalent read circuit of SHE-MTJ.

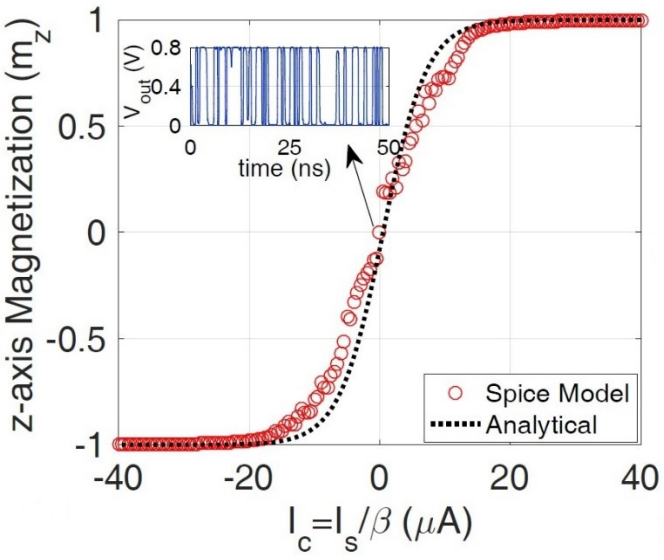


Figure 2.10: Time-averaged behavior of the SHE-MTJ based p-bit device showing the magnetization fluctuations.

2.1.5 Probabilistic Spintronic Device (p-bit)

The structure of the p-bit device is the same as Figure 2.9 (a), which consists of a Spin Hall Effect Magnetic Tunnel Junction (SHE-MTJ) with a circular unstable (low energy barrier) nanomagnet ($\Delta \ll 40kT$) [20, 55], whereby the output is amplified by two CMOS inverters. SHE-MTJ-based p-bit is a 3-terminal device, with separated read (*rd*) and write (*wr*) paths [27, 56]. It consists of an unstable MTJ with two ferromagnetic layers as *pinned layer* and *free layer*, separated by a thin oxide barrier on top of a Heavy Metal (HM) nanowire [52] made of β-tungsten (β-W) or β-tantalum (β-Ta) similar to the SHE-MTJ device discussed in the previous subsection [53]. The *pinned* layer is a stable nanomagnet with a fixed orientation whereas the free layer of the MTJ can be oriented as *parallel (P)* and *antiparallel (AP)*, providing two levels of resistance. As shown in Figure 2.10 (a), the resistance level can be manipulated by injecting a charge current (I_c) to the HM in the +x (/−x) direction [57, 58]. This charge current will initiate the accumulation of oppositely directed

spin vectors on each surface of the HM, which produces a spin current (I_s) and further a Spin-Orbit Torque (SOT) in +y (/−y) direction. Corresponding to the direction of the charge current, the spin current will change the magnetization configuration of the free layer in the ±z direction [54]. By taking a long-time average of magnetization fluctuations, the spin-current driven low energy barrier nanomagnet provides the sigmoidal function due to its intrinsic physics. Figure 2.9 (b) shows an equivalent read circuit of a SHE-MTJ based p-bit. To read the data, a small read voltage is applied to the MTJ ($V+$ and $V-$ terminals) to sense its resistance (R_{MTJ}). Then, a resistive voltage divider is realized through the R_{MTJ} and the reference resistor R0. The reference resistor is set to the MTJ average conductance ($R_0^-1 = \mathrm{GP} + \mathrm{GAP}/2$) where GP and GAP are the parallel (P) and anti-parallel (AP) state conductance. The corresponding voltage is fed to the input of the CMOS inverters which are adjusted to their middle point of DC operation. Thus, the output voltage (V_{out}) will stochastically fluctuate between "0" and "1", whereas the probability of either value is regulated by the input charge current [59]. The p-bit device generates a stochastic output under a behavior analogous to the sigmoid activation function, whose steady-state probability is modulated by an input current. For example, if the input current is a large positive number, the stochastic output of this device will be "0" with a high probability. However, if there is no input current, the output will randomly fluctuate between "0" and "1" with an equal probability of 0.5.

The device features are derived from the experimentally benchmarked models in [60] and the circuit simulations have been performed using SPICE platform. We are aiming to define the time-averaged behavior of the output as an analytical approximation. First, we link the flowing charge current in the spin Hall layer to the spin-current absorbed by the magnet. For simplicity, we assume short-circuit conditions, namely 100% spin absorption by the FM as expressed in the Equation 2.7.

By choosing an appropriate quantity for the A_{FM} and A_{HM}, the generated spin-current can be greater in magnitude than the "gain" generated by the charge current. The gain factor P_{SHE} for the parameters used herein as listed in Table 2.1, is ~ 10. Accordingly, a function of input spin-current polarized in the (±z) is used to estimate the magnetization behavior. Analytically, a distribution function for a magnet at steady state with a Perpendicular Magnetic Anisotropy (PMA) and spin-current in the ±z direction, can be written as below:

$$\rho(m_z) = \frac{1}{Z}\exp\left(\Delta m_z^2 + 2i_s m_z\right) \tag{2.8}$$

m_z being the magnetization along $+z$ direction, Z a constant for normalization, Δ the nanomagnet thermal barrier, and i_s the spin-current normalization quantity, which can be described as $i_s = I_s/(\frac{4q}{\hbar\alpha kT})$, where q is the electron charge, α is the magnets' damping coefficient, and $\hbar$ the reduced Planck constant. An average magnetization can be achieved using the Eq. (4) as follows: $< m_z > = \int_{-1}^{+1} dm_z\, m_z\rho(m_z)/\int_{-1}^{+1} dm_z\, \rho(m_z)$. Since $\Delta \ll kT$, the Langevin function $< m_z >= L(i_s)$ is realized by $< m_z >$ where $L(x) = \frac{1}{x} - coth\frac{1}{x}$. For a low energy barrier PMA magnet, this demonstrates an accurate average magnetization description around a z-directed spin-current [59]. However, in this work, we cannot obtain a simple analytical formula, as the p-bit device nanomagnet has a strong in-plane anisotropy with a circular shape. Consequently, we adjust the normalization current by a factor η, using a fitting parameter in the Langevin function, in a way that the adjusted normalization constant is converted to $(4q/\hbar\alpha kT\,)(\eta)$. With raising the shape anisotropy $(H_d \sim 4\pi Ms)$, this factor increases and becomes equal to "1" without a shape anisotropy. When the charge currents and the magnetization are connected, the CMOS inverter

output probability can be approximated by a phenomenological equation in addition to fitting parameter χ as follows: $p = \frac{V_{OUT}}{V_{DD}} \approx \frac{1}{2}[1 - tanh(\chi < m_z >)]$. This equation can be used to connect the output probability with the input charge current, by physical parameters. An evaluation of the Spice model and the aforementioned analytical equivalences is shown in Figure 2.10. This confirms the agreement of η with the magnetization, and χ with CMOS components [59].

2.2 Explored Neural Networks

This dissertation analyzes three distinct artificial neural networks: Generative Adversarial Networks (GANs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory networks (LSTMs) as described in the following subsections.

Table 2.1: p-bit device parameters.

Parameter	**Value**
Saturation Magnetization, M_s	300 emu/cm^3
Circular FM Diameter, ϕ	100 nm
Circular FM Thickness, t_{FM}	2 nm
Gilbert Damping Factor, α	0.01
Spin Diffusion Length, λ_{sf}	2.1 nm
Spin Hall Angle, θ_{SHM}	0.5
SHM Dimension $W_{SHM} \times L_{SHM} \times T_{SHM}$	100 × 100 × 3.15 nm^3
Spin Polarization, P	0.52
Conductance, G0	150 μS
Spin Hall Resistivity, ρ	200 μΩ-cm
Temperature, T	300 K

2.2.1 Generative Adversarial Networks

Compared to conventional CNN topologies, realization of Deep convolutional GAN (DCGAN) [61] implementations have several constraints: a) the strided convolutions and fractional-strided convolutions on D and G, respectively, are utilized instead of the pooling layers; b) Although in the last layer of both D and G models, Sigmoid and tanh activations are highly used, in the other layers of G and D models, *ReLU* and *LeakyReLU* activations are utilized, respectively; and c) batch normalization is leveraged on both D and G models to stabilize the training process.

DCGANs are composed of two learning subnetworks, a generator (G) as a deconvolutional neural network and a discriminator (D) as a CNN. Usually, these are developed as Deep Neural Networks (DNNs), which are trained simultaneously. Despite traditional unsupervised learning techniques, in GAN, feature representations can be learned from raw data, which results in higher accuracy. The generator learning model can be optimized to produce deceptive samples to fool the discriminator, whereas the discriminator learning model is trained in a way to distinguish the real samples from the artificial ones. The entire process is similar to a 2-player minimax game, which is expressed by:

$$min_G\ max_D\ V(D,G) = E_{x\sim P_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D\big(G(z)\big))], \tag{2.9}$$

where $P_{data}(x)$ is the distribution of data and z is the noise vector. By leveraging minibatch of data samples from D and fake images from G, we minimize $V(D,G)$ regarding G by assuming fixed D and maximize it regarding elements of D by assuming fixed G. Due to the nature of zero-sum game, each of D and G models try to improve their performance, finding a Nash equilibrium point [62], in a non-cooperative manner, which in turn causes several issues like no guarantee for

convergence. Some of the most recent and promising advancements in GAN training algorithms are Wasserstein GAN (WGAN) [63], WGAN with weight clipping (WGANCP) [64], and WGAN with gradient penalty (WGAN-GP) [64] leveraging modified loss functions. WGAN algorithm uses Wasserstein distance as a quantitative scheme to measure the distance between two probability distributions. Further improvements can be achieved by limiting the trained weights of D in a certain range in WGAN-CP and utilizing gradient penalty in WGAN-GP training algorithms.

Although GAN, particularly DCGAN, can be considered as a dominant algorithm for unsupervised learning technique, which is useful for self-learning IoT nodes [65], its deconvolution/convolutional layers occupy the largest portion of running time and consume significant computational resources, which is crucial for IoT nodes. Therefore, herein we focus on developing an optimized in-memory accelerator for both types of layers via algorithm and hardware codesign approach.

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have demonstrated notable achievements in machine learning applications involving classification, speech recognition, machine translation, and static image processing due to their ability to accumulate the effects of the input data over time [66]. As a group of Artificial Neural Networks (ANNs) focusing on sequential data, RNNs are based on a recurrent path of the information flow as shown in Figure 2.11 However, unlike feedforward ANNs, the output of RNNs depend both on current input and the previous computation results. Thus, the feedback, as a crucial and unique component, provides the memory to capture the computed information in RNNs [66].

In RNNs, as shown in Figure 2.11, a directed graph is shaped along temporal sequences with a connection between its nodes. The input vectors ($i(t)$) are fed into the network one at a time during forward propagation, regulated towards the neurons in the hidden layer. The states of the hidden neurons are updated upon arrival of the input vectors and corresponding synapse weights. The updated neuron state is retained for use upon arrival of subsequent input patterns. With arrival of a new input vector at the proceeding time step, the neurons in the hidden layer compute a new state vector based on the new input vector and the retained state vector [67]. Assuming that the W matrix in Figure 2.11 represents the recurring feedback synapses matrix in the hidden layer, Equation (2.10) and Equation (2.11) can give a mathematical representation of RNN updating the neuron state over time:

$$h(t) = f(U.i(t) + W.h(t-1) + b_h) \tag{2.10}$$

$$y(t) = V.h(t) \tag{2.11}$$

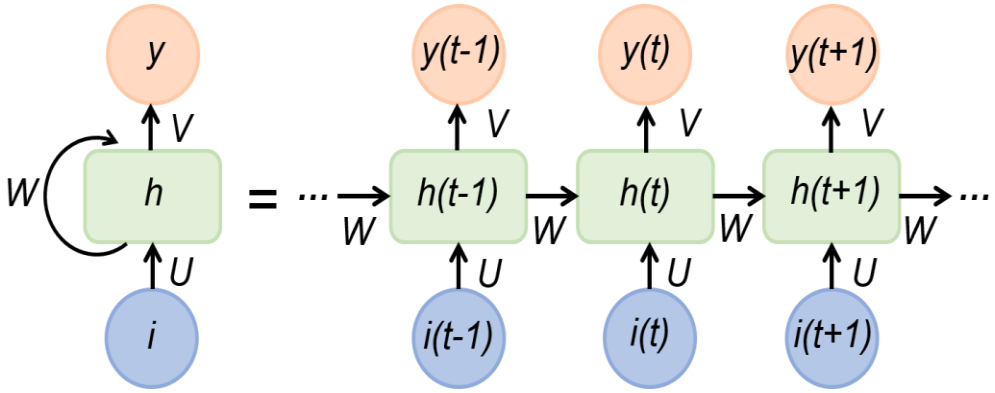


Figure 2.11: Folded and unfolded RNN structures.

where $h(t)$ represents the hidden neuron state and $y(t)$ denotes the output neurons state at time step t. f is the activation function in the hidden layer. U and V both denote the feedforward synapse

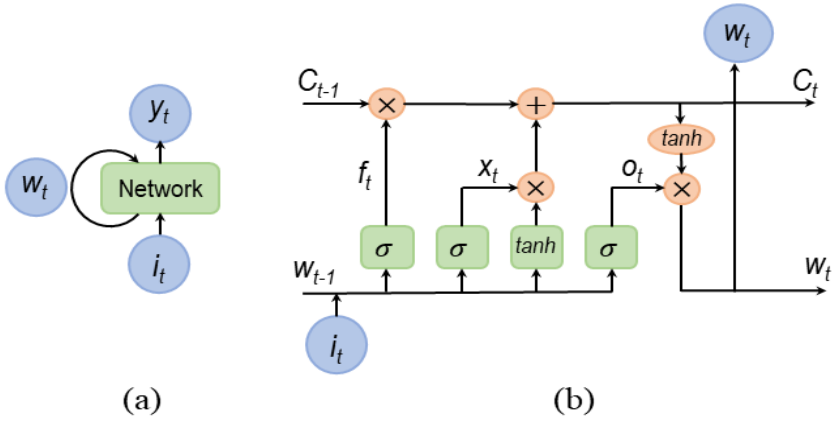


Figure 2.12: (a) Basic RNN structure, (b) LSTM.

matrices, where U holds the synapses from the input layer to the hidden layer and V represents the synapses from the hidden layer to the output layer. Finally, b_h denotes the bias in the neurons of the hidden layer. The synaptic weights and the bias vectors are initialized before training based on the network implementation [67].

2.2.3 Long Short-Term Memory (LSTM) Networks

LSTM is a specific type of RNN that is designed to overcome some of the drawbacks of the RNNs. Figure 2.12 (a) indicates the basic RNN structure. RNN output depends on both the current sample (i_t) and the previously calculated network state (w_t) as the network input. Unlike ANN, RNN has a feedback loop which gives RNN the capability to store the previous states and make future decision based on the previous values. The computational equations of a basic RNN cell are given below:

$$w_t = tanh\,(i_t U + w_{t-1} W + bias) \tag{2.12}$$

$$y_t = softmax(w_t V) \tag{2.13}$$

where i_t , w_t , and y_t are the current input, hidden state, and output for the current input, respectively; V, W, and U contain trainable parameter matrices. With the feedback loop RNN is expected to handle long-term dependencies but this is not true when it comes for practical application. RNN can't handle long-term dependencies in practice due to the vanishing gradient problem [68].

LSTM is a special kind of RNN which tries to solve the problem of vanishing gradients that we encounter during the backpropagation technique of neural networks [69]. Figure 2.12 (b) indicates an LSTM cell which contains three gates: input gate x_t, forget gate f_t , and output gate o_t . The forget gate decides which information from the previous cell state to be preserved and which must be forgotten. This decision is taken using a sigmoid layer which gives output between 0 and 1 [70]. The input gate decides which of the new cell contents are to be written to the cell state. It has two parts- the sigmoid layer decides which values of input (concatenation of new input values and output values from previous states) to update and the tanh layer generates a vector of new candidate values. The output gate decides which content of the cell to output based on given inputs and previous state values. The output vector is obtained by multiplying a new cell state which is normalized to values between -1 to 1 using tanh activation function and output of sigmoid layer that decides which part of cell state and given to output. The dimensions of all the gates is same as the dimensions of hidden state. The computational equations of LSTM are given below:

$$x = \boldsymbol{\sigma}(i_t U^x + w_{t-1} W^x + b_x) \tag{2.14}$$

$$f = \boldsymbol{\sigma}\big(i_t U^f + w_{t-1} W^f + b_f\big) \tag{2.15}$$

$$o = \boldsymbol{\sigma}(i_t U^o + w_{t-1} W^o + b_o) \tag{2.16}$$

$$g = tanh\left(i_t U^g + w_{t-1} W^g + b_g\right) \quad (2.17)$$

$$c_t = c_{t-1} \odot f + g \odot x \quad (2.18)$$

$$w_t = tanh(c_t) \odot o. \quad (2.19)$$

Three main operation types can be observed from the above equations: nonlinear functions (sigmoid **σ** and hyperbolic tangent $tanh$), matrix-vector multiplication (e.g., $w_{t-1} W^x$ and $i_t U^x$), and element-wise multiplication (e.g., $g \odot x$) [71].

LSTM Activation Functions: The conventional activation functions used in an LSTM are sigmoid or logistic-sigmoid and hyperbolic tangent in short $tanh$ activation functions. A sigmoid activation function alters any input value to value between 0 and 1. Similarly a $tanh$ activation function alters any input value to value between -1 and 1 [72]. This will help to allow or not allow the flow of information through the LSTM gates. The equations of these functions are given below:

$$\sigma\,(x) = 1/(1 + exp(-x)) \quad (2.20)$$

$$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \quad (2.21)$$

Where, x is the input, $\sigma\,(x)$ is the sigmoid function, and $\tanh(x)$ is the hyperbolic tangent function.

CHAPTER 3 : APPROXIMATE GENERATIVE ADVERSERIAL NETWORK (APGAN)

3.1. Fundamentals of Generative Adversarial Networks

Recently, deep Convolutional Neural Networks (CNNs) [73] have shown impressive performance for computer vision, e.g., image recognition tasks, achieving close to human-level perception rates. These neural network models are usually trained using a supervised approach, which limits scalability due to the requirement for large-scale labeled datasets. The processing demands of high-depth CNNs spanning hundreds of layers face serious challenges for their tractability in terms of memory and computation resources and because of so-called "CNN power and memory wall" phenomena, conventional processing platforms such as CPU cannot perform this training step. This has been motivating the development of alternative approaches in both SW/HW domains to improve conventional CNN efficiency.

In algorithm-based approaches, use of quantizing parameters [74], and network binarization [75] have been explored extensively to eliminate the need for intensive Multiplication-And-Accumulate (MAC) operations. Recently, utilizing weights with low bit-width and activations reduces both model size and computing complexity [75]. For instance, performing bit-wise convolution between the inputs and low bit-width weights has been demonstrated in [75] by converting conventional MAC operations into their corresponding AND bit count operations. Meanwhile to improve computing efficiency of CNNs from the hardware point of view extensive studies for developing deep learning accelerators using GPUs and FPGAs have been researched. However, within conventional isolated computing units and memory elements interconnected via

buses, there are serious challenges, such as limited memory bandwidth channels, long memory access latency, significant congestion at I/O chokepoints, and high leakage power consumption [76, 77].

Processing-in-Memory (PIM) paradigms built on top of non-volatile devices, such as Resistive Random Access Memory (ReRAM) [78, 79], Magnetic RAM (MRAM) [80-85], and Phase Change Memory (PCM) [86] have been introduced to address the aforementioned concerns, such as memory bottlenecks and high leakage power dissipation that has become increasingly prominent with technology scaling. Due to the interesting features of Non-Volatile Memory (NVM) technology such as near-zero standby power, high integration density, compatibility with CMOS fabrication processes, and radiation-hardness, they offer some promising attributes for in-memory processing implementations including the realization of logic functions along with an inherent state-holding capability [87-89].

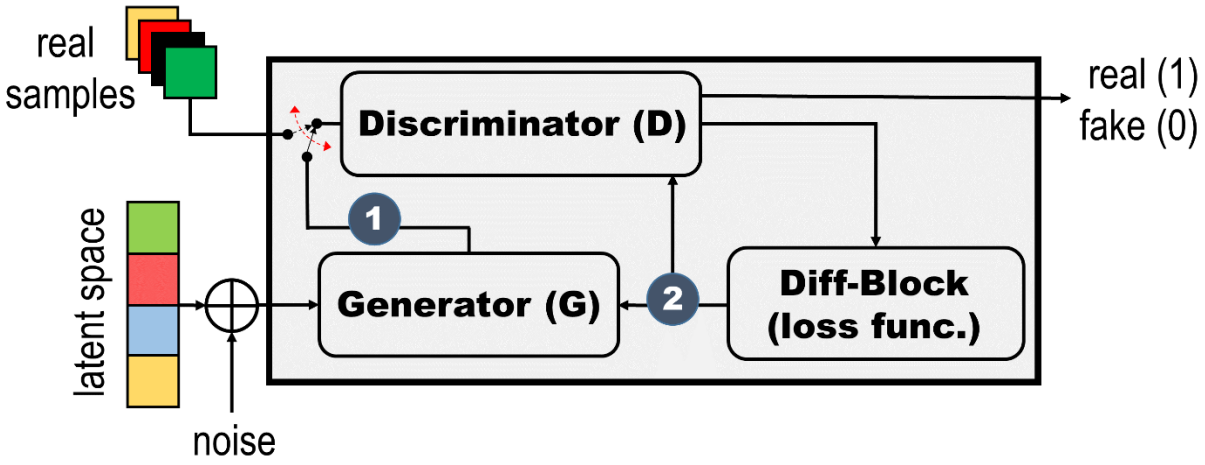


Figure 3.1: GAN structure. D downsamples the input data, while G is given a uniform noise distribution to generate fake samples (1) In (2), fine-tuning of training is performed.

Due to the abovementioned challenges, semi-supervised and unsupervised learning models, such as the Generative Adversarial Network (GAN) algorithm [90], especially Deep convolutional GANs (DCGANs) [61], are of increasing interest. The DCGAN architecture is composed of two separate models. A discriminator model (D) that estimates the probability of a given sample being legitimate or counterfeit. It is trained as a detective to discern between fake samples and real ones. Whereas, the other model, known as the generator (G), samples a uniform random noise input and also captures the real data distribution to generate images as real as possible to deceive the discriminator, as shown in Figure 3.1. Basically, this realizes a zero-sum game between the two models. Based on the GAN structure, two training processes, i.e., consisting of four forward and four backward passes are required, which are more sophisticated than CNN training with one forward pass and one backward pass. Therefore, implementing an efficient accelerator for GAN using the existing designs for energy and area-constrained IoT nodes, is vital but challenging.

In this section, to make GAN suitable for resource-limited edge devices, the advancements from both algorithm and hardware architecture perspectives to efficiently accelerate GAN training are deployed. The existing GAN algorithm is modified by replacing the multiplications in convolution layers in the generator (G) model and in the discriminator (D) model, with less complex and more efficient subtraction and addition.

3.2. Approximate GAN (ApGAN) Architecture

Figure 3.2 depicts the general architecture for our deep convolutional-based Approximate GAN (ApGAN), which consists of four deconvolution and four convolution layers for generator (G) and discriminator (D), respectively. In this section, first, the training procedure of ApGAN is analyzed

with respect to the partially-quantized layers. Afterwards, we introduce the method of partial approximate computing to further improvement at the cost of lower accuracy.

3.3 ApGAN Training

Since discriminator units are developed similar to conventional CNNs, all the proposed compression techniques such as quantization and pruning can be applied in the same way. However, due to the deconvolution process in G, local to global mapping instead of the global to local mapping process in D, leveraging these techniques have negative effects on the developed compression methods. On the other hand, as mentioned previously, GAN consumes massive computational power for the training phase, in which two distinct D and G models should be trained separately but simultaneously.

Therefore, to enhance the efficiency of training and facilitate hardware mapping, a novel training approach including partially-quantized layers, i.e., weight binarization, and modification of the loss function presented in [91], is introduced. Herein, both D and G networks are trained using

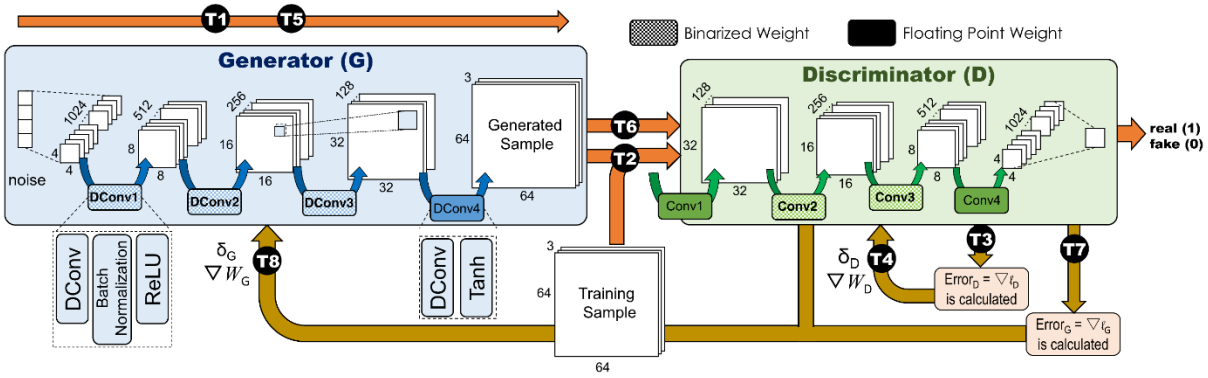


Figure 3.2: Approximate GAN system and its training loop from (T1) to (T8).

binarized weights (-1, +1), which results in the elimination of the computationally expensive multiplication operations. ApGAN training includes a) forward computation, computation phase, and b) backpropagation, update phase. After producing a series of fake samples by generator (T1), both real and fake samples are imported into the D network (T2). Next, regarding the output layer of D, the error is calculated based on the gradient of the loss function (T3). Then (T4) starts by feeding the error back into D. After passing the error to each layer of D, the weight of D is updated. Updating the G network starts by importing artificial sample (T5) into D (T6). The loss for training G is then computed (T7) and back-propagated to G (T8) to update its weights.

The eight-step training process can be summarized into three main phases, which are operating sequentially in an iterative manner: (I) weight binarization and statistical weight scaling, (II) binary weight-based inference to compute the loss function and (III) back propagation to update full precision weights. In (I), current full precision weights are binarized by only taking the sign function, expressed in Equation (3.1) and then the corresponding scaling factor will be computed based on the current statistical distribution of full precision weight.

$$Forward: \quad b = sgn(y) = \begin{cases} +1, & if\ y \geq 0 \\ -1, & otherwise \end{cases} \tag{3.1}$$

In this case, the sign function is non-convex, which results in the gradient becoming zero. Thus, a standard backpropagation approach will be impractical due to the vanishing gradient problem. Several studies have performed to make the sign function smooth by developing continuation methods such as *softsign* [92] and *appsign* [93], in which the original complex problem is split into several problems that can be optimized easier by reducing the smoothing rate steadily. Herein, due to similar observations between *appsign*(.) and *tanh*(.) functions also ease of implementation

of tanh activation function in hardware perspective, Equation (3.2) is considered in the forward path.

$$appsign(y) = \begin{cases} +1, & if\ y \geq 0 \\ y, & if\ 1 \geq y \geq -1 \\ -1, & if\ y \leq -1 \end{cases}$$

$$Forward: sign(y) = \lim_{\beta \to \infty} appsign\left(\beta_y\right) \approx \lim_{\beta \to \infty} \tanh\ (\beta_y) \tag{3.2}$$

In order to achieve a good binary representation, we use the modified Binarized Representation Entropy (BRE) regularization [91] to boost the variety of binary columns in the low-dimensional layer [92]. The BRE is calculated over a mini-batch of $X = \{x_1, x_2, \dots, x_k\}$ including two terms, marginal entropy (ME), and modified activation correlation [91].

$$l_{ME} = \frac{1}{d}\sum_{j=1}^{d}\left(\frac{1}{K}\sum_{k=1}^{K}(s_k, j)\right)^2$$

$$l_{MAC} = \sum_{j,k=1, j\neq k}^{N} \frac{\alpha_{k,j}}{\sum_{j,k=1, j\neq k}^{N} \alpha_{k,j}} . \frac{\left|S_{f,j}^{T}.S_{f,k}\right|}{d} \tag{3.3}$$

where s_k is the activation vector of $x \in X$, while the large parenthesis denotes the average of jth element of the s_k. Letter $\alpha_{k,j}$ are weights regarding $S_{f,j}^{T}.S_{f,k}$ pairs, and the sum inthe denominator is defined as a normalization constant. Therefore in (II), the input mini-batch takes the binarized model for inference and the loss function of the discriminator will be calculated, which can be expressed as follow:

$$L = \lambda_1 . l_D + \lambda_2 . l_{ME} + \lambda_3 . l_{MAC} \tag{3.4}$$

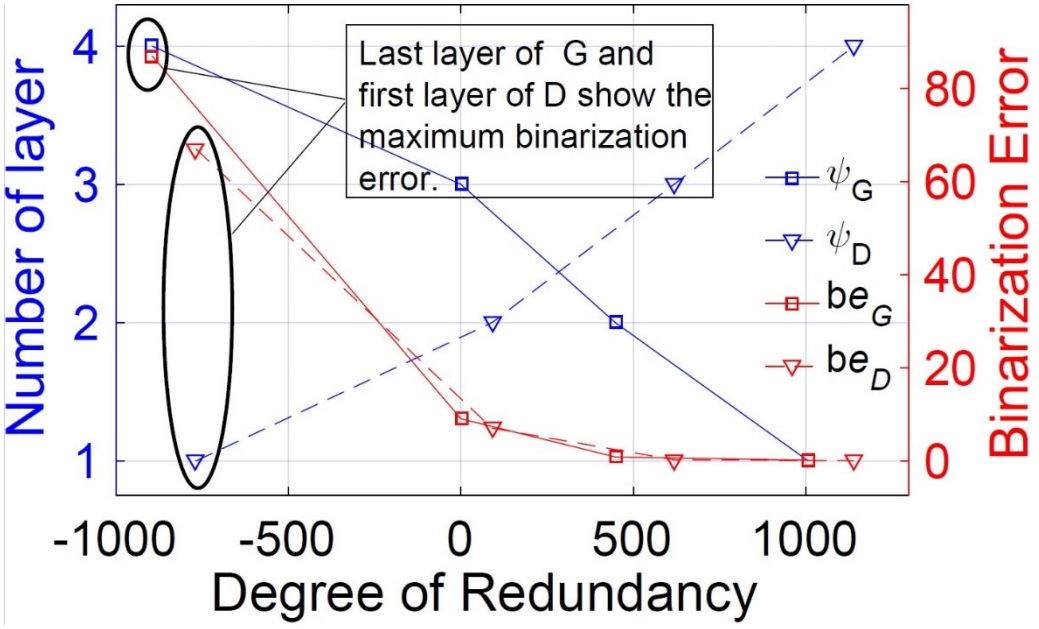


Figure 3.3: Number of layers and binarization error (be) w.r.t degree of redundancy (ψ).

where, l_D as adversarial loss is computed by Equation (2.9) and $\lambda_s(\lambda_1 - \lambda_3)$ are regularization constants. Whereas training D is performed by Equation (3.4), the G model is trained by $l_G = \left\| E_{x \sim P_{data(x)}} f(x) - E_{x \sim p_z(z)} f(G(z)) \right\|_2^2$, where the intermediate layer of D, penultimate layer, defines $f(x)$. In (III), the weights will be updated during back-propagation and stochastic gradient descent is utilized to minimize the loss. The next iteration starts to recompute the weight scaling factor and binarize weights as described in step (I).

To realize the possible layers to be quantized, in both G and D networks, *degree of redundancy* parameter [94], $(\psi) \approx (c_i - h_i w_i)$, is utilized. This term is definedand computed based on the input matrix dimension, where c_i is the number of channels, h_i and w_i are the number of height and width, respectively. It has been proven that the deconvolution layer with the negative value of ψ, which indicates that the dimension of theinput space is lower than the dimension of the

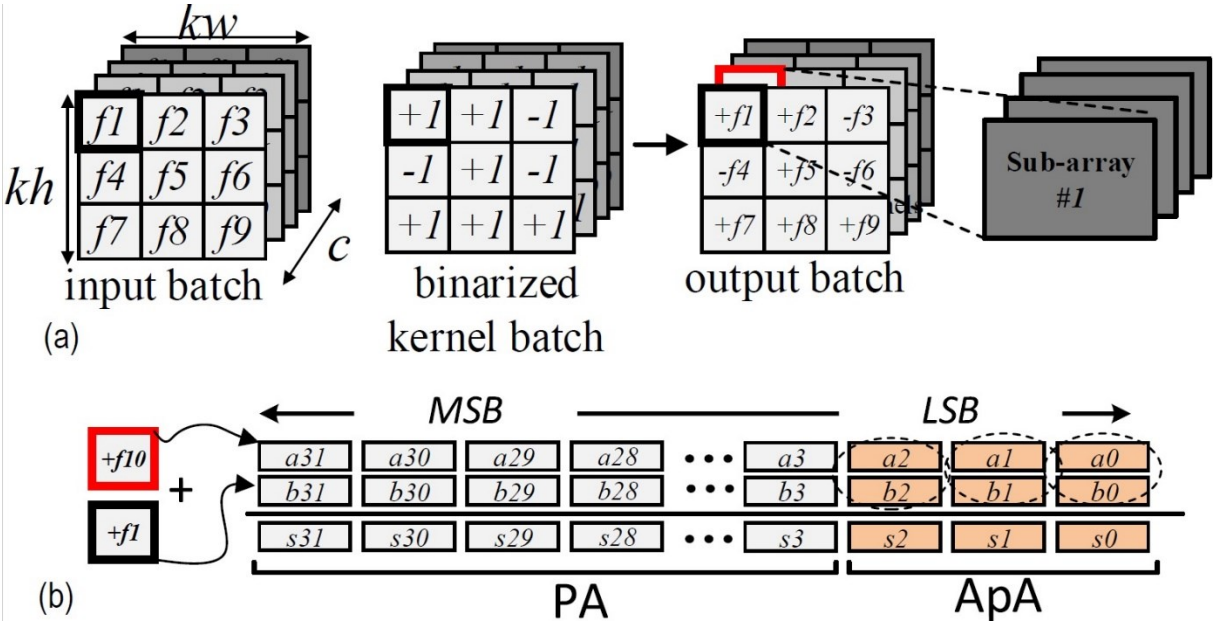


Figure 3.4: (a) ApGAN's binary convolution, and (b) partial approximate computing on three LSBs.

outputspace, is more susceptible to binarization errors. The obtained results regarding ApGAN, as shown in Figure 3.3, depict deeper layers, i.e., layer 1 (4) in D (G), generate the lowest values for degree of redundancy, means biggest negative number, which causes the maximum binarization errors. As a result, the shallower layers, layers with a higher degree of redundancy, will be binarized. To avoid further accuracy degradation in our ApGAN, all the deconvolution layers in G except the last one and all the convolution layers in D except the first and last layers (these layers are kept in floating point, un-binarized, format) are quantized.

3.4 Partial Approximate Computing Unit

Approximate computing paradigms can improve metrics such as energy, delay, and area at the cost of lower accuracy [95]. However, the technique needs to be applied judiciously to avoid

unacceptable error in output behaviors. Nowadays, approximate computing paradigms have been studied extensively to improve the performance efficiency of systems such as energy and area reduction at the cost of lower accuracy. Figure 3.4 (a) depicts the simplified computation of ApGAN's binarized convolutional layers. Initially, c channels (herein $c = 4$) in the size of $kh \times kw$ (herein 3×3 has been used) are selected from input batch and accordingly generates a combined batch with respect to the corresponding {-1, +1} kernel batch. The combined batch is then mapped to the designated computational sub-arrays of ApGAN accelerator (detailed in chapter 3.6). After this step, the main computation is to perform full-precision addition/subtraction between 32-bit output feature maps. Since, implementation of the whole design using approximate adders results in large errors in outputs, herein, a partial approximate computing unit consisting of a Precise Adder (PA) and an Approximate Adder (ApA) is developed. As shown in Figure 3.4 (b), the PA and ApA are used for the most significant bits (MSBs) and the least significant bits (LSBs), respectively, in a manner to maximize the accuracy and minimize energy consumption. In order to find the optimal number of LSBs for ApA, regarding accuracy and energy trade-off, PyTorch implementation of ApGAN inspired by BGAN and BRE regularization [91] method combined with depthwise separable convolution is developed and evaluated.

3.5 ApGAN Accelerator

3.5.1 Architecture

In order to address data transfer and computation limitations of various GAN architectures, we develop an in-memory accelerator for approximate GAN, based on memristive computational

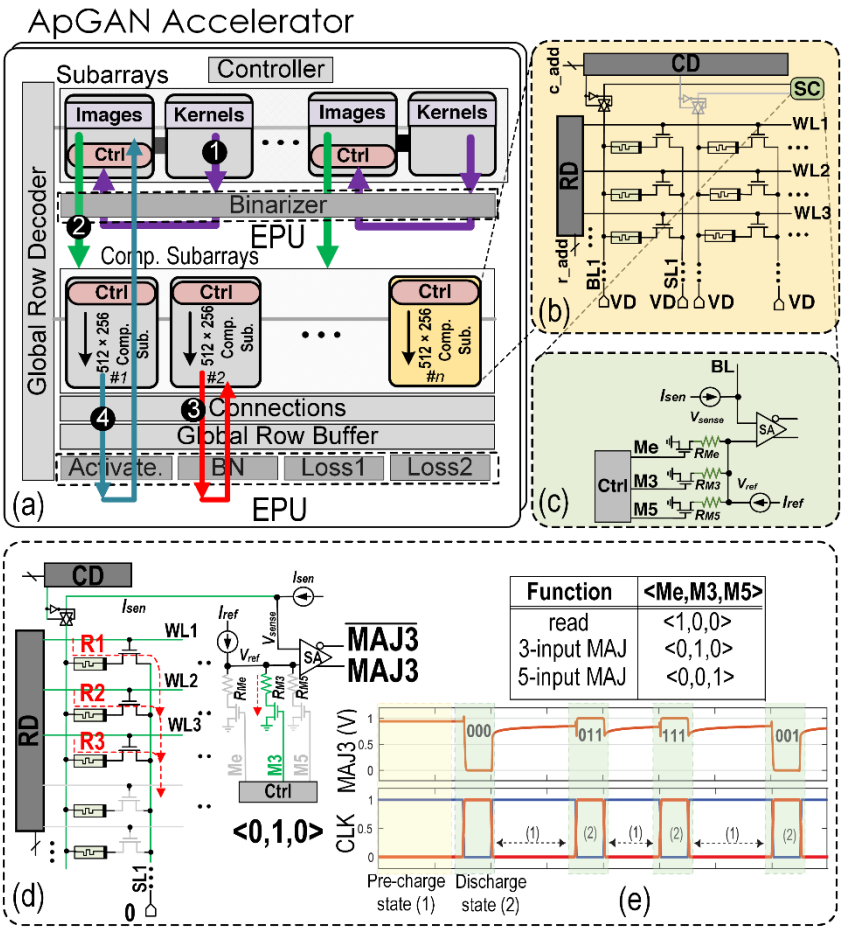


Figure 3.5: (a) The ApGAN accelerator, (b) memristive computational subarray architecture, (c) configurable memory sense amplifier, (d) 3-input majority functions realization using resistive references, and (e) MAJ3's transient response for four different inputs.

sub-array. In comparison to well-trained GANs using floating point operations on CPUs and GPUs, ApGAN has the least computational complexity on the underlying hardware, due to the binarization of weights in the forward path. The proposed accelerator can execute the entire GAN training step discussed in the previous sections and the forward path of training in both discriminator and generator units is focused upon herein. The architecture of ApGAN accelerator is shown in Figure 3.5 (a). It includes Image and Kernel sub-arrays, distributed across the memory banks, which are storing the original values of input feature-maps and weights, respectively. It also

contains the memristive computational sub-arrays and an External Processing Unit (EPU) with five computational components (i.e., Binarizer, Activation Function, Batch Normalization, Loss Functions 1and 2). Mathematically, a $DConv$ can be implemented with a direct $Conv$ [78]. This step is achieved by adding zeros, using zero padding between inputs in the feature maps, and then computing the convolution phase between the kernels and extended input feature maps. Since, in the forward path the binarized weights are utilized, all the $DConv$ and $Conv$ operations are converted to subtraction/addition (sub/add). Here, we give an overview of ApGAN accelerator's functionality. Initially, for each ApGAN layer, c channels in the size $kh \times kw$ are selected from input batch and accordingly produce a combined batch to which is the corresponding binary {-1, +1} kernel processing (1) performed by the EPU's binarizer. This step is readily accomplished by changing the sign-bit of input data with regards to the kernel data. After this step, the channels of a combined batch are transposed and mapped to the designated computational sub-arrays of ApGAN (2). The presented computational array architecture can support massively-parallel and flexible bit-width add/sub operations required in forward path of ApGAN's training as elaborated in the next part. After parallel processing over combined batches, EPU's shared components are employed to process (3) the batches (i.e., calculating the losses, etc.) and eventually generate output feature-maps (4) required for next layer.

3.5.2 Resistive Computational Sub-Array

The memristive sub-array architecture is shown in Figure 3.5 (b). This architecture includes one modified Row Decoder (RD), Column Decoder (CD), and Sense Circuitry (SC). SC includes one configurable sense amplifier per bit-line to maximize the throughput (Figure 3.5 (c)) and can be adjusted by $Ctrl$ unit to morph between write operation and 3 possible read-based in-memory

operations. Write is accomplished by activating the corresponding Word-Line (WL) using *RD* and then applying the differential voltage to the corresponding Bit-Line (BL) and Source-Line (SL) by voltage driver leading to a change in memristor resistivity to either High-R_H (/Low-R_L). Read operation is performed by activating the corresponding WL. The corresponding BL activated through CD is connected to the SC. The SC's sense amplifier generates a read current passing through the resistive device to the grounded SL to generate a sense voltage (V_{sen}), which is then compared with memory reference voltage activated by Me signal ($V_{sen,Low} < V_{Me} < V_{sen,High}$). Accordingly, the sense amplifier outputs Low-'0' (/High-'1') voltage if the path resistance is lower (/higher) than R_{Me}, memory reference resistance. We propose to extend the existing *SC* unit only by adding two low-overhead reference resistances per sense amplifier to enable required in-memory computing within ApGAN's sub-arrays. The proposed configurable memory sense amplifier (Figure 3.5 (c)) now consists of three reference-resistance branches that can be selected by control bits (Me, M3, M5) by the sub-array's $Ctrl$ to carry out one-threshold memory, 3-input (MAJ3), and 5-input (MAJ5) majority functions and their complement in a single memory cycle, respectively. To perform such in-memory computation, every three (/five) resistive cells located in the same bit-line could be activated by *RD* and sensed to implement MAJ3/MAJ5. To realize MAJ3 operation, as shown in Figure 3.5 (d), RM3 is set between $R_L//R_L//R_H$ ('0','0','1') and $R_L//R_H//R_H$ ('0','1','1'). For MAJ5, such reference is set between $R_L//R_L//R_L//R_H//R_H$ ('0','0','0','1','1') and $R_L//R_L//R_H//R_H//R_H$ ('0','0','1', '1','1'). Now, parallel resistances of selected three(/five) cells will be compared with the corresponding reference resistances to produce desired output.

3.5.3 Configurable In-Memory Addition Scheme

As the main operation of ApGAN, add/sub is widely used toprocess most iterative layers which consume the vast majority of the run-time in the network. Therefore, we present a parallel in-memory computation and mapping method for add/sub is based on ApGAN's resistive computational subarrays to accelerate multi-bit operations. A close observation on Full-Adder (FA) truth table clarifies that an approximate FA (25% -ER on *Sum*) could be implemented through making approximate *sum* like $Sum_{app} = \overline{C_{out}}$. Based on this, a streamlined and cost-effective approximate in-memory FA circuit can be designed by storing three input operands (R_i, R_j, R_k)as resistances in the same memory bit-line and then using the MAJ3 scheme (Figure 3.5 (d)). The C_{out} and Sum_{app} of such adder are generated through MAJ3(R_i, R_j, R_k) and $\overline{MAJ3(R_i, R_j, R_k)}$, respectively, in a single memory cycle. Moreover, the accurate sum (Sum_{Acc}) can be carried out through MAJ5($R_i, R_j, R_k, \overline{C_{out}}, \overline{C_{out}}$) with only writing back the $\overline{C_{out}}$ into memory and then applying MAJ5 scheme. In addition to transient response for MAJ3, Figure 3.5 (e) illustrates all possible functional modes.

3.5.4 Instructions

While ApGAN is designed to be an independent energy efficient and high-performance accelerator, we need to expose it to programmers and system-level libraries to use it. From a programmer perspective, ApGAN is a third-party accelerator that can be connected directly to the memory bus or through PCI-Express lanes rather than a memory unit, thus it is integrated similar to that of GPUs. Therefore, a virtual machine and ISA for general-purpose parallel thread execution need to be defined similar to PTX [96] for NVIDIA. In this way, the programs will be

translated to the ApGAN hardware instruction set at install time. ApGAN basically supports three main instructions of in-memory copy (consecutive read and write), MAJ3 and MAJ5. The in-memory copy takes two operands corresponding to destination and source row addresses. MAJ3 and MAJ5 takes the address of input operands and write back the result on a destination row. Such instructions is directly copied/written to a predefined memory-mapped address ranges, for example, in the memory type range registers (MTRRs), or by programming to Memory-Mapped I/O regions that are allocated through a simple device driver to do initialization/cleanup for required software memory structures. We allotted the subsection 3.6.4 to the aforementioned explanation as highlighted in the manuscript.

3.5.5 Hardware Mapping

Figure 3.6 elaborates the required data organization and computation steps of ApGAN with a straightforward and intuitive example only considering the add operation. Clearly, sub can be implemented based on add. Considering n-activated sub-arrays with the size of $x \times y$, each sub-array can handle the parallel add/sub of up to x elements of m-bit ($3m + 4 \leq y$) and so ApGAN could process $n \times x$ elements simultaneously within computational sub-arrays to maximize the throughput. After the mapping step (2) shown in Figure 3.5 (a), the parallel in-memory adder of ApGAN accelerator operates to produce the output feature maps. The memory sub-array organization for such parallel computation is delineated in Figure 3.6 Four reserved rows for Carry

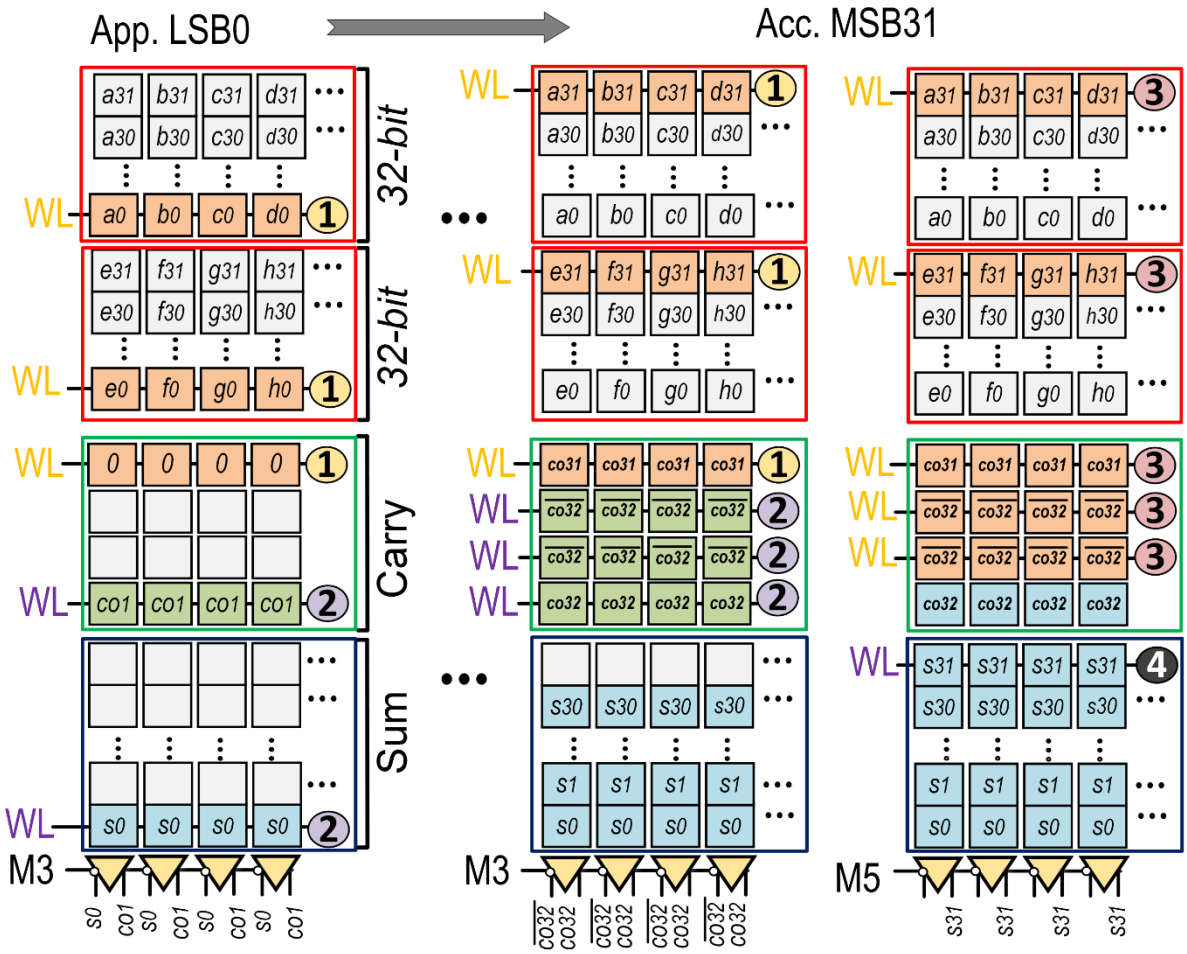


Figure 3.6: Mapping and parallel in-memory addition within the resistive computational sub-array of ApGAN.

results initialized by zero and 32 reserved rows are considered for Sum results. Every pair of corresponding elements to be added together have to be aligned in the same bit-line. Herein, channel 1 (Ch1) and Ch2 should be aligned in the same sub-array. Ch1 elements occupy the first 32 rows of the sub-array followed by Ch2 in the next 32 rows.

The addition algorithm starts bit-by-bit from the LSBs of the two words and continues towards MSBs. We consider approximate computation for LSBs and accurate computation for MSBs based

on conclusion drawn from algorithm level evaluations in chapter 3.5. Figure 3.6 L.H.S. shows App. LSB computation. There are 2 cycles for every bit-position to perform such computation. In step one (1 in Figure3.6), two WLs (accessing to LSBs of elements) and one reserved carry row are enabled to generate C_{out} and Sum_{app} in parallel for whole memory sub-array with $Ctrl$'s M3 command. During step (2), two WLs are activated to save back the results to the designated locations. This carry-out bit overwrites the data in the carry latch and becomes the carry-in of the next cycle. This process is concluded after $2 \times m$ cycles, where m is a number of bits in its elements. Figure 3.6 R.H.S. shows an Acc. MSB computation as a 4-cycle operation. In step (1), two WLs (accessing to LSBs of elements) and one reserved carry row are enabled to generate C_{out} in parallel. During step (2), three WLs are activated to store back the results of C_{out} and $\overline{C_{out}}$ to the designated locations. Now, five WLs are selected (step (3)) to generate the Sum_{acc} with $Ctrl$'s M5 command and write it back (step (4)) to the sub-array. The Acc. MCB computation is concluded after $4 \times m$ cycles, where m is a number of bits in its elements.

3.5.6 Parallelism

Here, we design a Fully-Pipelined Computation mechanism named FPC on top of the presented Spatial Parallelism (SP) method in [78] to boost ApGAN performance. The input data are usually processed in 8/32/64 batch size-b during the training phase. For the sake of simplicity, Figure 3.7 depicts FPC method with a batch size of 2. Obviously, regardless of pipelining, GAN training takes b ×(D1+D2+G) cycles. Typically, if all inputs in the prior batch are processed, a new batch can come into the pipeline. The key idea behind FPC is to duplicate the data for intermediate layers such that pipelining can be readily achieved in ApGAN. Figure 3.7 shows such pipeline for $b1$ and $b2$. Consider DL as the discriminator's layers, D1 needs DL+1+DL+(b-1) cycles, where b-1 cycles

are needed for draining a batch from a pipeline. The (SP) method [78] proves that for each input batch, as there is no data-dependency between the training phases of discriminator, they can

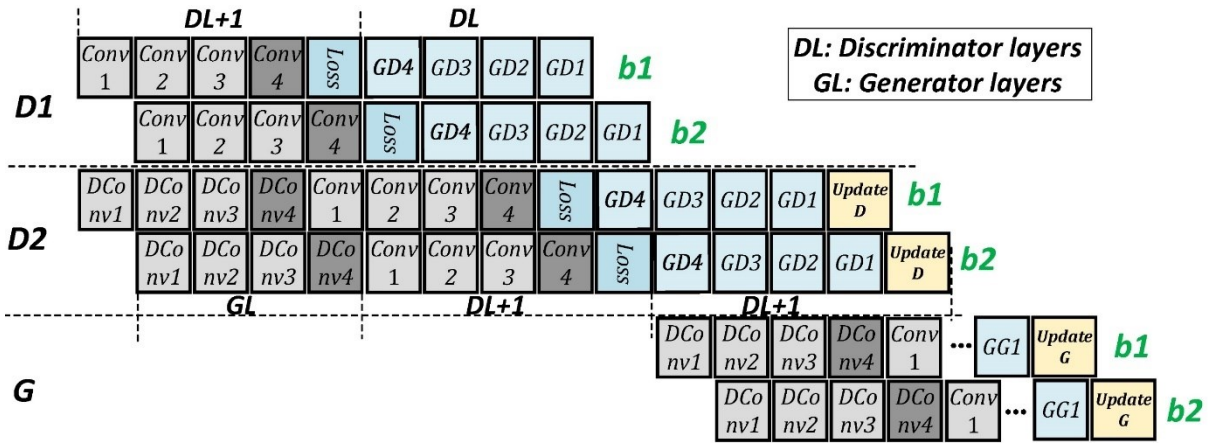


Figure 3.7: Fully-paralleled training method for ApGAN.

perform simultaneously. We exploit the SP method in FPC, as shown Figure 3.7; D1 and D2 training phases occupy different computational sub-arrays and both

$Conv$ and $DConv$ layers can be run at a same time. Consider GL as the generator's layers, D2 takes GL+DL+1+DL+1+(b-1) latency for updating the D. Besides, FPC takes advantage of this observation that after D2's loss function computation and back-propagation to GD4 layer, the training of generator for different batches can be started while the corresponding GD3 is being processed in D2. This phase takes 2DL+2GL+2+(b-1).

3.6 Performance Evaluation

3.6.1 Experimental Setup and Results

In order to perform a fair comparison between our design and the well-known GAN models, DCGAN, WGAN-CP, and WGAN-GP, the same architecture including four convolution and deconvolution layers for D and G, respectively, is leveraged.

Datasets: We conduct experiments of ApGAN on several datasets to evaluate the performance of the proposed algorithm, including MNIST [97], Fashion-MNIST [98], CIFAR-10 [99], STL-10 [100], and celeb-A [101]. MNIST is leveraged as a gray-scale dataset which contains 70,000 28×28 images of handwritten digits from 0 to 9, 60,000 images for training and 10,000 images for testing sets. Similar to MNIST, Fashion-MNIST consists of 28×28 gray-scale images but it includes 10,000 images for each of training and testing sets to form ten fashion categories. We use CIFAR-10 for RGB images of size 32×32. It has 60,000 images evenly distributed in ten distinct classes, in which 50,000 and 10,000 examples are used for training and testing, respectively. In addition to CIFAR-10, STL-10 is used, which is similar to CIFAR-10 dataset except that it has 100,000 unlabeled images for unsupervised learning and only 500 labeled images for training. Finally, we also exploit celeb-A to evaluate performance quantitatively. It includes 202,559 images of celebrity faces labeled with 40 different face attributes and because each image consists of only one face, the quality of the generated images is readily evaluated.

Evaluation Metrics: According to [102], which includes extensive studies for highly-used metrics i.e., log-likelihood to evaluate the performance of NN models, authors showed there is not necessarily a direct relationship between the good performance of GANs and the metric(s). Therefore herein, we use Inception Score (IS) [103] as an evaluation metric in our experiments, which is leveraged to measure information on the quality and variation of the generated images by using a pre-trained inception V3 [104] network. The IS's of generators is calculated by

$$IS_G = \exp\left(E_{x\sim P_G} D_{KL}(p(y|x)\|p(y))\right) \tag{3.5}$$

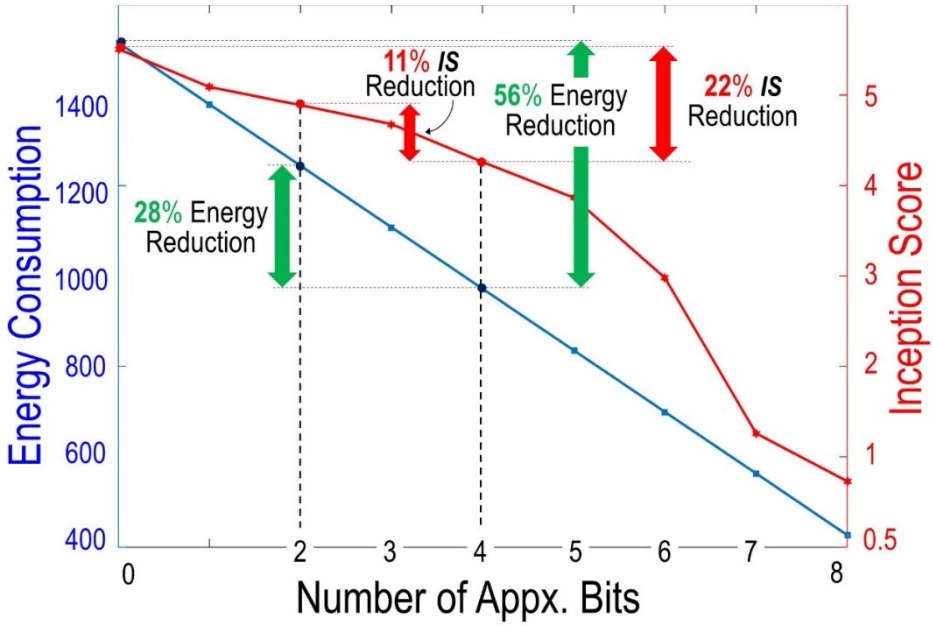


Figure 3.8: Energy consumption versus IS regarding number of approximated bits.

where x is an image, y is the output label which will be predicted, and $D_{KL}(p|q)$ is the KL divergence between two distributions, p and q. A high IS^2 illustrates diversity and clarity among generated images and it is achieved if $p(y|x)$ low entropy, means that the generated image includes clear objects, and $p(y)$ is high entropy, which indicates a high diversity of images from all categories.

Results and Analysis. Herein, several sets of experiments on both CIFAR-10 and STL-10 using DCGAN, WGAN-CP, and WGAN-GP are conducted. First, GAN networks are trained using 32-bit floating point number weights as the baseline. Next, several variant GANs are trained from scratch. Since the GAN training phase usually suffers from training instability and convergence problems, the change of IS is monitored after each epoch, which helps us to observe the stability of the proposed method compared to the full precision models.

Table 3.1: IS Values on CIFAR-10 and STL10 Datasets.

Model		CIFAR-10	STL-10
DCGAN	32-bit	5.46±0.2	2.93±0.2
	DoReFa-Net [74]	1.2±0.003	1.39±0.007
	TWN [105]	1.09±0.003	1.45±0.008
	TGAN [109]	4.52±0.1	2.91±0.3
	ApGAN	**5.01±0.08**	**2.46±0.07**
WGAN-CP	32-bit	4.69±0.15	3.13±0.1
	DoReFa-Net [74]	3.84±0.09	2.37±0.05
	TWN [105]	4.26±0.07	2.78±0.06
	TGAN [109]	3.76±0.07	2.31±0.09
	ApGAN	**4.46±0.15**	**2.39±0.1**
WGAN-GP	32-bit	5.51±0.008	3.04±0.09
	DoReFa-Net [74]	4.70±0.05	2.31±0.012
	TWN [105]	4.45±0.05	2.68±0.015
	TGAN [109]	4.98±0.01	2.81±0.05
	ApGAN	**5.08±0.05**	**2.61±0.09**

Figure 3.8 depicts the IS results for ApGAN on CIFAR-10 with respect to the number of approximated LSBs, and energy consumption of the convolution layers. The optimal condition occurs when 2 to 4 LSBs are approximated, which leads to a relatively high reduction in energy whereas IS is slightly decreased. Table 3.1 summarizes ISs of DCGAN, WGAN-CP, and WGAN-GP on CIFAR-10 and STL-10 datasets [105]. In addition to the 32-bit full-precision as the baseline, ApGAN and three other GANs including ternarized and binarized-weight training are examined. Based on the obtained results, the full precision WGAN-GP and WGAN-CP show the best ISs for CIFAR-10 (5.51) and STL-10 (3.13) datasets, respectively. Although the IS of our proposed ApGAN degrades roughly by 0.37 (in both examined datasets) compared to the best results, it shows better scores than 32-bitWGAN-CP and almost all of the proposed fully-quantized training approaches. Moreover, the training convergence behaviors for all the examined GANs are shown

in Figure 3.9. Although the baseline full-precision training has a faster convergence, our ApGAN achieves comparable IS results for CIFAR-10 and STL-10.

In addition to the quantitative comparison, Figure 3.10 depicts the generated images by ApGAN architectures for five different datasets as qualitative evidence. The generated images which look similar to the full-precision DCGAN's results verify the performance and functionality of ApGAN. Figure 3.11 depicts loss values for both discriminator (D) and generator(G) networks in full precision, fully-binarized and ApGAN in Celeb-A dataset. The **y** and **x** axes indicate the loss values and the number of epochs, respectively. As depicted in the fully-binarized network shown in Figure 3.11 (b), after a few epochs for initializing and competition steps, the convergence process and consequently improvement in the generated images stop.

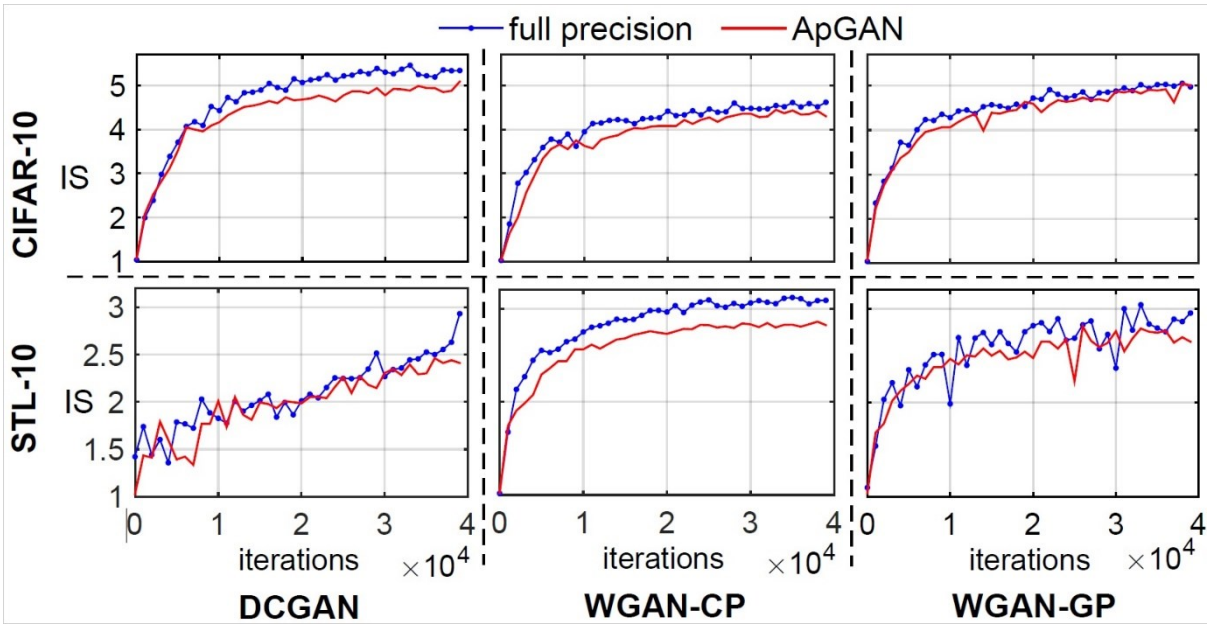


Figure 3.9: Inception score on CIFAR-10 and STL-10 datasets leveraging full precision and ApGAN for different GANs.

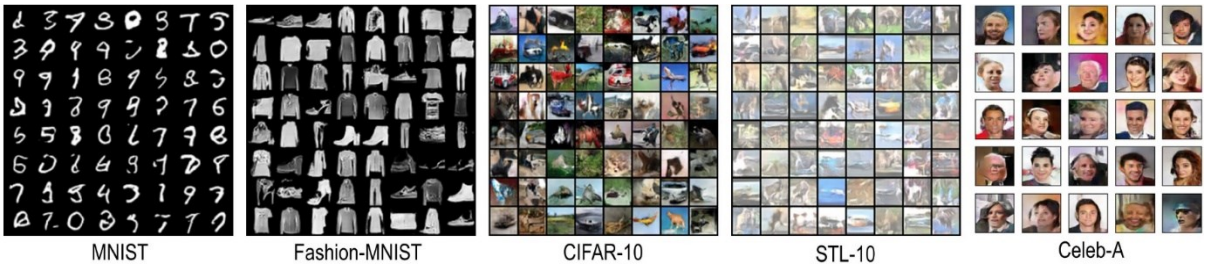


Figure 3.10: Generated images for various datasets by ApGAN.

Nonetheless, for ApGAN, after initial state, competition starts quickly to improves the quality of the generated images and due to the semi-balanced binarized structures for D and G, the competition continues for a sufficient number of epochs. The ApGAN actually converges in an almost similar manner as the original 32-bit full-precision training.

3.6.2 Hardware Setup and Results

In this section, we estimate ApGAN's energy-efficiency and performance and compare it with other feasible GAN accelerators (based on ASIC, SOT-MRAM, ReRAM, and GPU) based on three GAN architectures (DCGAN, WGAN-CP, and WGAN-GP). It is clear that the larger chip area is, then the higher performance for ApGAN and other accelerators are achieved due to having additional sub-arrays or computational units, albeit the memory die size impacts the area cost. To have a fair comparison in this work, we report the area-normalized results (performance/energy per area), henceforth.

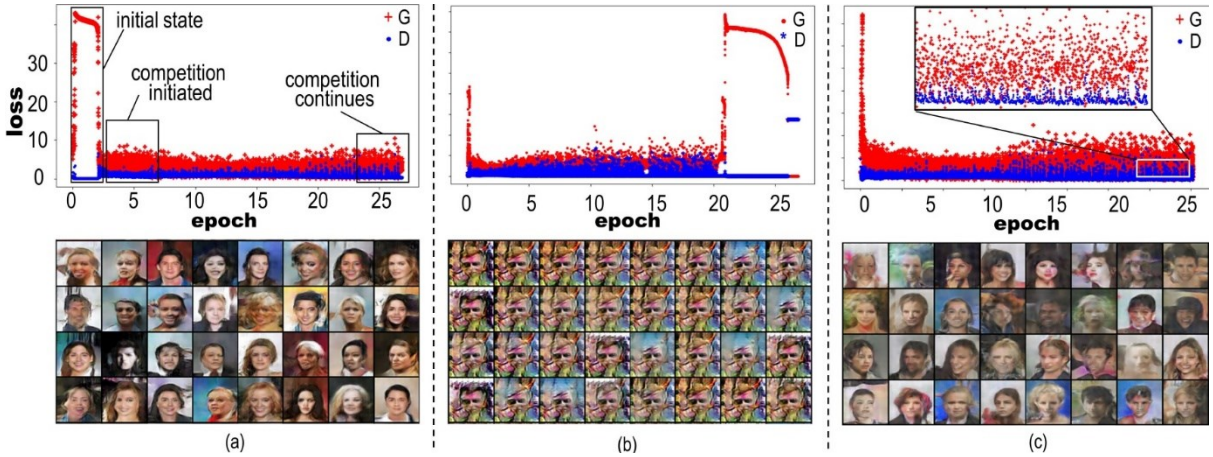


Figure 3.11: Value of losses in (a) 32-bit (full precision) DCGAN, (b) fully-binarized DCGAN, and (c) proposed ApGAN.

Experiment Setup: To assess the performance of the proposed accelerator at the circuit-level, we use the SPICE model for memristors with the Ag-Si memristor device parameters from [106]. We then combine the SPICE models of CMOS transistors and memristors under NCSU 45nm CMOS PDK [107].To perform the system-level evaluations, we modified the memory evaluation tool NVSim [108] to co-simulate with our developed in-house C++ code based on circuit-level results. We configure the memory organization of the sub-arrays with 512 rows and 256 columns per memory matrix (mat) considering an H-tree routing method, 2×2 mats per bank, 8×8 banks per group; in total 16 groups leading to a 512 Mb total capacity. For comparison, a ReRAM-based in-memory accelerator based on [78] was developed with 256 fully functional sub-arrays with the size of 256×256 and eight-bit configurable SAs. To perform the evaluations, NVSim was extensively modified to estimate the system energy and performance adopting its default ReRAM cell file (.cell). We developed a SOT-MRAM-based accelerator based on PIM-TGAN [109]. For the circuit level simulation, a Verilog-A model of 2T1R SOT-MRAM device is developed to co-simulate with the interface CMOS circuits in SPICE. Finally, an architectural-level simulator was

built on top of NVSim. To compare the result with ASIC accelerators, we developed a YodaNN-like [110] design with two 8×8 tiles configuration. Then, the designs were synthesized using Design Compiler [111] with 45nm technology. The SRAM and eDRAM performances were calculated using CACTI [112]. We created a comprehensive Verilog model for EPU to interact with our SPICE circuit code to perform the evaluation. Activation functions were developed based on lookup-table-based transformations [113] with case-statement codes. Batch normalization unit generally performs an affine function ($y = kx + h$) [114], where y and x represent the corresponding output and input feature map pixels, respectively. During inference mode, all the other parameters (k, h) are pre-computed and stored in ApGAN sub-arrays, therefore, Batch normalization unit can readily fetch each pixel of input feature map, fed forward to the batch-norm

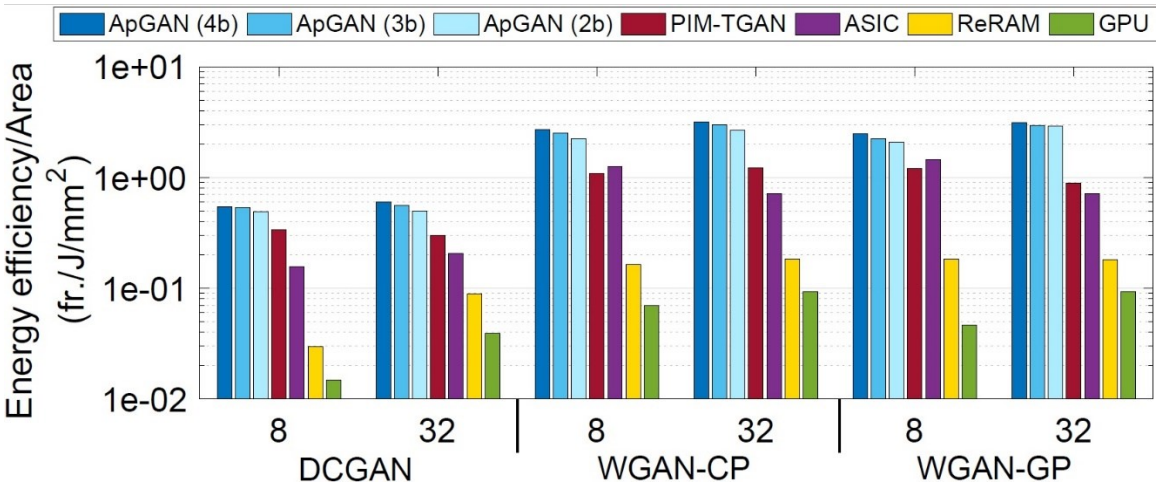


Figure 3.12: Energy-efficiency evaluation of various platforms normalized to the area (Y-axis: log scale).

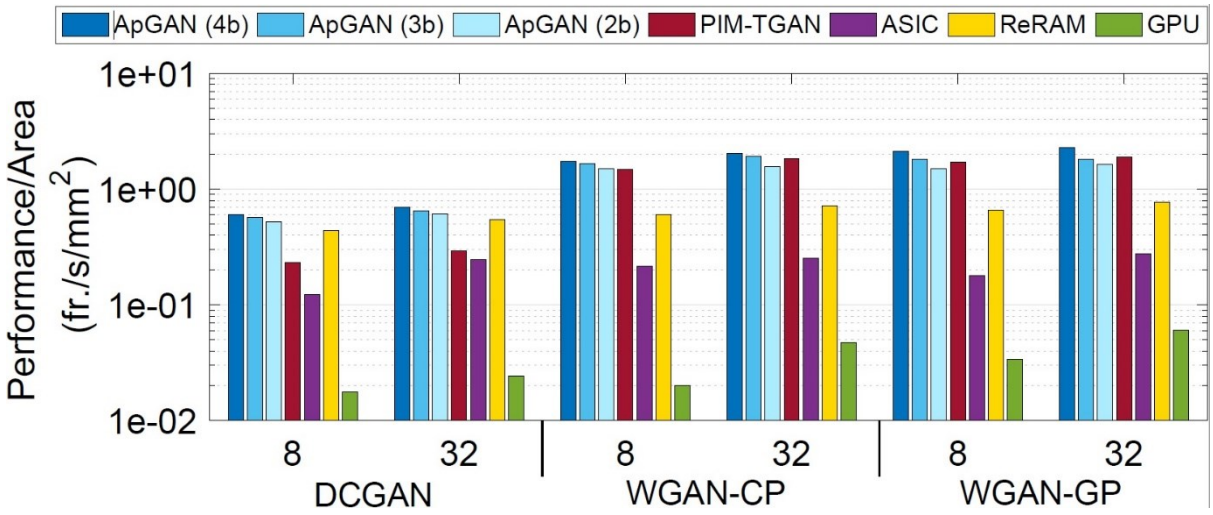


Figure 3.13: Performance evaluation of various platforms normalized to the area (Y-axis: log scale).

layer, and write back the corresponding normalized pixel employing an internal, multiplexed CMOS adder and multiplier to perform this computation efficiently.

Energy Efficiency: Figure 3.12 shows ApGAN's energy efficiency (frames per joule) results implemented by *FPC* method for three possible approximation degree (i.e., 2-, 3-, and 4-bit) compared with other designs, running a similar task under two batch size configuration, i.e., 8 and 32. Here, as the batch size gets larger, higher energy-efficiency is obtained. We can see that ApGAN-4b has the highest energy efficiency normalized to the area, related to other methods, as a result of its 4-bit approximated, parallel, energy-efficient operations. ApGAN-3b shows ~2.5 ×, 13.1 ×, and 28.6 × higher energy-efficiency than that of the leading ASIC, ReRAM, and GPU-based solutions. This energy reduction arises from three sources: 1) standard $Conv$ and $DeConv$ operations in the forward path are replaced with energy-efficient add/sub operations due to binarization, 2) ApGAN's interlayer parallelism which massively reduces the latency of operations

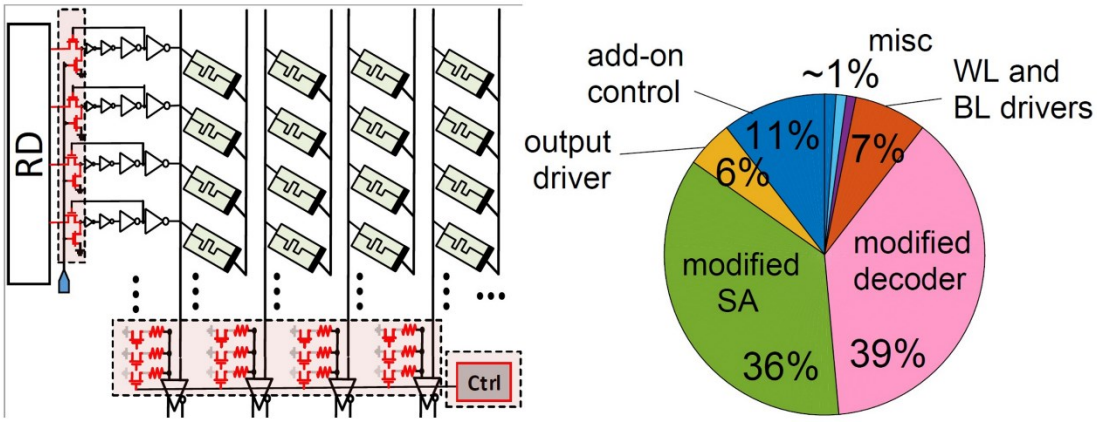


Figure 3.14: : (a) Three main hardware cost sources in ApGAN's sub-array. Note: access transistors and CD are not shown for simplicity, and (b) area overhead breakdown of ApGAN.

and 3) bulk and energy-efficient approximated in-memory operations of ApGAN. Compared to the recent processing-in-MRAM platform in [109], ApGAN reduces energy consumption by ~2.3 ×.

Throughput: Figure 3.13 compares the ApGAN throughput (frames per second) results for three possible approximation degree (i.e., 2-, 3-, and 4-bit), normalized with the area, for different accelerators. Based on the results, ApGAN-3b is 35 × and 5.8 × faster on average than GPU andASIC-64 methods. This efficiency can be related to parallel and ultra-fast in-memory operations of ApGAN compared to multi-cycle ASIC and GPU operations as well as the potential mismatch between data movement and computation in ASIC and GPU methods. Additionally, ApGAN is 1.9 × faster than ReRAM method. It is worth pointing out that ReRAM accelerators suffer matrix splitting owning to intrinsically-limited bit levels of ReRAM device, thus more sub-arrays need to be occupied. This can further limit parallelism methods. Additionally, a ReRAM

crossbar imposes a large peripheral circuit overhead due to existing DAC/ADC and buffers occupying roughly 85 percent of area [76, 115]. We also observe that ApGAN achieves ~40 percent better performance compared to that of PIM-TGAN platform [109].

Area Overhead: To assess the area overhead of ApGAN on top of commodity RRAM chip, three main hardware cost sources must be taken into consideration as shown in Figure 3.14 (a). First, add-on transistors to SAs; in our design, each SA requires 2 additional transistors connected to each *BL* (Figure 3.5 (c)) to enable in-memory computing; Second, the modified MRD overhead; we modify each *WL* driver by adding two more transistors in the typical buffer chain based on the method used in [116]. Third, the $ctrl$'s overhead to control enable bits; $ctrl$ generates the activation bits with MUX units with 6 transistors. To sum it up, ApGAN roughly imposes 3 additional rows per sub-array, which can be interpreted as ~2 percent of memory chip area. The detailed breakdown of area overhead is shown in Figure 3.14 (b).

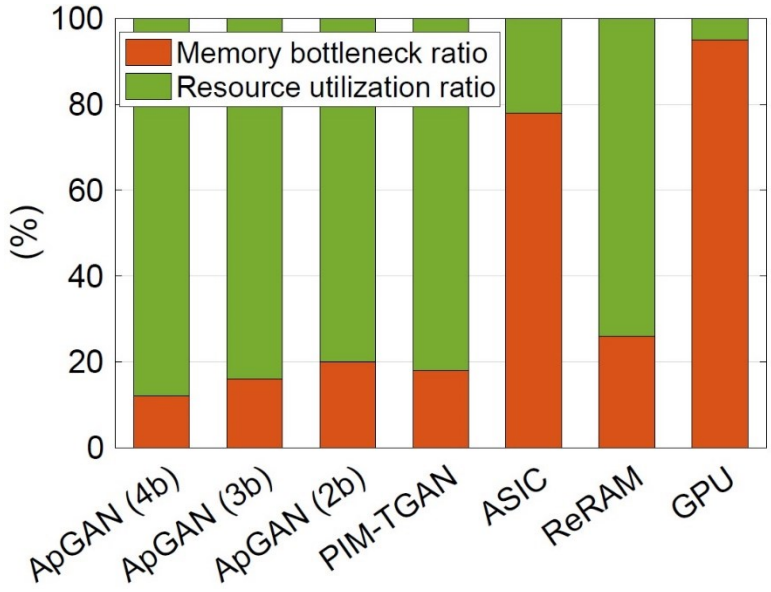


Figure 3.15: Memory bottleneck ratio for different platforms

Resource Utilization: We estimated the time fraction at which the computation has to wait for data and on-/off chip data transfer limits the performance referred to as memory bottleneck ratio for different platforms, as depicted in Figure 3.15. This evaluation is done through the peak performance and experimentally extracted results for each platform considering a number of memory access. We observe that processing-in-memory solutions i.e., ApGAN, PIMTGAN, and ReRAM spend less than 30 percent time for data transfer and memory access. But ASIC and GPU spend over 50 and 90 percent time, respectively, waiting for the loading data from the memory. In this way, we can define a resource utilization ratio for different platforms. We observe that ApGAN-4b achieves the highest ratio by efficiently utilizing up to 88 percent of its computation resources. This number is limited to 5 percent for GPUs performing the similar task.

3.7. Conclusion

In this chapter, we presented a novel hardware-optimized GAN training algorithm using binary weights for three and two layers of generator and discriminator networks, respectively. Moreover, we developed a reconfigurable addition approach in which both approximate and accurate add operations are performed. In order to further accelerate the ApGAN training process, a new PIM accelerator based on memristor was implemented. Finally, in addition to focus on the computational performance of ApGAN exploiting its intrinsic in-memory parallelism to increase the throughput of the system, we developed *FPC* optimization as a spatial parallelism method. The performance of the ApGAN in both quantitative and qualitative approaches have been evaluated on different data-sets including Fashion-MNIST, CIFAR-10, STL-10, and celeb-A. The generated images by ApGAN look similar to the full-precision DCGAN's result. Moreover, the obtained

simulation results showed that our PIM-ApGAN can achieve $\sim 2.5\times$ better energy-efficiency and $5.1\times$ speedup compared to CMOS-ASIC accelerator, whereas IS is degraded by 11 percent. Hence, due to the small IS degradation and a significant reduction in the hardware aspect, the ApGAN can be a promising weight training scheme for resource-limited IoT devices. Since in an environment, tens to hundreds of IoT nodes are distributed, similar approaches which are used in random forest methods, majority voters and mean prediction methods, can be leveraged.

CHAPTER 4 : STM-LTM ARCHITECTURE

4.1. Fundamentals of Biologically-Inspired Computing

Neuromorphic computing offers potential advantages to various applications including high performance, robust learning capabilities, and a more efficient intrinsically-executed approach to processing. Such a computing paradigm is not limited to the separation of memory and processing, and has a high level of parallelism unlike conventional von Neumann architectures [117]. With the significant growth in neuromorphic computing research, various biologically inspired architectures and synaptic learning rules, such as Spike Time Dependent Plasticity (STDP), have been proposed [118]. However, there are still important but underexplored concepts motivated from biology, which can be emulated to improve neuromorphic designs in terms of performance and reliability. One vital example is the realization of biologically-inspired mechanisms of memory. Biological memory systems are extremely complex entities, constantly responding to a vast amount of dynamic multi-modal information. Collection and integration of temporal

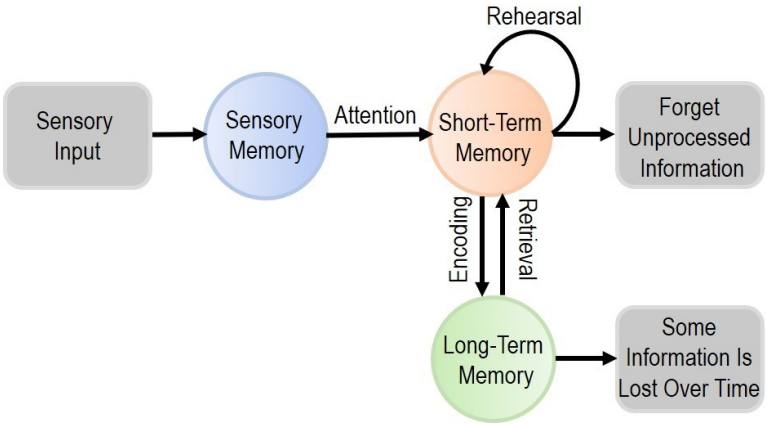


Figure 4.1: The Schematic of biological multistore memory model.

information is one of the fundamental parts of this system, which consists of two main storage mechanisms: Short-Term memory (STM) and Long-Term Memory (LTM) [119].

Figure 4.1 shows a simplified representation of a biological memory, which consists of three different memory models. The sensory memory retains immediate information from the environment and is considered as the first stage of the memory, lasting only for a few milliseconds. This mechanism helps the brain to regulate the flow to avoid a flood of information. However, this information can be transferred to STM through detection and enforcement of temporal focus, once a selected stimulus has been cognitively perceived [120]. The STM can span on the order of seconds to minutes, during the interval when biological brains initiate memory formation via their molecular and cellular machinery. However, retention of information in STM can only be sustained by repeated stimulus. Repeated stimulation of synaptic structures increases the probability of STM to LTM transformation, a process termed consolidation [121]. Under requisite conditions, STM is transitioned to LTM, depending on the strength of molecular reactions and encoding. Thus, the LTM can last from months to years or become permanent, despite the attenuation which would occur otherwise without continuous stimulation [121].

From a hardware implementation perspective, emerging electronic devices can offer a viable way to mimic several plasticity measurements observed in biological synapses as opposed to conventional complementary metal-oxide semiconductor (CMOS) circuits [122]. Memristors with resistive coupling have been widely exploited to implement synapses in addition to the integrate-and-fire capability of a McCulloch–Pitts model neuron [123]. However, since the accessible signal gain and endurance in such fully-memristive networks are limited, other resistive paradigms such as spintronic devices have been taken into consideration [124]. There are a variety of hybrid

arrangements of device technologies that can exploit alternative mechanisms, such as capacitive synapses used in place of resistive coupling, which feature an ultra-small static power dissipation [125-129]. In [127], a capacitive neural network has been proposed that utilizes a charge-based capacitor crossbar to perform multiply-and-accumulate (MAC) operation. Such designs realize the weighted summation of inputs through capacitive coupling and voltage division and generates the output in a read-like operation. Nevertheless, most of the research to realize synapse plasticity change in response to neuron spiking trains has been so far limited to long-term plasticity [59, 76, 130], while the volatility of biological memory has been overlooked.

In [131] and [132] the authors show the functional resemblance of two different emerging devices to the short-term to long-term memory transition. In [132] the authors demonstrate that stimulating a memristor device with repeated voltage pulses can result in an effect analogous to memory transition in biological systems. A similar approach has been taken in [131] with a Magnetic Tunnel Junction (MTJ), where a sufficient input stimulus can change its magnetization. Both of these works have focused on implementing the memory transition process with a single emerging device module. Although a homogenous device technology approaches aim at the same behavior as biological memory, it does not allow data undergoing consolidation to be used in computation until such a transition has completed. Consequently, a mechanism is sought which not only exhibits this behavior of biological memory but can also utilize the introduced data efficiently. This can be achieved by designing separate modules for STM and LTM in the memory architecture. As in [133], the researchers proposed such a design implemented by two separate spin Hall effect-driven Magnetic Tunnel Junctions (SHE-MTJs), in which the STM synapse potentiates the inputs with a greater probability and forgets at a higher rate than the LTM synapse.

However, the biological STM-to-LTM transition process was not addressed in detail nor optimized for efficient processing.

In this work, we propose an energy-efficient and biologically-inspired long-term and short-term memory architecture, to mimic both biological STM and LTM synaptic connections and timing dependencies of the stimuli, via volatile and non-volatile hybrid spin-CMOS devices with respect to the synaptic memory reinforcement.

4.2. Biologically Inspired STM-LTM Architecture

The proposed biologically-inspired binary STM-LTM memory architecture, shown in Figure 4.2, consists of a 2-D array of memory components leveraging a pair of Volatile Memory (VM) and Non-volatile Memory (NVM) as the memory bit-cell to realize STM and LTM, respectively. The VM utilizes a capacitor, controlled by an access transistor, in a fashion analogous to a DRAM structure. The NVM is designed with a SHE-MTJ [53]. Each memory bit cell is connected to a *Bit-Line (BL)*, *Word-Line (WL)*, and *Source-Line (SL)* managed by the control unit's voltage driver. The *BL* and *WL* are shared amongst the cells within the same row and the *SL* is shared between cells within the same column, as shown in Figure 4.2, to allow the architecture operate in three distinct modes as explained in subsection 4.2.2.

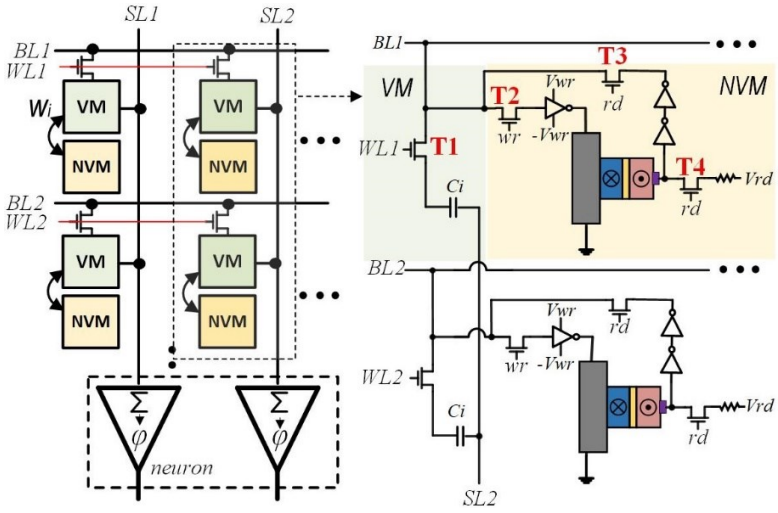


Figure 4.2: The proposed STM-LTM memory architecture with VM and NVM components.

4.2.1. Memory Units

Capacitor as STM: Conventional DRAM is the most abundant, low-cost and simple type of memory offering relatively high speed and density, consisting of one access transistor and one capacitor as the storage element. Recently, several works have explored the potentials of such capacitor-based memories in neural network applications [127, 134]. Training neural networks to high degrees of accuracy requires consecutive, small changes in weights, which NVMs are not ideal for them due to limited speed and endurance. Thus, DRAM offers a suitable mechanism for online (in situ) training due to its relatively high speed and symmetrical read/write with infinite endurance, which is a critical aspect for networks that necessitate constant training in an extended period such as IoT edge devices [128, 135].

In digital capacitor-based accelerators [3, 134], every memory bit-line can perform bitwise digital Boolean logic operations, where each capacitor stores a binary synaptic weight and so a low-bit-width and parallel computation has been realized. These accelerators typically do not require large

peripheral circuits such as ADC, DAC, and router contrary to resistive NVM accelerators [76]. Recently, the analog capacitive cross-bar networks have been demonstrated greatly-reduced static power dissipation to near-zero levels compared with the weighted sum of currents in a resistively coupled network [128, 135]. However, for such networks, the volatility of the capacitor can be a huge disadvantage as it will require the training to start over upon losing power. Thus, leakage and the resulting volatility will increase energy consumption while processing delay can be less than or equal to the total training time.

Here, we aim to implement a capacitive crossbar enhanced with a non-volatile memory in a new fashion based on the STM-LTM features inspired from biology. Each memory bit-cell's capacitor represents a binary synaptic weight ('1' or '0') stored as the "charged" or "discharged" capacitor states. The STM's access transistor (T1 in Figure 4.3 (c)) is controlled by WL enabling selective write/read operation on the cells located within one row.

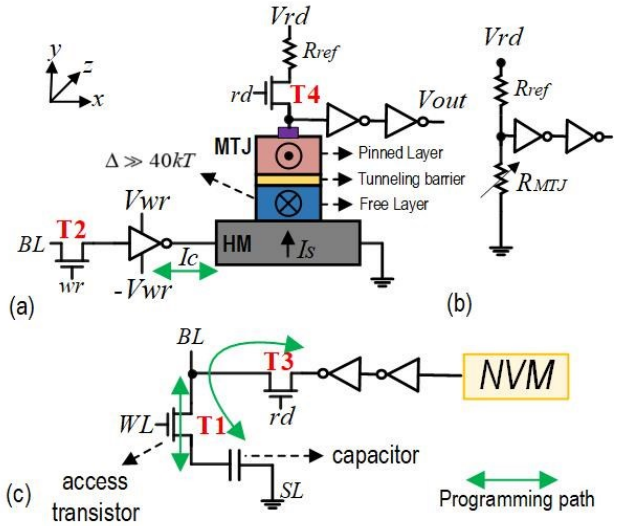


Figure 4.3: (a) Structure of a SHE-MTJ as NVM, (b) Resistive equivalent read circuit of SHE-MTJ, (c) VM structure programming path.

Storing the network weights in the STM (through a write operation) and strengthening the memory (through STM-to- LTM transfer) are two crucial tasks that need to be carried out. For both operations, the capacitor is initially in the Precharged State (P.S.), i.e. the *BL* voltage is preset to $\sim\frac{VDD}{2}$ by the voltage driver. To save a weight on a capacitor as tabulated in Table 4.1, the memory decoder first activates the corresponding *WL* and the *BL* is set to high (VDD) or low voltage (GND). This will provide enough bias voltage to change the capacitor data in a DRAM fashion. The synaptic weight representing STM will be then used to perform the computation or STM-to-LTM transfer.

SHE-MTJ as LTM: The NVM element in the STM-LTM memory architecture is the spintronic SHE-MTJ device described in 2.1.4. that uses a stable nanomagnet (Δ>>40kT), with two CMOS inverters to amplify the output, as shown in Figure 4.3 (a). Figure 4.3 (b) shows an equivalent read circuit of a SHE-MTJ. To read out the data from the SHE-MTJ, a read voltage is applied to sense the resistance of the device through realizing a resistive voltage divider. We have considered 3 access transistors to control the functionality of the SHE-MTJ with respect to our volatile element as shown in Figure 4.2. The T3 and T4 transistors are devised to activate the read path and T2 is to control NVM and VM data transfer.

Table 4.1. The operation modes of the STM-LTM architecture

Operation	***BL***	***WL***	***SL***	***wr***	***rd***
STM Write (1 or 0)	V_{DD} or 0	V_{DD}	0	0	0
Computation	V_{neuron}	V_{DD}	I_{sum}	0	0
STM to LTM	V_{DD} /2	V_{DD}	0	V_{DD}	0
LTM to STM	V_{DD} /2	V_{DD}	0	0	V_{DD}

4.2.2. Circuit Architecture

1) Computing mode using crossbar operation: In this mode, by activating multiple WLs simultaneously (T1 is ON in Figure 4.2) and applying input voltages on BLs, VMs can modulate the input and realize the weighted summation of inputs using a capacitive voltage divider circuit and send it to the output neuron via SL, while NVM is deactivated (T2-T4 are OFF in Figure 4.2). The control signals required for this operation are tabulated in Table 4.1. The realization of an n×m capacitive network inspired by [127, 128] is shown in Figure 4.4. The memory decoder outputs are enhanced by the inverter chain (blue shaded area) to activate multiple WLs simultaneously. The controller governs the timing of the signal going through the crossbar by controlling the memory address and assigning suitable input voltages through the voltage driver. The input signals

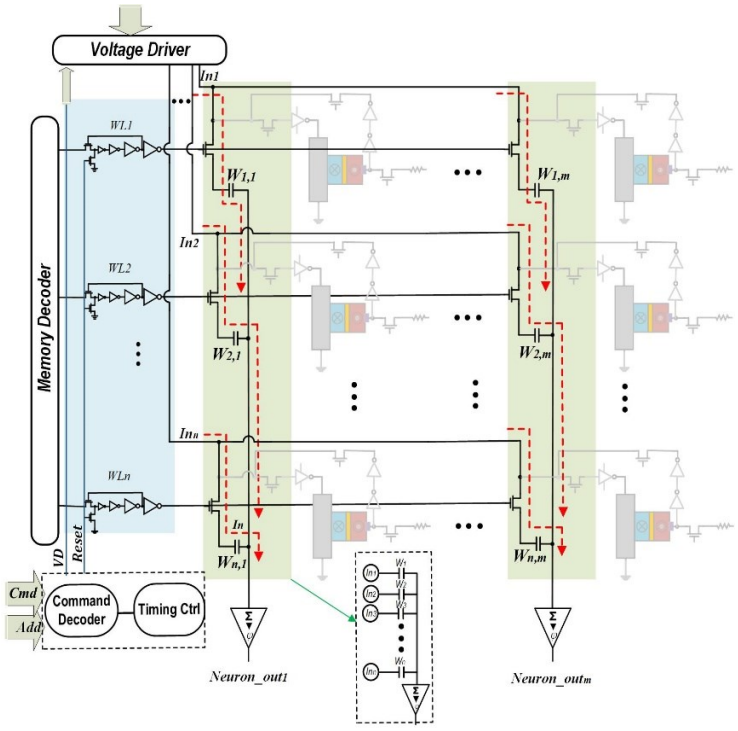


Figure 4.4: Realization of the capacitive network [128] within the proposed LTM-STM memory architecture.

are encoded as voltage pulse and simultaneously charge the array in each capacitive node. In order to perform MAC operation, by applying the Vin as input signal to each row, the charges in capacitors will be redistributed and averaged by a reference capacitance and finally the output voltage can be written as $V_{out} = \frac{\sum_{i=1}^{input} C_{i,j} V_{in,i}}{C_{ref}}$ through voltage division between the cells located in the same column [108].

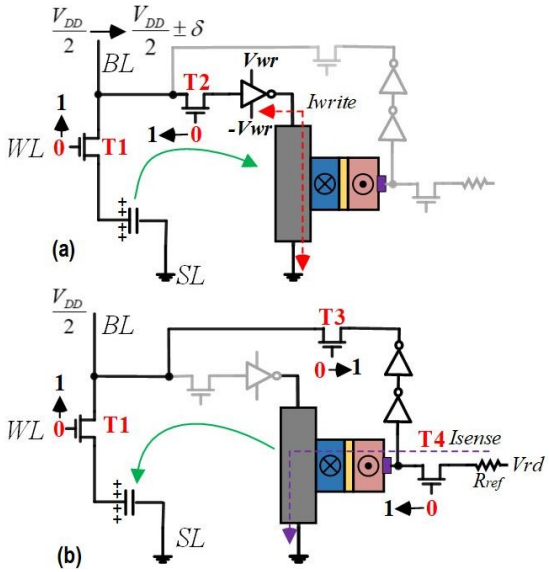


Figure 4.5: (a) STM to LTM transfer and (b) LTM to STM transfer modes.

2) STM to LTM transfer: One of the most significant aspects of memory in biological systems is STM into LTM consolidation after repeated use. To realize this, controller readily keeps the count of input voltages applied to a specific BL, which is implemented using a counting unit within the controller. Accordingly, the controller determines the reinforcement ratio of the synapses. As shown in Figure 4.5 (a), for STM to LTM transfer, at initial state, the BL voltage is precharged to $\sim \frac{VDD}{2}$, while SL is grounded. Now, activating the WL (T1: ON), the selected cell (storing VDD or

0) shares its charge with the BL leading to a small deviation in the initial voltage of BL ($\frac{VDD}{2}\pm\delta$). Then, by activating the T2 transistor by wr signal, the SHE-MTJ's write circuit amplifies the δ of the BL voltage toward bipolar write voltage (Vwr or -Vwr) through voltage amplification. It is worth pointing out that wr signal is shared among the cells located in the same row and controlled by voltage driver to guarantee the simultaneous STM-to-LTM transfer for synapses connected to one particular neuron. Here the flow of write charge current through the Spin Hall Magnet switches the magnetization through SOT mechanism. If the capacitor is charged-'1' (/discharged-'0'), the SHE-MTJ write terminal is set to -Vwr (/Vwr) write voltage. This allows adequate charge current to flow from the write circuit output to the ground (/ground to the inverter output), changing the MTJ state to High-RAP (/Low-RP).

3) LTM to STM transfer: To retrieve the data stored in SHE-MTJ for crossbar computation, an LTM-to-STM mode is considered in the architecture. As shown in Figure 4.5 (b), for this transfer, the BL voltage is first set to $\frac{VDD}{2}$, while SL is grounded. Now, activating the WL (T1: ON), the resistance states i.e. High-RAP (/Low-RP) can be readout by a sensing circuit. The controller activates T3 and T4 transistors and a small read voltage is applied on the SHE-MTJ realizing a voltage divider between its resistance state and a fixed reference resistor. The amplified readout data can accordingly charge (/discharge) the bit-cell capacitor with regard to the control signals in Table 4.1.

4.3. STM-LTM Transition

The proposed STM-LTM architecture is optimized to perform two specific tasks. First, the STM-to-LTM transition is realized with timing constrained by the hardware parameters; existing capacitive networks refresh all cells at a rate determined by the leakiest cell in the device, which is typically around 64ms. Second, LTM-to-STM transition is achieved for computing purposes. To efficiently mimic the biological memory, the sub-array controller should actively keep the count of stimuli (inputs-In_k) received at every *BL*. Therefore, we define an STM-to-LTM threshold (N_{th}) that can be readily adjusted for energy and performance tradeoffs. Algorithm 1 indicates the required procedure to accomplish STM-to-LTM transition and LTM-to-STM retrieval based on a defined time interval for the STM-LTM sub-array controller. The algorithm starts iterating on all the sub-array rows storing binary weights (W_k). As long as the capacitive network has not reached a Refresh Interval (*RI*), the controller counts the input data (In_k) applied to each row and then this data is used to analyze the number of stimuli (N_{st}) with regards to a specified Pulse Interval (*PI*). For example, Figure 4.6 shows a sample *PI (min)* of 20ns for STM-LTM controller and number of stimuli recorded by it (N_{st}=3) [136]. When N_{st} reaches the preset N_{th}, the STM-to-LTM transition is accomplished for each synaptic weight according to the mechanism explained in Section 4.2.2. Therefore, the data will be stored in LTM only when both conditions are met, first the pulse interval of the input is equal or less than the specified minimum pulse interval (*PI (min)*), meaning we are analyzing the data in a specific timeframe and second, the number of stimuli is equal or greater than the specified threshold. On frequent stimulations, the STM-to-LTM transfer can be

successfully accomplished according to rehearsal (reinforcement) shown in Figure 4.1. Additionally, memory decay (forget) is realized by capacitor charge leakage over time.

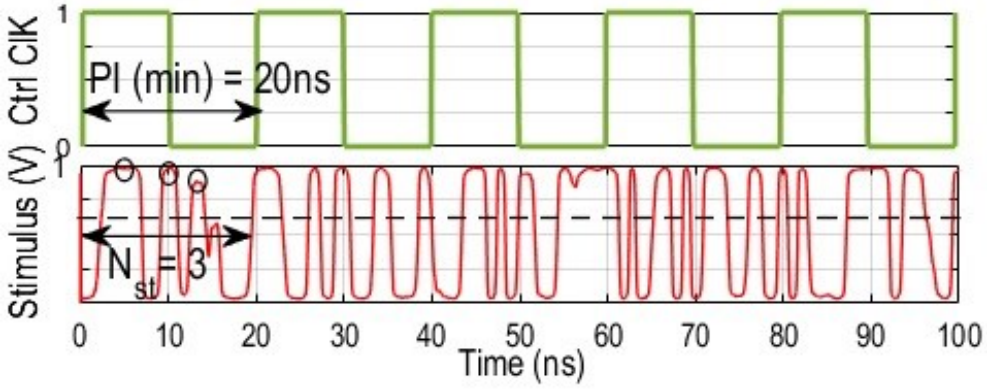


Figure 4.6: A sample pulse interval (PI (min)) of 20ns and number of stimuli recorded by STM-LTM memory controller. When Nst reaches the preset Nth, STM-to-LTM transition is accomplished.

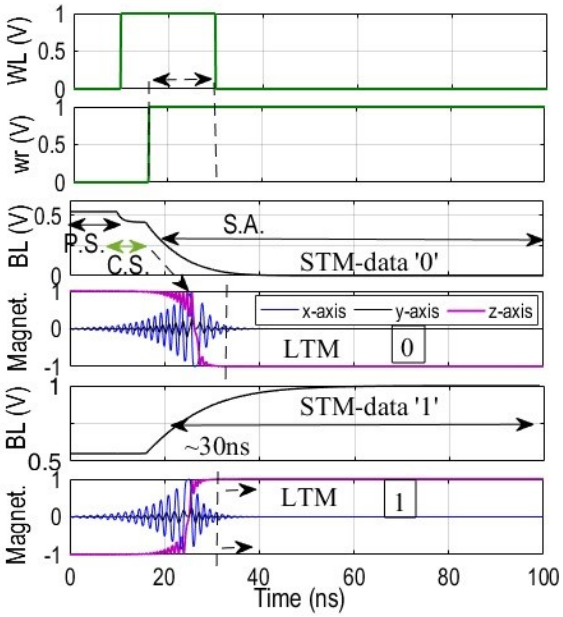


Figure 4.7: The transient simulation results of moving data from STM to LTM. Glossary: P.S., C.S., and S.A. stand for Precharged State, Charge Sharing state and Sense Amplification state.

In the last step, upon arrival of the capacitor refresh interval, the data in LTM will be used to retrieve the capacitor's data according to the mechanism explained in Section 4.2.2. This data will be later used for crossbar computation.

4.4. Simulation Results

4.4.1. Evaluation Setup

We developed a bottom-up simulation framework to evaluate the STM-LTM architecture and estimate its energy and performance tradeoffs. We use STM cell parameters from the Rambus power model [137] with access transistor W/L = 90nm/55nm and capacitance 22fF to evaluate the functionality and performance of our design. We modeled the leakage in SPICE considering a capacitor in parallel with a relatively large-value resistor ($R_{leakage}$) and an equivalent resistance in series (R_{ESR}). The SHE-MTJ electrical model was developed in Verilog-A, which incorporates the Landau-Lifshitz-Gilbert (LLG) equation to model the free layer magnetization dynamics and

Table 4.2. SHE-MTJ simulation Parameters

Parameter	**Value**
MTJ Dimension $W_{MTJ} \times L_{MTJ} \times T_{MTJ}$	$40 \times 120 \times 1.5$ nm^3
SHM Dimension $W_{SHM} \times L_{SHM} \times T_{SHM}$	$120 \times 80 \times 2.8$ nm^3
Demagnetization Factor D_x, D_y, D_z	0.066, 0.911, 0.022
Gilbert Damping Factor, α	0.007
Spin Flip Length, λ_{sf}	1.4 nm
Saturation Magnetization, M_s	850 kA/m
Gyromagnetic Ratio, γ	1.76×10^{11} Am^2/Js
Spin Hall Angle, θ_{SHM}	0.3
Oxide Thickness, t_{ox}	1.3 nm
Energy Barrier, E_a	42 kT
RA Product, RA_p / TMR	22.33 $\Omega \cdot \mu m^2$ / 187.2%
Resistivity, ρ_β-w	200 $\mu\Omega \cdot$ cm
Supply Voltage	1 V
CMOS Technology	45 nm

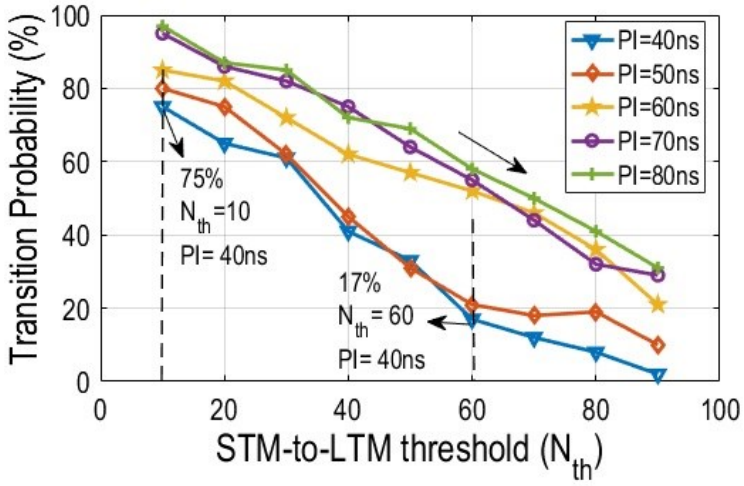


Figure 4.8: The transition probability versus STM to LTM threshold under different pulse intervals.

Non-Equilibrium Green's Function (NEGF) to calculate the resistance range (R_P, R_{AP}) with the device simulation parameters tabulated in Table 4.2. To analyze the VM and NVM modules functionality, we co-designed them in SPICE. Thus, we obtain an analytical approximation to the time-averaged behavior of the full circuit characteristics in 45nm technology node. The controller unit is also simulated by Synopsis Design Compiler [107] with the same technology node. We then modified the NVSIM [138] evaluation tool to report the performance parameters in array-level.

4.4.2. Results

1) Circuit Design: Figure 4.7 shows the transient simulation results of moving data ('0' and '1') from STM to LTM. The *BL* is initially precharged to $\sim\frac{VDD}{2}$ prior to turning on the *WL*. In order to transfer the data into the SHE-MTJ, the controller turns on the corresponding *WL* and the *wr* signals, leading to charge sharing between the *BL* and STM's capacitor. The deviation on the *BL* voltage ($\frac{VDD}{2}\pm\delta$) will be then amplified using the write circuit with bipolar write voltage during

Sense Amplification state (S.A.) as shown in Figure 4.7, to provide the corresponding write voltage for the SHE-MTJ. Su ch voltage allows sufficient charge current to flow in the SHE-MTJ's write terminals and changes free layer magnetization in *z*-axis from +1 to -1 or vice versa, after ~30ns with our memory configuration. Therefore, the VM data is successfully transferred to NVM.

We analyze the STM-to-LTM transition algorithm performance in Section 4.3 with the real random inputs from a probabilistic spin logic neuron referred to as a p-bit device [136]. Such activation function is connected to memory *BLs*. We investigate the transient probability from STM to LTM with different parameters. We first increase the N_{th} from 10 to 90 under a constant *PI* (=40ns) plotted in Figure 4.8. We observe that by increasing the N_{th} the probability of transferring data from STM to LTM reduces. For example, when N_{th}=10, the transition probability is ~75%. However, N_{th}=60 reduces transition probability to ~17% when a larger threshold is desired. Thus, the threshold can be accurately set with regards to the application requirements. We then explore the impact of different *PIs* on STM-to-LTM transition by increasing the expected time from 40ns to 90ns. It can be observed that in a certain N_{th}, by increasing the *PI*, the transition probability will increase.

2) Energy vs. Array Size: In order to compute the energy consumption of the design, we use four different fixed-size capacitive networks (32×32, 64×64, 128×128, and 256×256) leveraging 32, 64, 128 and 256 p-bit output neurons, respectively, to explore the energy consumption of the STM-LTM platform and yield a fair estimate. We analyze the MNIST data-set of handwritten digits with a two-layer perceptron with a net configuration (784×128 as layer 1 and 128×10 as layer 2)

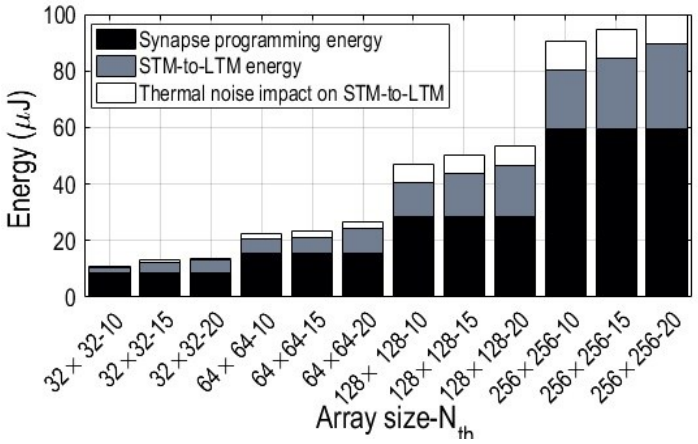


Figure 4.9: The breakdown of energy consumption for different array sizes with the impact of thermal noise.

developed in MATLAB. To assess raw performance, we haven't used any optimization algorithm to map the data into the sub-arrays, so the estimation is solely based on the number of used capacitive crossbars whose performance is given through a bottom-up analysis using our simulation platform. We calculated the average programming energy of the network by dividing the energy of network by total time period per epoch for all training images. The average programming energy of 65pJ is achieved per synapse for a 32×32 crossbar. Thus, the power dissipation of 39pW per synapse is incurred by the network for 1500 images over a time period of 1.1msec per epoch. Figure 4.9 depicts the programming energy as well as STM-to-LTM transfer energy (including controller counting unit) for different array sizes under three various N_{th}. Our first observation is that by increasing the array size under a fixed N_{th}, a larger programming energy is required and the STM-to-LTM energy increases almost linearly. The second observation is that by increasing N_{th}, the STM-to-LTM energy increases due to redundant counting operations. For example, by changing N_{th} from 10 to 15 in 32×32 array, the STM-to-LTM energy increases by

~1.8x. With Figure 4.8 and Figure 4.9, the designer can observe the trade-offs between array size, energy, STM-to-LTM transition probability, etc. to adjust system parameters.

3) Process Variation: We modeled the thermal effects on STM-to-LTM transfer by a randomly fluctuating field, H_{noise} on LTM module, with x, y, and z components from a Gaussian distribution with standard deviation $\sqrt{2\alpha K_B T/\gamma M_s V \Delta t}$ [139] and zero mean. Here, α denotes Gilbert damping factor, KB represents Boltzmann's constant, V denotes the volume of free layer, Ms denotes the saturation magnetization, γ is the gyromagnetic ratio, and Δt represents the time step for solving LLG equation [139, 140]. We carried out the Monte-Carlo simulations with 1,000 iterations introducing a Gaussian spread (σ = 5%) in the SHE-MTJ device parameters Ms and α and thermal effects (300K) in the standard deviation. Under the effect of thermal noise, the switching behavior of the SHE-MTJ changes for different samples. Such change has no adverse impact on the

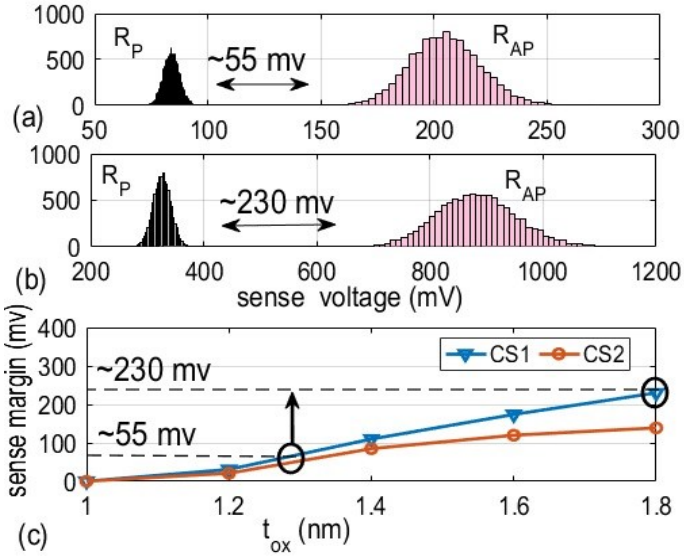


Figure 4.10: (a) Monte-Carlo simulation of sense voltage of SHE-MTJ with (a) tox =1.3nm (b) tox =1.8nm, (c) Voltage margin of SHE-MTJ vs. thickness of MTJ oxide in two case studies.

transition probability of STM-LTM. Based on our observation, the thermal noise increases the energy budget for STM-to-LTM transfer. This energy consumption overhead after applying thermal noise and device variations is shown in Figure 4.9. This comes from the increase in the number of unsuccessful STM-to-LTM transfer.

To assess the variation tolerance of the LTM for different parameters specifically oxide thickness (t_{ox}), we run the Monte-Carlo simulation with 1,000 iterations with 2% Gaussian variation on the Resistance-Area product (RAP) and 5% process variation on the Tunneling-Magnetoresistance Ratio (TMR) and profile the voltage margin between two different resistance level (R_{AP} and R_P), as shown in Figure 4.10 (a). We then increased t_{ox}, from the original 1.3nm to 1.8nm to show how t_{ox} variation impacts the sense margin (Figure 4.10 (b)). We observe the same trend experimentally demonstrated in [40], where the increase in the t_{ox} leads to a higher voltage margin that will considerably enhance the reliability of LTM operation. To further explore the impact of t_{ox} variation, we plotted the voltage margin of SHE-MTJ vs. thickness of MTJ oxide from 1nm to

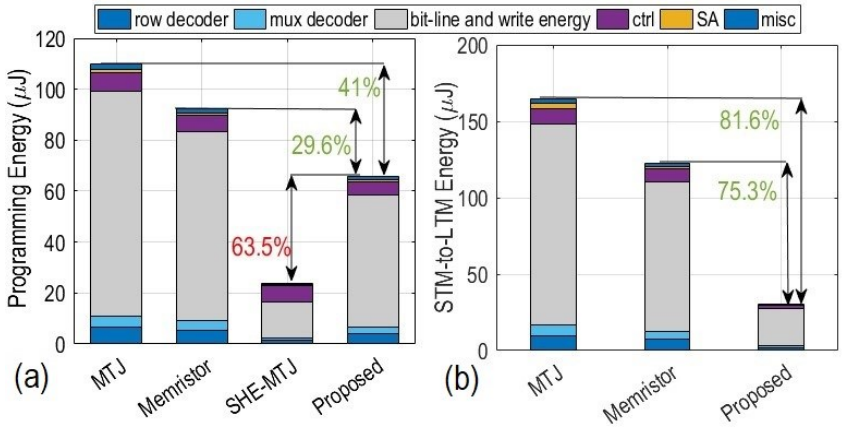


Figure 4.11: The breakdown of (a) Synapse programming energy and (b) STM-to-LTM energy reported in Table 4.3.

1.8nm in two case studies (CSs). The CS1 is under RAP (2%)-TMR (5%) and CS2 is under RAP (5%)-TMR (5%) variation.

Table 4.3. Comparison between STM-LTM architectures

	Sengupta et al. [131]	Srinivasan et al. [133]	Chang et al. [132]	Herein
STM-LTM synapse technology	MTJ	SHE-MTJ/CMOS	Memristor	SHE-MTJ/CMOS
Memory implementation	No	Yes	No	Yes
Separate LTM/STM modules	No	Yes	No	Yes
Compute with STM	No	Yes	No	Yes
Refresh required	No	No	No	Yes
Synapse programming energy (*pJ*)	110	23.7	92.4	65
STM-to-LTM Delay (ns)	~30 on constant stimulation	N/A**	~80 on constant stimulation	~30
STM-to-LTM energy (*pJ*)	165*	N/A**	122.7	~30.2
LTM endurance	$10^{10} – 10^{15}$	$10^{10} – 10^{15}$	$10^{5} – 10^{10}$	$10^{10} – 10^{15}$

* With the 5 input stimulus magnitude of 100μA with 3ns duration

** The STM-to-LTM transfer mechanism is not realized, so the performance cannot be reported.

4.4.3 Energy/Delay Comparison

Table 4.3 compares the STM-LTM platform herein with existing designs in terms of technology, applicability and potentials of a single synapse unit. The listed designs use different methods to implement the STM-LTM transition so different comparison metrics are appropriate. While the MTJ-based [131] and memristor-based [132] synaptic designs demonstrate a single MTJ and memristor mimicking long-term potentiation according to the magnitude, duration, and frequency of input stimulus, the crucial STM state is only a transient state to get to LTM state and not practically useful. The aforementioned designs do not present any circuit implementation to support utilization of STM during computation. Srinivasan et al. [133] presents a fully-functional binary synaptic element that uses two separate SHE-MTJ driven by a relatively different read voltage to improve the synaptic learning efficiency. Separate modules for LTM and STM provides the design with faster and more reliable functionality. To the best of our knowledge, the SHE-MTJ design in [133] is the only design that proposes a practical STM. However, the biological STM-to-LTM transition process was not addressed in detail nor optimized for efficient processing. Our STM-LTM platform brings a solution to make the STM state even more like biological memory by being practically available in the computation phase. Table 4.3 compares different designs in terms of a single synapse programming energy and STM-to-LTM energy. We designed a proper write/read circuitry for MTJ- and memristor-based designs to make them comparable. All designs are implemented with 45 nm technology as well. Based on our evaluation, our design herein consumes ~30.2pJ energy for STM-to-LTM (VM-to-NVM) transfer and ~65pJ for programming (of VM) purposes. The proposed design improves the synapse programming energy consumption by ~29.6% and ~41% compared with memristor and MTJ designs, respectively. The SHE-MTJ

design in [133] achieves the least synapse programming energy consumption (23.7pJ) between all designs. It should be noted that the STM state in our design still incurs capacitive network refresh power. The design herein improves the STM-to-LTM energy over memristor and MTJ by 75.3% and 81.6%, respectively. From STM-to-LTM transition delay perspective, our design requires ~30ns as depicted in Figure 4.7, while memristor and MTJ designs require 80ns and 30ns, respectively, on constant stimulation.

Figure 4.11 shows the breakdown of energy consumption for both programming and STM-to-LTM operations, where the colored legend indicates the contribution of each hardware component to the total programming energy. The synapse programming energy can be mainly translated to write energy for different platforms, as shown in Figure 4.11 (a). The SHE-MTJ intrinsically requires lower write energy compared to the MTJs and Memristors [27]. From STM-to-LTM transition perspective (Figure 4.11 (b)), our design utilizes distinct modules, while the memristor and MTJ-based designs work with consecutive stimulations in the same component leading to a lower STM-to-LTM energy. The two primary influences that impact energy consumption of the proposed STM-LTM design are reading the capacitor's voltage and writing that to the SHE-MTJ.

4.5. Conclusion

Intrinsic computing capabilities provided by hybrid device technology designs offer novel approaches for realizing biologically-inspired features such as consolidation mechanisms present in STM-LTM. The design proposed herein utilizes distinct modules for STM and LTM to realize a synapse contrary to previous designs. This follows biological principles wherein transfer of information to LTM is facilitated through repeated access while providing faster and more reliable

functionality. We then presented a hardware-enabled STM-LTM transition algorithm for the platform considering the real hardware parameters. Our simulations showed the proposed design has the potential of reaching pico-Joule energy level for STM-to-LTM transfer and STM programming.

CHAPTER 5 : ENERGY-EFFICIENT RECURRENT NEURAL NETWORKS

Despite several algorithm-level advances for RNNs, such as Gated Recurrent Unit (GRU) [66] and Long Short-Term Memory (LSTM) [69], there is still a need for an energy-efficient hardware accelerator for such networks. The RNN hardware implementations on FPGA [141], ASIC [142], and GPU [143] have been investigated in prior works. Generally, regardless of the remarkable advances to improve the performance, ANNs based on von-Neumann architecture still face the well-known memory-wall challenge, which results from the limited memory bandwidth, high energy consumption for data movement between memory and processing units, and long memory access latency [144]. To achieve an efficient ANN hardware implementation, Computing-in-Memory (CiM) architectures and mechanisms provide a practical non-von-Neumann infrastructure to increase the parallelism and mitigate the data movement issue, circumventing the memory-wall challenge [145-147]. Various CiM platforms have advanced the computing speed and energy-efficiency significantly and demonstrated extensive data-level parallelism [148]. However, hardware implementation of such designs on top of mature volatile memories (i.e. SRAM/DRAM) requires large complex circuits consuming significant switching energy to execute Multiplication and Accumulation (MAC) and activation functions as the fundamental operations of neural networks [148], [8].

Alternatively, emerging Non-Volatile Memory (NVM) devices such as Spin-Transfer Torque Magnetic Random-Access Memories (STT-MRAMs) [149], Resistive Random-Access Memory (ReRAM) [150], and Phase Change Memory (PCM) [86] have been explored to implement MAC

operation through the intrinsic weighted summation property of CiM cross-bar architecture. Up to now, ReRAM crossbar accelerators have attracted considerable attention due to their high R_{on}/R_{off} ratio ($\sim 10^6$), ultra-low power consumption, and high scalability and switching speed [150, 151] to realize several feedforward neural networks such as Multi-layer Perceptron (MLP), Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs) [18], etc. However, there are only a few works focused on ReRAM based RNNs, mainly due to two challenges. First, realizing a feedback component as an essential part of RNNs requires inevitable write-back operation, which is an inefficient, high latency (>20ns [151]) and energy-consuming operation for NVMs. Second, the accuracy of RNNs heavily depend on the structure of the utilized activation function. A suitable CMOS design for implementing the non-linear sigmoid and tanh activation functions, as the primary thresholding functions used in RNNs, requires significant area and power budget. Their minimization is an important but underexplored concept in neuromorphic computing paradigm. Most of the prior works proposing a hardware for various neural networks utilize CMOS based activation functions with a built-in truth table [71], which impose large area and additional clock cycles to compute the desired function. The RNN implementation in [67] utilizes ReRAM crossbar arrays as synapses along with CMOS-based activation functions. Although this work presents a comprehensive hardware implementation for RNNs and provides an efficient synaptic connection, the CMOS-based neuron is a large compound circuit consisting of four distinct parts. All these parts eventually impose high energy consumption on the overall design. The ReRAM-based CiM architecture for RNNs in [71] provides a detailed design with an exclusive processing engine employing three distinct subarrays for processing the data, including the use of a ReRAM-based crossbar, specialized functional units, and a multiplier. However, the

neuron design still occupies extensive silicon area and relatively high order of energy consumption.

In this chapter, we develop a ReRAM-based RNN and LSTM architecture with feedback using spin-based Adjustable Probabilistic Activation Function (APAF) to achieve high energy- and area-efficiency, while keeping the accuracy loss and processing speed comparable with the baseline designs. The proposed activation function design is based on low energy barrier probabilistic spin logic devices referred to in the literature as probabilistic bits (p-bits) [55].

The remainder of the section is organized as follows. Section 5.1. presents the prior work on activation function unit. Section 5.2. delineates the proposed energy-efficient RNN and LSTM architecture, including the accelerator design, APAF unit and its corresponding algorithm. Section 5.3. details the simulation results including the evaluation framework, comparison and achieved accuracy. Section 5.4. concludes the chapter.

5.1. Prior Work on Activation Function Unit

Hardware implementation of an ideal low-power activation function with small area overhead is one of the challenging research goals in ANNs. There have been various activation function designs proposed for ANNs utilizing both CMOS-based and emerging device-based technologies thus far. However, considering the high number of activation functions employed in each layer of ANNs, these designs still impose high energy consumption or large area overhead and are not

readily suitable for evolving compound multi-layer networks. We briefly study some of these activation functions here. The tanh activation function design in [152] is a CMOS-based stochastic design with Finite State Machines (FSMs) as its building block, aiming to reduce power dissipation

and area overhead by utilizing simpler stochastic arithmetic. However, this design requires long bit-stream lengths generated by Linear Feedback Shift Registers (LFSRs), and CMOS pseudo-random number generators for implementing the probabilistic behavior leading to longer latencies and higher energy consumption. In [153], the authors indicate that implementing a precise sigmoid function leads to excessive area and energy overheads, and therefore, a simplified hardware design based on subsampling and approximation can achieve energy-efficiency while incurring a small

accuracy loss. Although this approach is very practical, the implemented activation function uses logic gates for its approximation unit and a 64×16 lookup table on top of a pseudo random number generator, which still imposes high energy and area overheads. In [67], a CMOS-based activation function consisting of four distinct parts as current generator, function generator, pulse generator and a digital controller is presented with a large circuit footprint and high energy consumption. The special function unit in [71] utilizes the Chebyshev approximation [154] approach, with relatively high power and area compared to the other similar approaches, to implement an approximate tanh activation function. In this method, the CPU initially calculates the coefficients and loads them into the local register. Later, during RNN computing mode, the unit will read the register and calculate the nonlinear function. On the other hand, there are other efforts based on hybrid spin-CMOS p-bit device, which leverage the physical behaviors of nano magnets to perform the computation intrinsically [59]. Although this stochastic activation function offers ultra-low footprint and power consumption, the output of this circuit is probabilistic binary (either "0" or "1"), which is not feasible to be used in RNNs with deterministic sigmoid and tanh functions. Hence, we were motivated to propose a novel activation function based on the p-bit

device with software support, capable of performing non-linear functions in a semi-probabilistic manner to attain good accuracy.

5.2. Proposed RNN Architecture

In this section, we propose an energy-efficient RNN platform with ReRAM crossbar to comprehensively realize a low-latency feedback component and low area-overhead activation function required by this network.

5.2.1 Microarchitectural Design

A detailed representation of the proposed CiM architecture is shown in Figure 5.1. This architecture is essentially developed on top of the 1T1R-resistive main memory architecture [76, 150] by dividing every memory chip into multiple memory banks. Each memory bank is then divided into multiple computational sub-arrays realized using Resistive Crossbars as shown in Figure 5.1. In order to make the ReRAM-based accelerator suitable for RNN computation, we have grouped the resistive crossbar units at the bank level into sets of three interconnected subunits indicated by U-Array, W-Array, and V-Array. Definitions of U, V, W are described in Equation (2.10-2.11). All crossbar units are developed with typical memory peripheral circuits and only differ from an interconnection perspective. Figure 5.2 shows the circuit and interconnection scheme developed for the sub-units. In each crossbar, the Source-Line (SL) is shared amongst the resistive synapses in the same column connected to neurons and the Bit-line (BL) and Word-line (WL) are shared amongst the synapses in the same row. Thus, each synapse in the resistive crossbar is controlled with three signals. The W and V resistive crossbar arrays are connected to a shared

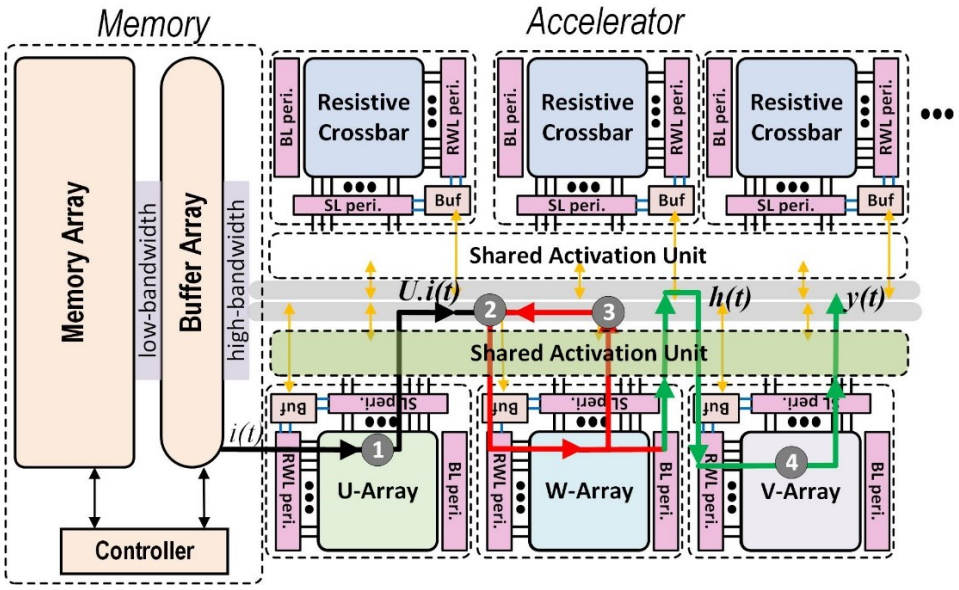


Figure 5.1: The proposed RNN CiM accelerator architecture.

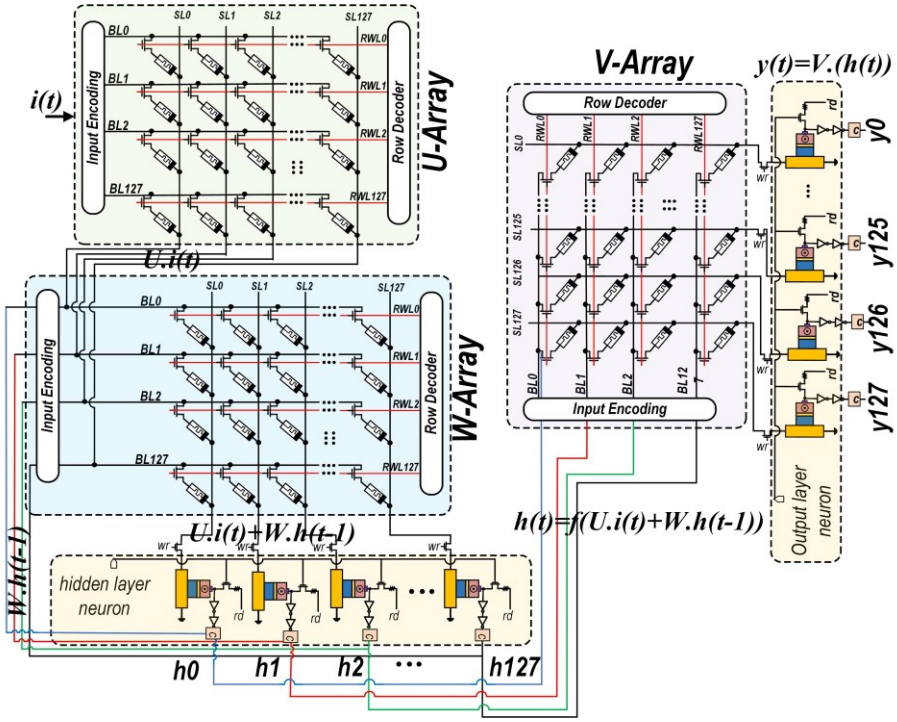


Figure 5.2: The proposed ReRAM-based RNN architecture with stochastic activation functions as neurons.

activation function unit through *SL* peripheral to reduce the area overhead and save energy. However, U-array is solely connected to internal memory bus. A buffer component (Buf in Figure 5.1) is also connected to the *SL* peripherals in all sub-arrays to store the output value before feeding it to the activation functions. As shown in Figure 5.2, the digital input *i(t)* is first converted by the Digital-to-Analog Converter (DAC) using *Input Encoding* component into analog current (I_n) and then is applied to the crossbar. Considering $w_{i,j}$ as the synaptic weight, the dot-product computation is accomplished by every ReRAM crossbar through the intrinsic current-mode weighted summation operation ($\sum_{n=0}^{127} I(n).wi,j$). To realize the RNN's mathematical representation in Equation (2.10-2.11), U-array generates *U.i(t)* dot-product (weighted summation current) in a single memory cycle in the feedforward path (step-1 in Figure 5.1). Then, without converting the current back to voltage, through resistive voltage divider, Analog-to-Digital Converter (ADC) and activating the outputs, such weighted current ($I_{U.i(t)}$) is directed to W-array through the memory bus (step-2 in Figure 5.1). Now, inspired by [67], we designed a new interconnect scheme to direct the activated outputs (*W.h(t-1)*) of hidden layer neurons (W-array) to its inputs to implement the feedback component in RNNs (step-3 in Figure 5.1). In this way, W-array receives *U.i(t)* and *W.h(t-1)* current components and calculates the summation to generate *h(t)* in the second memory cycle leveraging the hidden layer neuron explained in the next sub-section. The *h(t)* is then sent to V-array to generate *y(t)* output represented in Equation (2.11) (step-4 in Figure 5.1).

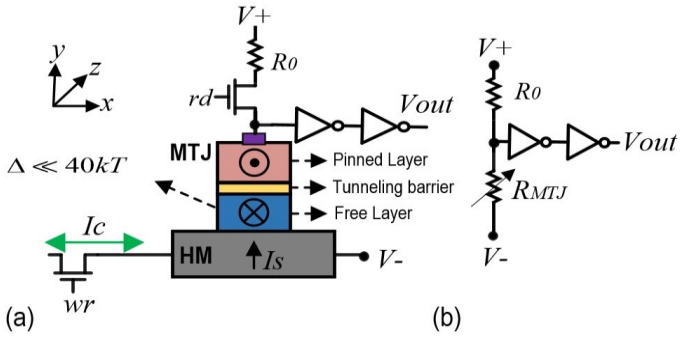


Figure 5.3: The building block of Spin-based activation functions (p-bit) [55].

5.2.2. Adjustable Probabilistic Activation Function (APAF)

Spin-based Building Block (p-bit): The primary building block of the proposed APAF design is the spintronic device described in 5.1.2. providing a novel probabilistic logic (p-bit) [55].

Proposed Design: RNNs and LSTMs utilize the sigmoid and tanh functions for gating purposes in input, output, and forget components. As explained, the p-bit device has an intrinsic probabilistic behavior which follows the sigmoid function behavior in an average time interval. Since the sigmoid function outputs a value between "0" and "1", it can either allow complete flow or no flow of information throughout the gates in RNNs. From the circuit implementation perspective, such sigmoidal behavior can be modulated with the p-bit device by connecting a proper inverter to VDD and GND, as depicted in Figure 5.3 (a). In Figure. 5.4 (b), the black dotted curve indicates the analytical output given by the sigmoid function, $\sigma(z) = 1/(1 + \exp(-z))$, where z is the input and the gree n-circle curve is the p-bit running output average, fitted to the analytical sigmoid function. Similarly, the nonlinear hyperbolic tangent or tanh function output values, which are between "+1" and "-1", could be designed on top of the sigmoidal function mathematically as

$\tanh(x) = 2\sigma(2x) - 1$. This can be readily implemented in the circuit-level by inserting a proper inverter (connected to VDD and -VDD) after sensing the p-bit device resistance. Figure 5.4 (c)

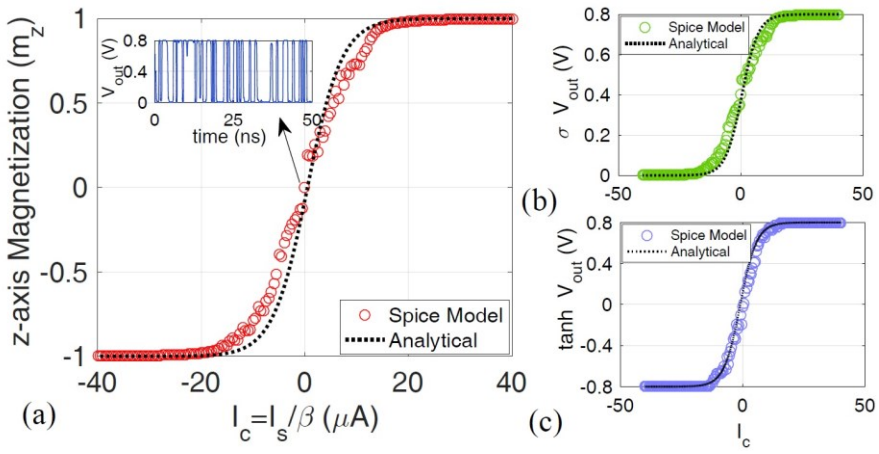


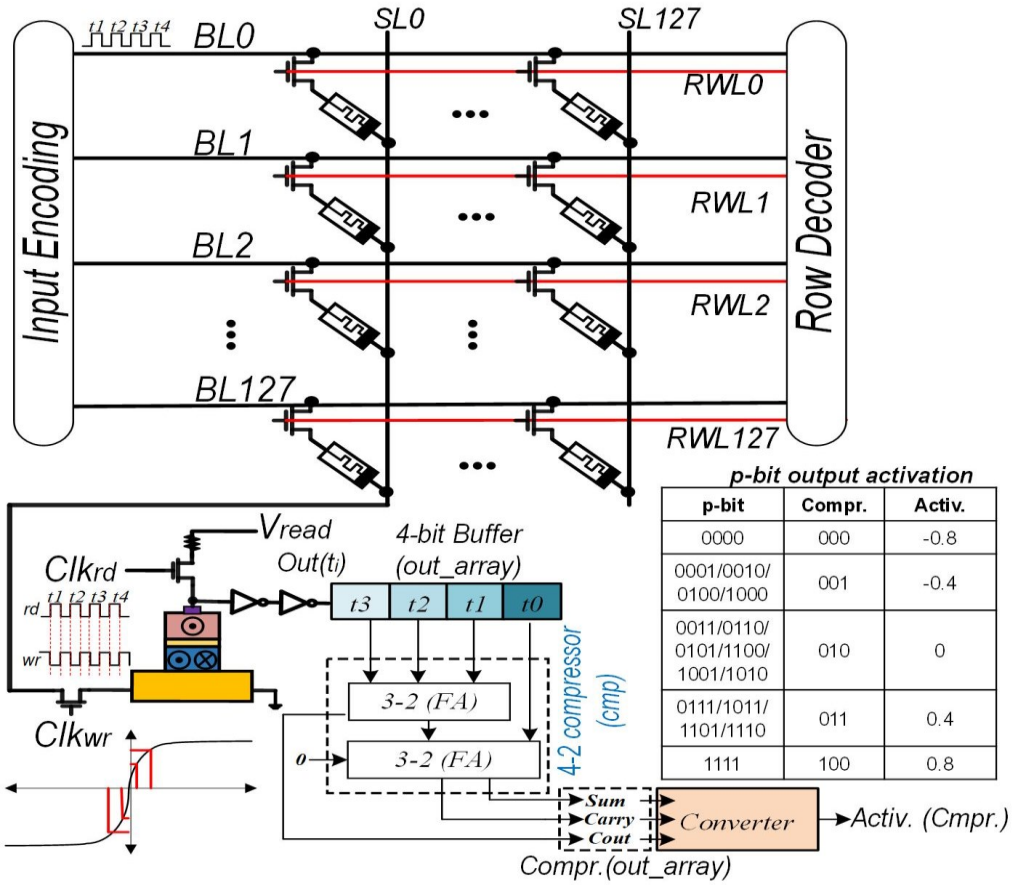
Figure 5.4: Time-averaged behavior of the SHE-MTJ based p-bit device, (a) is the magnetization fluctuations, (b) and (c) are the implemented sigmoid and tanh behaviors respectively.

shows the analytical output values of the tanh function by the dotted black curve along with the time-averaged value of the slightly modified p-bit device's output voltage by the blue-circle curve i.e. *tanh (I_c)*, when the input current increases from negative to positive values. Based on this Figure, at each time step, if the input is zero, the p-bit output takes on a value of "−1" or "+1" with equal probability. A negative input I_c makes negative values more likely and vice versa.

Therefore, the time-averaged output of the p-bit device can provide both sigmoid and tanh function behaviors via slightly different circuit designs. However, in practice, the p-bit device output for each input at a time is a binary "0" or "1" (AP or P). On the other hand, the ideal mathematical sigmoid or tanh functions are not limited to binary states and have a specific output from a limited range for each input number. Training neural networks to high levels of accuracy is one of the major goals of every RNN and the binary output of the p-bit device limits the accuracy of such

networks. Utilizing the p-bit device as a practical activation function, capable of mimicking the ideal mathematical sigmoid or tanh functions with a range of output numbers, requires a novel complementary activation circuit and mechasnism. The key observation to utilize p-bit behavior in order to implement a non-binary activation function is that the stochasticity for a range of input current values close to zero is at the highest level, whereas the stochasticity decreases as the input current values reach to their minimum/maximum levels. The APAF design extracts this behavior with a symmetric range of output voltage numbers by running the p-bit with the same input for multiple time intervals and storing the output for each one. The stored output combinations will be later mapped into a voltage value utilizing a low-overhead Look-Up Table (LUT). This idea allows the p-bit to function in an enhanced non-binary state while maintaining its low-power and low-area properties compared with its CMOS counterpart. For hardware implementation, we enhanced the p-bit stochastic activation function by adding three components, as depicted in Figure 5.5. First, a 2^n-bit buffer (here, 4-bit) is added to latch the output voltage of p-bit circuit (*out_array*). Second, a compressor unit (*cmp*) consisting of CMOS full-adders are leveraged to efficiently sum up and compress the saved binary data in *out_array* (here, 4-2 *cmp*). Third, a LUT is used to eventually generate the activation function output. To avoid multiple crossbar computations, we synchronized the write/read access transistors of the p-bit device. This method provides the ability to maintain a valid crossbar output current and apply it to the activation function unit based on the required number of times. In this way, we consider two complement signals for *wr* and *rd* as shown in Figure 5.5. For every sample, first the *wr* signal goes high and the p-bit device is programmed based on the crossbar output current. Accordingly, the *wr* signal

goes low and the *rd* signal goes high to readout the p-bit resistance and generate the output bit. Moreover, to exploit the full capability of APAF and achieve the full parallelism and input-output



p-bit	Compr.	Activ.
0000	000	-0.8
0001/0010/ 0100/1000	001	-0.4
0011/0110/ 0101/1100/ 1001/1010	010	0
0111/1011/ 1101/1110	011	0.4
1111	100	0.8

Figure 5.5: The proposed Adjustable Probabilistic Activation Function (APAF) design with Al= 5.

Algorithm 1: Adjustable Probabilistic Activation Function (APAF) applied to each p-bit in crossbar array

Leveraging p-bit based approximation: By iteratively applying the input feature maps, the presented method is able to approximate the sigmoid and tanh functions.

input: $fmap_{in:}$ Input feature map, *PI:* Pulse Interval,
A_l: Accuracy Level
output: Activated $fmap_{out}$

1: Initialization: *PI (min),* A_l
2: **if** (*PI* ($fmap_{in}$) > *PI (min)*)
3: $I_{sum} \leftarrow \sum_{\forall i} V(fmap_{in}).Gi$ /*Crossbar compute */
4: **for** $i \leftarrow 0$ to $i < A_l$ /*Iterating based on the A_l */
5: $t_i \leftarrow$pbit (I_{sum}) /*Store in buffer*/
6: *cmp=cmp+* t_i /*Compressing generated bit-stream*/
7: **end for**
8: **else** break
9: **end if**
10: $V(fmap_{out}) \leftarrow Conv(cmp)$
11: return $V(fmap_{out})$

synchronization, we propose the software support in Algorithm 1. Here, Pulse Interval *(PI),* and Accuracy Level (A_l) are regulatory parameters. As we aim to apply the crossbar input to the activation unit multiple times, the algorithm requires to maintain the input current for a specific time window. *PI* enables the controller to issue the read command for inputs in a preset time range and separates each set of inputs based on the system requirements and restrictions. This attribute further enhances the algorithm by restricting the possible noises that can be applied to the system. The A_l parameter is defined to adjust the accuracy level of the activation unit based on the desired performance and tradeoffs.

Given the proposed ReRAM-based RNN architecture shown in Figure 5.2, by having the crossbar input $U.i(t) + W.h(t-1)$, the algorithm first checks the input pulse interval (*PI* ($fmap_{in}$)) to ensure it is greater than the required minimum pulse interval *PI (min)* (line-2). It then, applies the

input currents to compute the weighted summation in W-array and generates probabilistic h(t) output. This is shown in Figure 5.5 for the SL0. In the next step, the algorithm iterates based on the accuracy level and applies the same input to W-array (line-4) for $i = 2^n$ times (rather than only one time) and profiles the p-bit activation function output every time by storing them in a 2^n-bit (here 4-bit) buffer (line-5). Using this method, the accuracy level can be mathematically represented as $A_l = 2^n + 1$. As Figure 5.5 shows, we have exemplified the performance of the system with 5 levels of accuracy as output voltages (-0.8, -0.4, 0, 0.4, 0.8). To reach $A_l = 5, i$ should be equal to 4, meaning 4 iterations are needed. The 4-bit buffered data is then compressed by *cmp* unit and given to the converter (Resistive-LUT). LUT is prestored with the sampled floating-point activation values corresponding to *cmp* output combinations. For example, considering 4 iterations, regardless of p-bit output combination, if the compressed value is 001, the converter selects -0.4 as the output. This could come from either 0001/0010/0100/1000 p-bit output bitstreams. Such APAF design is applicable in a variety of ANN applications needing non-linear and deterministic tanh and sigmoid activation functions.

5.3 Results

5.3.1 Evaluation Framework

To evaluate the performance of the proposed RNN accelerator and perform a fair comparison with state-of-the-art designs, we developed a novel bottom-up evaluation framework, as shown in Figure 5.6. The presented HW/SW cross layer framework starts with device-level modeling of memristive synapse and spin-based p-bit neuron components. We used the SPICE model for memristors with the Ag-Si memristor device parameters from [106]. The SHE-MTJ electrical

model is developed in Verilog-A, which incorporates the Landau Lifshitz–Gilbert (LLG) equation to model the free layer magnetization dynamics and nonequilibrium Green's function (NEGF) to calculate the resistance range (R_P, R_{AP}) with the device parameters tabulated in Table 2.1. We then combine the SPICE models of CMOS transistors and memristors under 14nm PTM-MG library [155]. At the circuit level, we developed crossbar arrays under several sizes (32×32, 64×64, 128×128, 256×256) for the RNN evaluations and developed crossbar arrays under two sizes (32×32, 128×128) for LSTM evaluation with p-bit activation functions in HSPICE. We implemented all peripheral circuits including row address decoders, array controller, etc. in Synopsys Design Compiler [111]. For the architecture level assessment, we extensively modified the MNSim simulator [156] to co-simulate with our developed RNN and LSTM library. This library takes the circuit-level data as input and feeds it into memory level evaluations. We used the resistive parameters in [157] with R_{low} = 315K and R_{high} = 1.1G to assess the latency, area, and energy of the crossbar arrays in MNSim. For the application level performance, we built an image recognition classifier in Pytorch using the MNIST dataset for the RNN design [158]. Such RNN takes an image of hand-written numbers from 0–9 as input and classifies it based on the shape. We also built three distinct name predictor LSTM networks via ideal, binary, and the proposed non-binary APAF neuron, employing the popular names dataset available as national data [159]. For the hardware mapping, every input feature-map is treated as a 2D matrix and then partitioned and mapped to the crossbar array. The mapped data is fed into RNN-enabled MNSim to extract the architecture-level performance parameters as depicted in Figure 5.6.

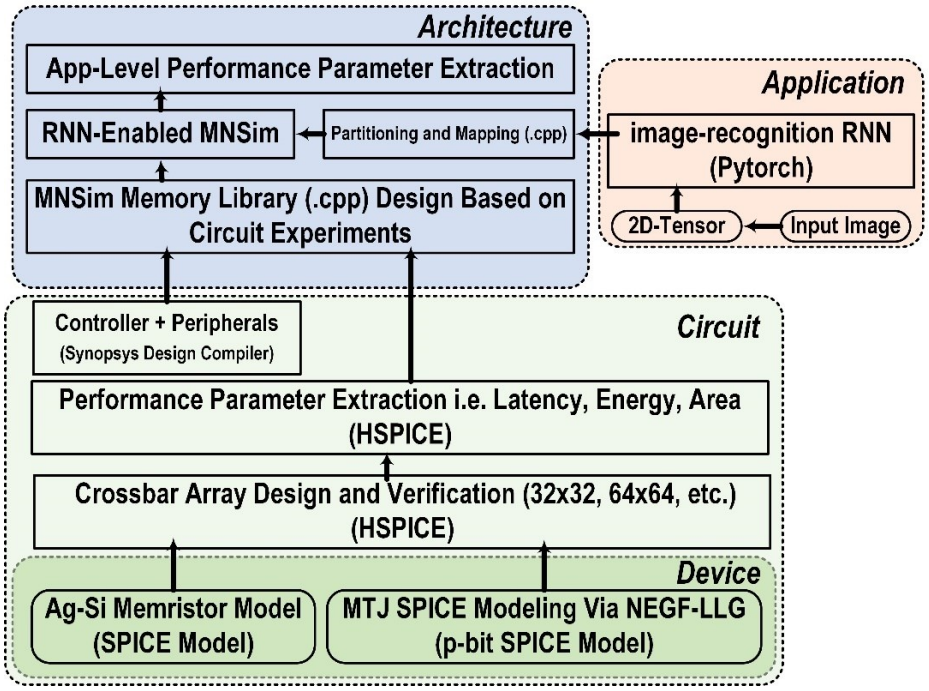


Figure 5.6: HW-SW cross-layer evaluation framework developed in this work.

Bit-Width Setup: We consider three degrees of weight quantization to explore the accuracy of the platform (W=4, 2, 1) with I=3-/5-bit APAF output. Hence , we report the accuracy for 6 bit-width configuration of <I:W> (<3:4>, <5:4>, <3:2>, <5:2>, <3:1>, <5:1>).

Hardware Setup: The under-test RNN structure generally consists of 128 hidden layer neurons and 10 output neurons. It has two linear layers, similar to Figure 2.11, that function over input and hidden states, with APAF design mimicking the tanh function followed by one fully-connected layer with LogSoftmax activation function. This is mapped into ReRAM crossbar units with the proposed layer connectivity in Section 5.2.

5.3.2. Functionality Analysis of APAF

Figure 5.7. shows the SPICE simulation waveforms verifying the functionality of the proposed circuitry for the APAF design. In this Figure, we evaluate the output of APAF's p-bit component four times (labeled by p-bit 1 to p-bit 4) for four consecutive clock cycles, under five different input currents, which later will be mapped into five outputs. Here, I_{sum} represents the weighted summation of input currents realized by the resistive sub-array, ranging from -50µA to +50µA, flowing into the p-bit device. When the I_{sum} is equal to -50µA or +50µA, the output of all four p-bit devices for the entire four clock cycles are "1" and "0", respectively. This indicates the deterministic behavior of the activation function under these charge currents. These outputs will later be mapped to 0.8v and -0.8v, respectively, by the converter. When the I_{sum} is -5µA, we

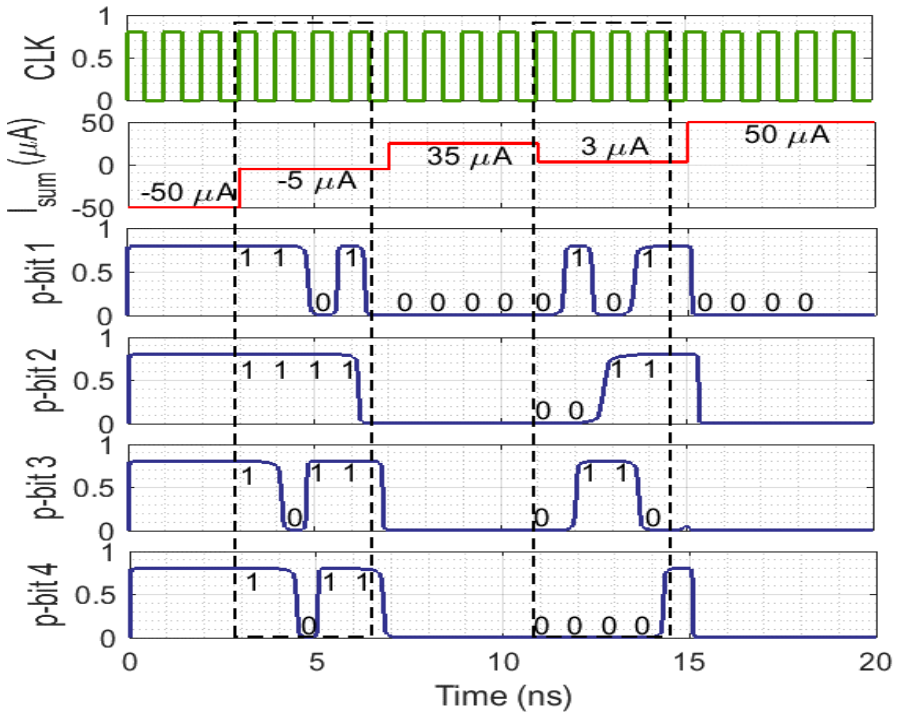


Figure 5.7: The transient simulation result of the neuron w.r.t. the crossbar SL current.

Table 5.1: The p-bit output error rate vs. the APAF error rate.

		Variation (m%)			
Accuracy level	**Probed Outputs**	**±5%**	**±10%**	**±15%**	**±20%**
A_I=3	p-bit	0.00%	0.22%	0.56%	5.90%
	APAF	0.00%	0.00%	0.19%	1.45%
A_I=5	p-bit	0.00%	0.25%	0.55%	5.39%
	APAF	0.00%	0.00%	0.00%	0.28%

observe different outputs for each p-bit device. The p-bit output (p-bit) 1, 3 and 4 have one "0" and three "1"s in four consecutive clock cycles that will be mapped to 0.4v. However, we observe "1111" for the p-bit output 2, indicating a slight error in this case. When the I_{sum} is 35μA the output of all p-bit devices for the entire four clock cycles are "0000", which is again in the deterministic range of the p-bit device as shown in Figure 5.4 (a). Finally, when the I_{sum} is 3μA, we again observe different outputs for the p-bit devices. Accordingly, the p-bit output (p-bit) 1, 2, and 3 will be mapped to 0v and the p-bit output (p-bit) 4 will be mapped to -0.8 as an error.

We extensively analyzed the variation tolerance of the APAF by running a rigorous Monte-Carlo simulation at the sub-array level with 10,000 trials, by adding a σ= 10% variation to crossbar conductance and an *m*% process variation on the Tunnel Magneto-Resistance (TMR) of p-bit device. We reported the calculated error rate in Table 5.1 for both p-bit output and the APAF after the converter. Our first observation is that even considering a typical 10% variation on MTJ's TMR, when p-bit shows an error rate of 0.22% for A_I=3, the APAF design's error rate was found

Table 5.2: The comparison of APAF with CMOS-based designs.

	32x32			**128x128**		
xbar Size	[153]	[152]	Here	[153]	[152]	Here
xbar #	68	68	68	5	5	5
Area (mm^2)	0.17	0.07	0.06	0.06	0.02	0.02
Energy (uJ)	N/A	4.04	0.14	N/A	1.03	0.03

to be 0.00%; thus, the overall functionality provides a reasonable approximation to the tanh function. Our second observation is by increasing the accuracy level from 3 to 5, while the p-bit output error rate does not necessarily change, with a specific *m* value, the APAF error rate remarkably reduces. This mainly stems from the fact that the converter's LUT is now able to convert the compressed value to a specific output level with higher precision. Additionally, we compared the area and energy consumption of the proposed design with [153] and [152] CMOS-based designs, under two distinct sub-array sizes as tabulated in Table 5.2. The simulations show that our proposed neuron achieves up to 34× improvement in energy efficiency and 2× area reduction compared to the CMOS-based non-binary designs. The energy consumption results for [153] could not be appropriately reported.

5.3.3. Application-level Evaluation

To fairly compare this result with other ReRAM accelerators, we implemented the CMOS analog design (represented by D1, henceforth) [152], CMOS digital design (D2) [153], an RNN-enabled Prime [76] platform (D3), and ReRAM RNN design (D4) [67] from scratch in the same technology node as our design using the presented cross-layer platform.

Energy Consumption: Figure 5.8 reports and compares the energy consumption breakdown of the proposed design with the previous works under four distinct sub-array sizes (32×32, 64×64, 128×128, and 256×256) to run MNIST dataset. Based on the experiment, a significant amount of energy (over 87%) is consumed by DAC/ADC/CMOS activation units in all ReRAM implementations. We observe that replacing the large CMOS ADC and activation in counterpart designs with the APAF provides an outstanding energy-saving for the RNN accelerator. In this experiment, by setting the APAF accuracy-level to 5 (A_l=5), on average the proposed design

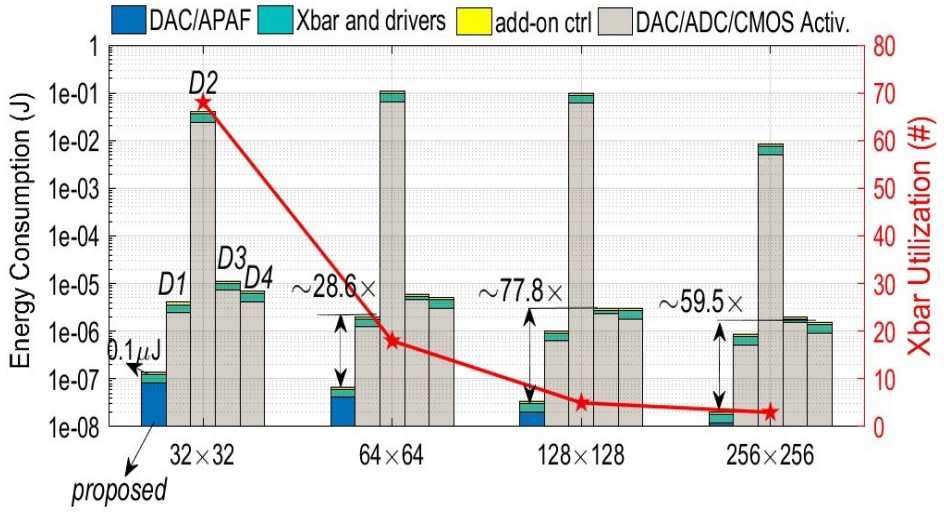


Figure 5.8: Components of energy consumption for ReRAM crossbar designs with various sub-array sizes (note: left y-axis: log-scaled).

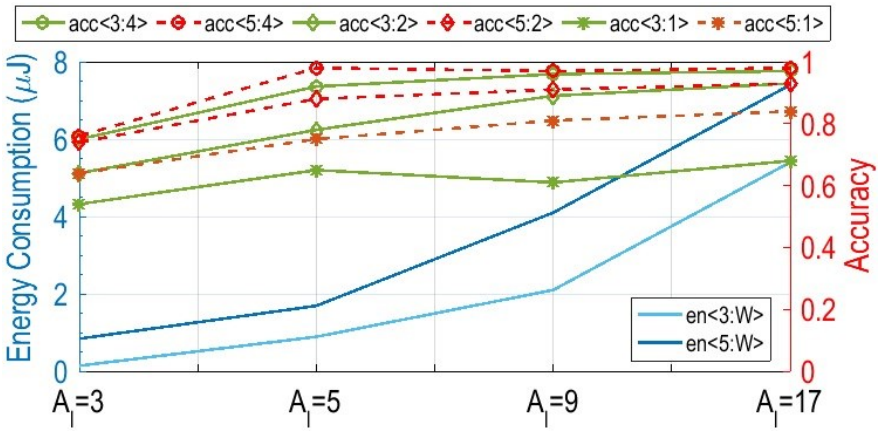


Figure 5.9: Trade-off between energy consumption and accuracy w.r.t. Al on MNIST data-set.

achieves 28.7×, 74.5×, and 51× improvements in term of energy consumption compared with the D1, D3, D4, respectively, while reducing the activation function footprint by a factor of 11 on average over 4 different sub-array sizes. It is worth noting that D1 [152] uses LFSR, bit-wise AND, tree adder, FSM-based, and CMOS tanh as activation function components and D2 [153] utilizes 64×16 LUTs, CMOS pseudo random number generator, and comparator as the main activation components. Moreover, Figure 5.8 shows the correlation between energy consumption and sub-array size. It can be observed that the larger the sub-array size is, the less crossbar utilization and energy budget are required to process the input feature maps.

Accuracy-Energy Trade-offs: We explore the existing trade-off between the overall inference accuracy of the RNN running MNIST dataset and energy consumption of APAF with respect to the A_l, as shown in Figure 5.9. Here, we consider two bit-width configurations for the input i.e. <I>=3-/5-bit (corresponding to the previous layer's APAF output), while quantizing the weight to 1, 2, and 4 bits. The blue curves in this plot are dedicated to demonstrating the energy consumption of APAF unit for 3-/5-bit input bit-width versus A_l. They confirm that an increase in A_l comes at

the cost of higher energy-consumption for the platform, as we are utilizing the APAF a greater number of times. In addition, the green (/red) curves are dedicated to show the accuracy of <I>=3-bit (/5-bit) configuration when <W> changes. Higher weight bit-width provides a higher accuracy for the platform in a particular A_l. Therefore, the higher overall accuracy could be met by increasing A_l. We can see when A_l = 5, <5:4> configuration achieves close to 98%, which is comparable to full precision network accuracy on CNNs [81].

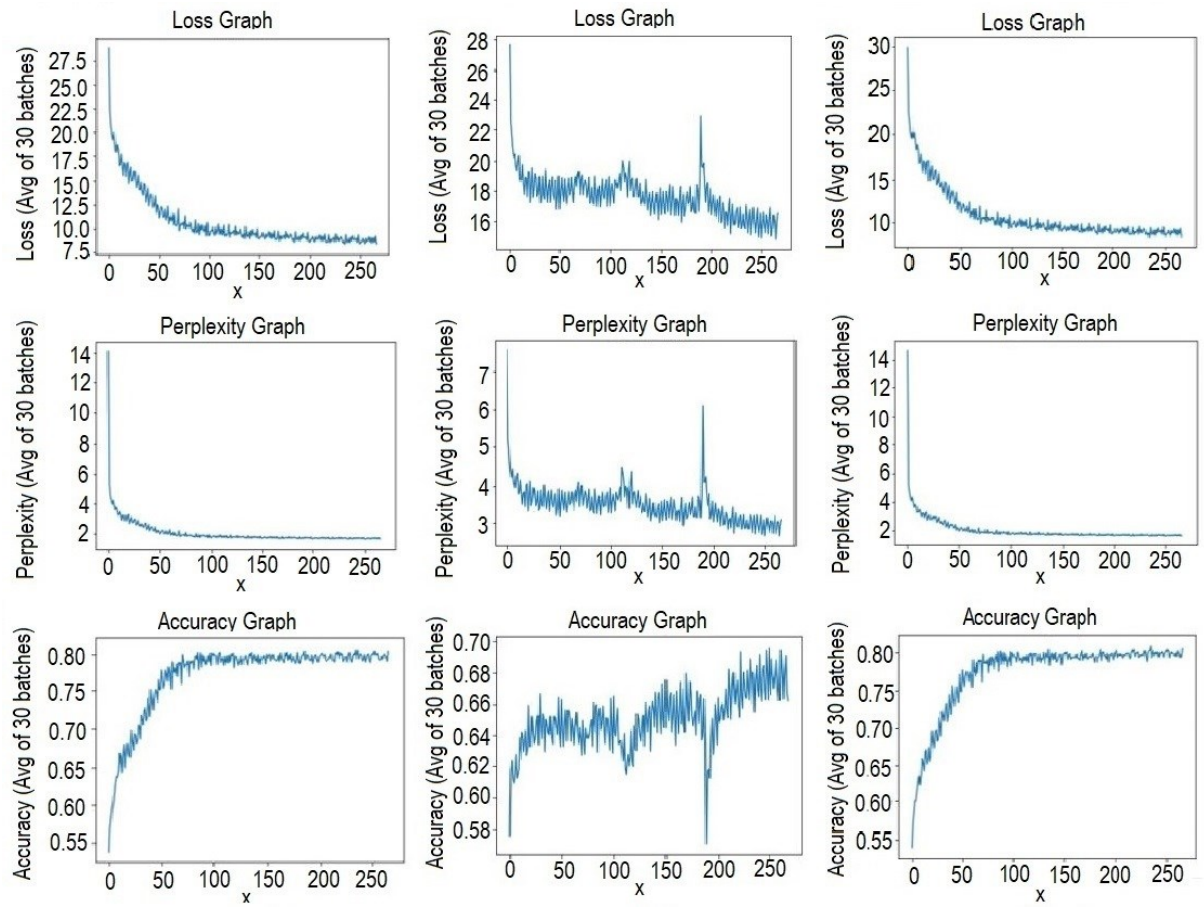


Figure 5.10: The experimental results of the LSTM network with (a) ideal, (b) binary and (c) proposed non-binary APAF-based neuron.

Figure 5.10 shows the experimental results for three distinct neuron designs including loss, perplexity, and accuracy fluctuations for all cases in a name predictor LSTM networks via ideal, binary, and the APAF design, employing the popular names dataset. Unlike accuracy, for loss and perplexity parameters, lower values are preferred. The plotted data is an average of 30 training sample batch sets. The accuracy indicates the performance of the neural network while the perplexity graph evaluates the currently implemented network regarding the sample data modeling. In Fig. 6. (a), the ideal sigmoid neuron displays the limits on possibility with an approximation for all plots. In the binary neuron case shown in Figure 5.10 (b), there is a sharp rise in accuracy in the first sets of batches. However, it initially does not reach the performance of the ideal sigmoidal model (Figure 5.10 (a)). Consequently, the results of the binary case have a long tail that starts around set number 50, in which the system gradually improves as it progresses towards the end of the batches. Additionally, the perplexity graph shows that disturbance from discontinuity of the binary activation causes the training algorithm to struggle in modeling the samples using the network. After 8,000 training samples, the network with the binary neuron shows 58% degradation at modeling the data compared to the ideal sigmoid neuron.

Utilizing the proposed non-binary neuron, the results are very close to the ideal case as shown in Figure 5.10 (c). The enhanced activation mechanism allows it to mimic the ideal sigmoidal system. This is reflected in the perplexity graphs converging to similar values, with the proposed non-binary neuron with only 7% degradation compared to the sigmoidal system. However, the proposed neuron, also starts with a slightly slower training speed, as the binary activation function. But this tail is much shorter, lasting over the course of approximately 1,050 training samples.

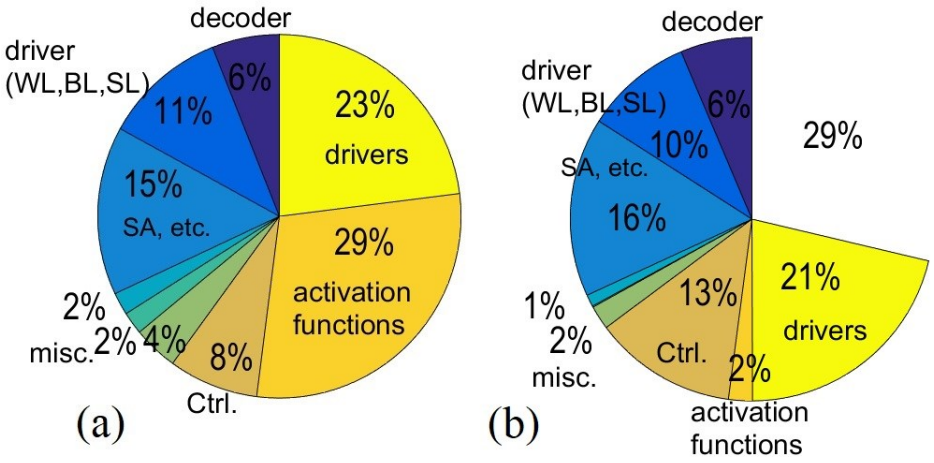


Figure 5.11: Breakdown of area overhead of peripherals for (a) D3 as the base-line and (b) the proposed RNN accelerator with Al= 5.

Area Overhead: To evaluate the area overhead of the presented accelerator on top of commodity ReRAM chip, we took three main hardware cost sources into consideration: (1) hybrid spin-CMOS APAF unit connected to every SL; (2) the ctrl's overhead to adjust regulatory parameters based on APAF algorithm; (3) the output driver's overhead to connect with the shared activation functions and buffer array as depicted in Figure 5.1. The detailed breakdowns of area overhead of D3's design and our presented design are shown in Figure 5.10. We take D3 in Figure 5.10 (a) as the baseline for comparison and assess the area alteration with respect to the different components. As shown in the Figure, D3 requires ~37% area increase (activation functions + ctrl) to support RNN computation. However, the presented design only imposes 15%-20% area overhead, depending on the demanded accuracy level (here A_l= 5).

Latency: An APAF-based mechanism offers significant improvements in terms of energy-efficiency and peripheral footprint and it does not rely on multiple weighted summation computation to realize sigmoidal or tanh functions. Although we have multiple computations for

activation function unit, the added latency to overall computation will be negligible as the crossbar computation will be performed once. Based on our experiments, the crossbar computation latency with APAF is comparable to D1 and D2, due to two prevailing reasons. First, the APAF eliminates the latency of multi-cycle LFSR, tree adder, FSM, and CMOS pseudo random number generator to realize the activation function. The APAF generates the output in a single memory cycle (<1ns). Second, the proposed parallel feedback component combined with APAF discussed earlier actively eliminates the need for the high-latency write-back operation in the previous platform, D1-D4.

Resource Utilization: We further explored the memory bottleneck ratio and resource utilization of different in-memory computing accelerator (D1-D4 and our proposed design) compared with an NVIDIA GTX 1080Ti Pascal GPU with 3,584 CUDA cores running at 1.5GHz. As shown in Figure 5.11, we projected the memory bottleneck as the time fraction at which the crossbar must wait for data and on-/off-chip data communication hinders the overall performance. We extracted the results for each platform considering the number of memory accesses. In this plot, we can see that ReRAM crossbars spend less than 38% time for memory access and data transfer. However, the GPU spends over 90% of its time loading data from the memory. The proposed platform with APAF and D4 achieve the highest utilization ratio by efficiently utilizing up to 88% of the computation resources.

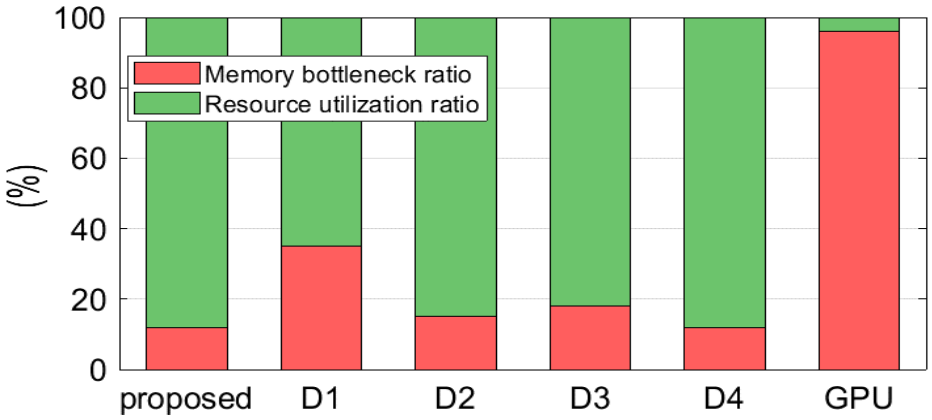


Figure 5.12: Resource utilization ratio for different platforms.

5.4. Conclusion

Energy-efficient hardware design for data-intensive complex networks such as RNNs has been one of the main challenges in this area. To achieve high levels of energy-efficiency, we proposed a comprehensive hardware implementation for RNNs based on CiM architecture. The proposed design is formed upon ReRAM based crossbars and ultra-low power spin-based p-bit devices as the building block for APAF design. The presented simulation results show that the proposed APAF design provides the desired functionality while showing a high tolerance in crossbar conductance variations and process variations. The comparison of the proposed design with recognized state-of-the-art designs shows up to 74.5× improvements in energy consumption, ~11× area reduction in activation function unit, comparable accuracy and latency, and up to 88% resource utilization verifying its advancements.

CHAPTER 6 : CONCLUSION AND FUTURE WORK

As discussed in the previous chapters, recently the neuromorphic computing paradigms have achieved impressive outcomes in the algorithm-level studies. These models perform computationally intensive operations such as Multiplication and Addition (MAC), on large datasets, which requires high power consumption and face the famous memory-wall challenge. In order to overcome these challenges energy-efficient non von Neuman computing architectures are required that can be implemented with low-power devices. Unlike the algorithm-level improvements in various neuromorphic paradigms, there are fewer studies that propose an architecture-level solution. In this proposal, three distinct energy-efficient accelerators have been proposed for GANs, biologically-inspired networks, and recurrent neural networks. On the other hand, there are still several other neural networks that are in dire need of a suitable hardware implementation to increase the efficiency and speed of the training, such as Long-Short Term Memory networks (LSTMs) and Spiking neural networks. These type of networks are based on unsupervised neural networks and are widely used in making predictions based on the temporal sequence of the input data and utilize a memory unit to predict the next data based on the previous input in the memory unit. These networks interact with larger datasets and have a more complex network structure compared to the other artificial neural networks. Therefore, these networks require an efficient hardware implementation, which enables them to train faster and has a built-in non-volatile element as the required memory unit for the feedback component. MRAM-based architecture is a perfect candidate to provide both power efficiency and non-volatility for such designs.

6.1. Technical Summary

In summary, the major contributions in this dissertation can be listed as follows:

1) In the first cross-layer design for GANs, a partial replacement approach is introduced which can find the locations of layers in both G and D networks to be quantized in a way to achieve the best performance, a maximum number of quantized layers and lowest accuracy loss. It can massively reduce the required storage and computational resources in the inference paths with the minimum performance degradation compared to the full-precision model.
2) Further improvement in the performance efficiency of systems such as energy and area reduction are achieved by developing a new approximate arithmetic unit. To avoid unacceptable error in output behaviors a partial approximate computing datapath consisting of a precise adder and an approximate adder is developed.
3) A PIM accelerator is proposed for GAN, namely ApGAN, based on memristor computational subarrays and ultra-low power activation function to efficiently accelerate its training within the non-volatile memory. Moreover, we present a pipeline computation optimization approach to further enhance the training efficiency of ApGAN in hardware level.
4) Finally, the evaluation of system accuracy in different data precision and the system performance in speed and energy are carried out. Applying steps 1 to 3 causes an extensive reduction in energy and area as well, whereas an acceptable accuracy is achieved. Our

experimental results show that it improves the energy-efficiency and speed by ~21× and 35.5× speedup compared with GPU platform.

5) A new binary STM-LTM platform is proposed with composite synapse of SHE-MTJ and a capacitive memory bit-cell to mimic the behavior of biological synapses. Our design realizes the memory potentiation through continual update using STM-to-LTM transfer.
6) We present a hardware-enabled STM-LTM transition algorithm for the platform considering the hardware parameters.
7) We explore the efficiency of the platform running the STM-LTM transition algorithm considering the correlation between energy, array size, and STM-to-LTM threshold.
8) We propose an energy-efficient and high performance CiM platform, based on a new set of circuit-level and micro-architectural techniques with cross-layer (circuit, architecture, and algorithm) co-optimization to implement various RNNs.
9) We introduce novel concept of Adjustable Probabilistic Activation Function (APAF) using a p-bit device as a part of the shared activation unit in conjunction with the core computations within the crossbar. This design extracts the stochastic behavior of the p-bit device to implement the approximate sigmoid and tanh functions within the CiM platform with a high level of energy-efficiency.
10) We propose a customized matching algorithm for the APAF design to regulate the implemented shared activation unit with regards to its hardware constraints.
11) We develop a cross-layer device-to-application evaluation framework to analyze the efficiency and performance of the proposed platform considering the array size and accuracy level to compare those with leading alternative designs in the literature.

6.2. Future Work

The competition between all the different training methods and enhancement approaches in various deep neural network paradigms narrows down to the main goal of this research field which is the inference accuracy. The research in this dissertation can be extended to develop high precision deep neural networks utilizing novel multibit spin devices as its building block. High precision networks are especially important in artificial neural networks that aim to mimic natural neural networks more closely such as SNNs and LSTM networks. In artificial neural networks, high precision computation plays a vital role as the process of mapping and training in the hardware requires extensive quantization methods that leads to a forced accuracy drop.

To implement multibit synapses, we propose utilizing an MRAM-based compound synapse. This SOT-MRAM-based multibit resistive device shown in Figure 6.1, provides a separate read and write path which will reduce the read error rate, since write operation is performed using the spin

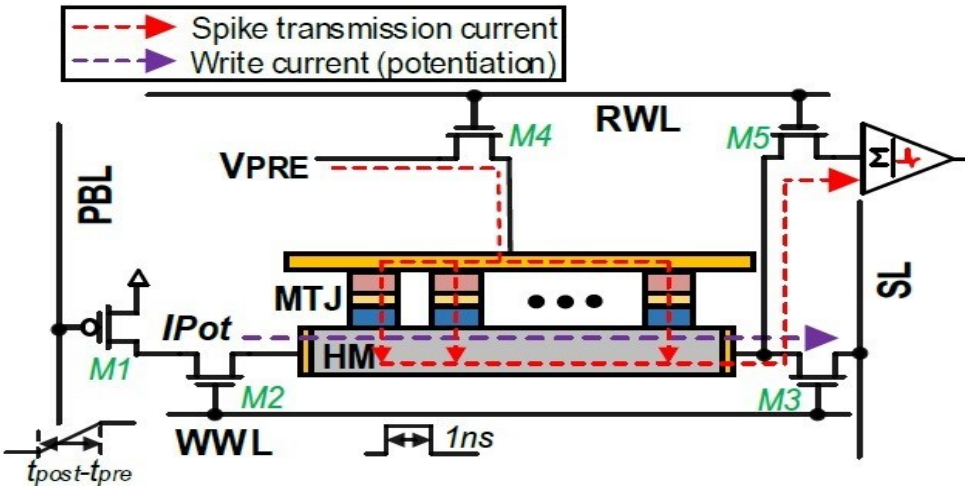


Figure 6.1: Multibit stochastic SOT-MRAM-based.

Hall Effect (SHE) write mechanism [160]. Additionally, SOT-MRAM is expected to perform better in terms of endurance, power consumption, and speed [27].

APPENDIX: COPYRIGHT PERMISSIONS

REFERENCES

[1] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum,* vol. 34, no. 6, pp. 52-59, 1997.

[2] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS scaling for high performance and low power-the next ten years," *Proceedings of the IEEE,* vol. 83, no. 4, pp. 595-606, 1995.

[3] S. Angizi and D. Fan, "ReDRAM: A Reconfigurable Processing-in-DRAM Platform for Accelerating Bulk Bit-Wise Operations," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1-8: IEEE.

[4] S. Salehi Mobarakeh, "Energy-Efficient Signal Conversion and In-Memory Computing using Emerging Spin-based Devices," 2020.

[5] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems,* vol. 35, no. 1, pp. 1-22, 2015.

[6] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proceedings of the IEEE,* vol. 101, no. 12, pp. 2498-2533, 2013.

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

[8] S. Sheikhfaal and R. F. DeMara, "Short-Term Long-Term Compute-In-Memory Architecture: A Hybrid Spin/CMOS Approach Supporting Intrinsic Consolidation," *IEEE Journal on Exploratory Solid-State Computational Devices Circuits,* 2020.

[9] A. Roohi, "Normally-off computing design methodology using spintronics: from device to architectures," in *2020 11th International Green and Sustainable Computing Workshops (IGSC)*, 2020, pp. 1-4: IEEE.

[10] D. Silver *et al.*, "Mastering the game of go without human knowledge," *nature,* vol. 550, no. 7676, pp. 354-359, 2017.

[11] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE,* vol. 107, no. 8, pp. 1655-1674, 2019.

[12] W. Shi and S. Dustdar, "The promise of edge computing," *Computer,* vol. 49, no. 5, pp. 78-81, 2016.

[13] M. Horowitz, "Energy table for 45nm process," in *Stanford VLSI wiki*, 2014.

[14] S. Angizi and D. Fan, "Imc: energy-efficient in-memory convolver for accelerating binarized deep neural network," in *Proceedings of the Neuromorphic Computing Symposium*, 2017, pp. 1-8.

[15] S. Angizi and D. Fan, "Graphide: A graph processing accelerator leveraging in-dram-computing," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019, pp. 45-50.

[16] S. Angizi, Z. He, F. Parveen, and D. Fan, "Rimpa: A new reconfigurable dual-mode in-memory processing architecture with spin hall effect-driven domain wall motion device," in *2017 IEEE Computer Society annual symposium on VLSI (ISVLSI)*, 2017, pp. 45-50: IEEE.

[17] Z. He, S. Angizi, and D. Fan, "Exploring STT-MRAM based in-memory computing paradigm with application of image edge extraction," in *2017 IEEE International Conference on Computer Design (ICCD)*, 2017, pp. 439-446: IEEE.
[18] A. Roohi, S. Sheikhfaal, S. Angizi, D. Fan, and R. F. DeMara, "ApGAN: Approximate GAN for Robust Low Energy Learning From Imprecise Components," *IEEE Transactions on Computers,* vol. 69, no. 3, pp. 349-360, 2019.
[19] J. C. Slonczewski, "Current-driven excitation of magnetic multilayers," *Journal of Magnetism and Magnetic Materials,* vol. 159, no. 1-2, pp. L1-L7, 1996.
[20] 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.
[21] S. Wolf *et al.*, "Spintronics: a spin-based electronics vision for the future," *science,* vol. 294, no. 5546, pp. 1488-1495, 2001.
[22] I. Prejbeanu *et al.*, "Thermally assisted MRAM," *Journal of Physics: Condensed Matter,* vol. 19, no. 16, p. 165218, 2007.
[23] T. Devolder, C. Chappert, J. Katine, M. Carey, and K. Ito, "Distribution of the magnetization reversal duration in subnanosecond spin-transfer switching," *Physical Review B,* vol. 75, no. 6, p. 064402, 2007.
[24] H. Zhao *et al.*, "Low writing energy and sub nanosecond spin torque transfer switching of in-plane magnetic tunnel junction for spin torque transfer random access memory," *Journal of Applied Physics,* vol. 109, no. 7, p. 07C720, 2011.
[25] D. Fan, S. Maji, K. Yogendra, M. Sharad, and K. Roy, "Injection-locked spin hall-induced coupled-oscillators for energy efficient associative computing," *IEEE Transactions on Nanotechnology,* vol. 14, no. 6, pp. 1083-1093, 2015.
[26] W. Kang, Z. Wang, Y. Zhang, J.-O. Klein, W. Lv, and W. Zhao, "Spintronic logic design methodology based on spin Hall effect–driven magnetic tunnel junctions," *Journal of Physics D: Applied Physics,* vol. 49, no. 6, p. 065008, 2016.
[27] S. Manipatruni, D. E. Nikonov, and I. A. Young, "Energy-delay performance of giant spin Hall effect switching for dense magnetic memory," *Applied Physics Express,* vol. 7, no. 10, p. 103001, 2014.
[28] C. Chappert, A. Fert, and F. N. Van Dau, "The emergence of spin electronics in data storage," *Nanoscience Technology: A Collection of Reviews from Nature Journals,* pp. 147-157, 2010.
[29] A. Roohi, "Normally-Off Computing Design Methodology Using Spintronics: From Devices to Architectures," 2019.
[30] M. N. Baibich *et al.*, "Giant magnetoresistance of (001) Fe/(001) Cr magnetic superlattices," *Physical review letters,* vol. 61, no. 21, p. 2472, 1988.
[31] G. Binasch, P. Grünberg, F. Saurenbach, and W. Zinn, "Enhanced magnetoresistance in layered magnetic structures with antiferromagnetic interlayer exchange," *Physical review B,* vol. 39, no. 7, p. 4828, 1989.
[32] J. S. Moodera, L. R. Kinder, T. M. Wong, and R. Meservey, "Large magnetoresistance at room temperature in ferromagnetic thin film tunnel junctions," *Physical review letters,* vol. 74, no. 16, p. 3273, 1995.
[33] J. Rowell, M. Gurvitch, and J. Geerk, "Modification of tunneling barriers on Nb by a few monolayers of Al," *Physical Review B,* vol. 24, no. 4, p. 2278, 1981.

[34] S. S. Parkin *et al.*, "Giant tunnelling magnetoresistance at room temperature with MgO (100) tunnel barriers," *Nature materials,* vol. 3, no. 12, pp. 862-867, 2004.
[35] D. Wang, C. Nordman, J. M. Daughton, Z. Qian, and J. Fink, "70% TMR at room temperature for SDT sandwich junctions with CoFeB as free and reference layers," *IEEE Transactions on Magnetics,* vol. 40, no. 4, pp. 2269-2271, 2004.
[36] W. Pratt Jr, S.-F. Lee, J. Slaughter, R. Loloee, P. Schroeder, and J. Bass, "Perpendicular giant magnetoresistances of Ag/Co multilayers," *Physical review letters,* vol. 66, no. 23, p. 3060, 1991.
[37] M. Julliere, "Tunneling between ferromagnetic films," *Physics letters A,* vol. 54, no. 3, pp. 225-226, 1975.
[38] Q. Xu, H. Chen, J. Lu, G. NI, and Y. DU, "Giant Magnetic Tunneling Effect in Fe/Al~ 2O~ 3/Fe," *Journal of Functional Materials,* vol. 31, no. SUPP, pp. 42-42, 2000.
[39] S. Yuasa, A. Fukushima, H. Kubota, Y. Suzuki, and K. Ando, "Giant tunneling magnetoresistance up to 410% at room temperature in fully epitaxial Co⁄ Mg O⁄ Co magnetic tunnel junctions with bcc Co (001) electrodes," *Applied Physics Letters,* vol. 89, no. 4, p. 042505, 2006.
[40] S. Yuasa, T. Nagahama, A. Fukushima, Y. Suzuki, and K. Ando, "Giant room-temperature magnetoresistance in single-crystal Fe/MgO/Fe magnetic tunnel junctions," *Nature materials,* vol. 3, no. 12, pp. 868-871, 2004.
[41] S. Ikeda *et al.*, "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. 082508, 2008.
[42] W. J. Gallagher and S. S. Parkin, "Development of the magnetic tunnel junction MRAM at IBM: From first junctions to a 16-Mb MRAM demonstrator chip," *IBM Journal of Research Development,* vol. 50, no. 1, pp. 5-23, 2006.
[43] T. L. Gilbert, "A Lagrangian formulation of the gyromagnetic equation of the magnetization field," *Phys. Rev.,* vol. 100, p. 1243, 1955.
[44] A. Brataas, A. D. Kent, and H. Ohno, "Current-induced torques in magnetic materials," *Nature materials,* vol. 11, no. 5, pp. 372-381, 2012.
[45] D. C. Ralph and M. D. Stiles, "Spin transfer torques," *Journal of Magnetism and Magnetic Materials,* vol. 320, no. 7, pp. 1190-1216, 2008.
[46] J. Grollier *et al.*, "Spin-polarized current induced switching in Co/Cu/Co pillars," *Applied Physics Letters,* vol. 78, no. 23, pp. 3663-3665, 2001.
[47] J. Hayakawa *et al.*, "Current-induced magnetization switching in MgO barrier based magnetic tunnel junctions with CoFeB/Ru/CoFeB synthetic ferrimagnetic free layer," *Japanese journal of applied physics,* vol. 45, no. 10L, p. L1057, 2006.
[48] Y. Huai, F. Albert, P. Nguyen, M. Pakala, and T. Valet, "Observation of spin-transfer switching in deep submicron-sized and low-resistance magnetic tunnel junctions," *Applied Physics Letters,* vol. 84, no. 16, pp. 3118-3120, 2004.
[49] H. Zhao *et al.*, "Sub-200 ps spin transfer torque switching in in-plane magnetic tunnel junctions with interface perpendicular anisotropy," *Journal of Physics D: Applied Physics,* vol. 45, no. 2, p. 025001, 2011.

[50] Z. Wang, W. Zhao, E. Deng, J.-O. Klein, and C. Chappert, "Perpendicular-anisotropy magnetic tunnel junction switched by spin-Hall-assisted spin-transfer torque," *Journal of Physics D: Applied Physics,* vol. 48, no. 6, p. 065001, 2015.
[51] L. Liu, C.-F. Pai, D. Ralph, and R. Buhrman, "Magnetic oscillations driven by the spin Hall effect in 3-terminal magnetic tunnel junction devices," *Physical review letters,* vol. 109, no. 18, p. 186602, 2012.
[52] L. Liu, T. Moriyama, D. Ralph, and R. Buhrman, "Spin-torque ferromagnetic resonance induced by the spin Hall effect," *Physical review letters,* vol. 106, no. 3, p. 036601, 2011.
[53] C.-F. Pai, L. Liu, Y. Li, H. Tseng, D. Ralph, and R. Buhrman, "Spin transfer torque devices utilizing the giant spin Hall effect of tungsten," *Applied Physics Letters,* vol. 101, no. 12, p. 122404, 2012.
[54] A. Roohi, R. Zand, D. Fan, and R. F. DeMara, "Voltage-based concatenatable full adder using spin hall effect switching," *IEEE transactions on computer-aided design of integrated circuits systems,* vol. 36, no. 12, pp. 2134-2138, 2017.
[55] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, "Stochastic p-bits for invertible logic," *Physical Review X,* vol. 7, no. 3, p. 031014, 2017.
[56] R. Andrawis, A. Jaiswal, and K. Roy, "Design and comparative analysis of spintronic memories based on current and voltage driven switching," *IEEE Transactions on Electron Devices,* vol. 65, no. 7, pp. 2682-2693, 2018.
[57] H. Pourmeidani, S. Sheikhfaal, R. Zand, and R. F. DeMara, "Probabilistic interpolation recoder for energy-error-product efficient DBNs with p-bit devices," *IEEE Transactions on Emerging Topics in Computing,* 2020.
[58] S. Sheikhfaal, S. D. Pyle, S. Salehi, and R. F. DeMara, "An Ultra-Low Power Spintronic Stochastic Spiking Neuron with Self-Adaptive Discrete Sampling," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2019, pp. 49-52: IEEE.
[59] R. Zand, K. Y. Camsari, S. D. Pyle, I. Ahmed, C. H. Kim, and R. F. DeMara, "Low-energy deep belief networks using intrinsic sigmoidal spintronic-based probabilistic neurons," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 15-20.
[60] K. Y. Camsari, S. Ganguly, and S. Datta, "Modular approach to spintronics," *Scientific reports,* vol. 5, p. 10571, 2015.
[61] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:.06434,* 2015.
[62] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:.03498,* 2016.
[63] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, 2017, pp. 214-223: PMLR.
[64] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:.00028,* 2017.
[65] M. Song, J. Zhang, H. Chen, and T. Li, "Towards efficient microarchitectural design for accelerating unsupervised gan-based deep learning," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 66-77: IEEE.
[66] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:.* 2014.

[67] Y. Long, E. M. Jung, J. Kung, and S. Mukhopadhyay, "Reram crossbar based recurrent neural network for human activity detection," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 939-946: IEEE.

[68] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks,* vol. 5, no. 2, pp. 157-166, 1994.

[69] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, and l. systems, "LSTM: A search space odyssey," *IEEE transactions on neural networks,* vol. 28, no. 10, pp. 2222-2232, 2016.

[70] K. Smagulova, O. Krestinskaya, A. P. James, and S. Processing, "A memristor-based long short term memory circuit," *Analog Integrated Circuits Signal Processing,* vol. 95, no. 3, pp. 467-472, 2018.

[71] Y. Long, T. Na, and S. Mukhopadhyay, "ReRAM-based processing-in-memory architecture for recurrent neural network acceleration," *IEEE Transactions on Very Large Scale Integration Systems,* vol. 26, no. 12, pp. 2781-2794, 2018.

[72] J. Brownlee, "A gentle introduction to the rectified linear unit (ReLU)," *Machine learning mastery,* vol. 6, 2019.

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* vol. 25, pp. 1097-1105, 2012.

[74] 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:.06160,* 2016.

[75] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*, 2016, pp. 525-542: Springer.

[76] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News,* vol. 44, no. 3, pp. 27-39, 2016.

[77] M. Imani, Y. Kim, and T. Rosing, "Mpim: Multi-purpose in-memory processing using configurable resistive memory," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2017, pp. 757-763: IEEE.

[78] F. Chen, L. Song, and Y. Chen, "Regan: A pipelined reram-based accelerator for generative adversarial networks," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 178-183: IEEE.

[79] B. Li, L. Song, F. Chen, X. Qian, Y. Chen, and H. H. Li, "Reram-based accelerator for deep learning," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 815-820: IEEE.

[80] S. Angizi, Z. He, and D. Fan, "PIMA-logic: a novel processing-in-memory architecture for highly flexible and energy-efficient logic computation," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1-6.

[81] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "Cmp-pim: an energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1-6.

[82] S. Angizi, J. Sun, W. Zhang, and D. Fan, "GraphS: A graph processing accelerator leveraging SOT-MRAM," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 378-383: IEEE.
[83] A. Roohi, S. Angizi, D. Fan, and R. F. DeMara, "Processing-in-memory acceleration of convolutional neural networks for energy-effciency, and power-intermittency resilience," in *20th International Symposium on Quality Electronic Design (ISQED)*, 2019, pp. 8-13: IEEE.
[84] A. Roohi and R. F. DeMara, "NV-clustering: Normally-off computing using non-volatile datapaths," *IEEE Transactions on Computers,* vol. 67, no. 7, pp. 949-959, 2018.
[85] A. Roohi and R. F. DeMara, "PARC: A novel design methodology for power analysis resilient circuits using spintronics," *IEEE Transactions on Nanotechnology,* vol. 18, pp. 885-889, 2019.
[86] G. W. Burr *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.
[87] S. Angizi, N. A. Fahmi, W. Zhang, and D. Fan, "PIM-Assembler: A processing-in-memory platform for genome assembly," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1-6: IEEE.
[88] D. Reis *et al.*, "Modeling and benchmarking computing-in-memory for design space exploration," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 39-44.
[89] L. Yang, S. Angizi, and D. Fan, "A flexible processing-in-memory accelerator for dynamic channel-adaptive deep neural networks," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 313-318: IEEE.
[90] I. J. Goodfellow *et al.*, "Generative adversarial networks," *arXiv preprint arXiv:.* 2014.
[91] M. Zieba, P. Semberecki, T. El-Gaaly, and T. Trzcinski, "Bingan: Learning compact binary descriptors with a regularized gan," *arXiv preprint arXiv:.06778,* 2018.
[92] Y. Cao, G. W. Ding, K. Y.-C. Lui, and R. Huang, "Improving GAN training via binarized representation entropy (BRE) regularization," *arXiv preprint arXiv:.03644,* 2018.
[93] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen, "Binary generative adversarial networks for image retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32, no. 1.
[94] J. Liu, J. Zhang, Y. Ding, X. Xu, M. Jiang, and Y. Shi, "PBGen: Partial Binarization of Deconvolution-Based Generators for Edge Intelligence," *arXiv preprint arXiv:.09153,* 2018.
[95] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems,* vol. 32, no. 1, pp. 124-137, 2012.
[96] N. Compute. (2010). *PTX: Parallel thread execution ISA version 2.3*. Available: Dostopno na: http://developer. download. nvidia. com/compute/cuda
[97] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.
[98] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:.07747,* 2017.

[99] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript,* vol. 40, no. 7, pp. 1-9, 2010.
[100] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215-223: JMLR Workshop and Conference Proceedings.
[101] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730-3738.
[102] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv preprint arXiv:.01844,* 2015.
[103] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:.01973,* 2018.
[104] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818-2826.
[105] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:.04711,* 2016.
[106] L. Gao, F. Alibart, and D. B. Strukov, "Analog-input analog-weight dot-product operation with Ag/a-Si/Pt memristive devices," in *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, 2012, pp. 88-93: IEEE.
[107] (2011). *Design Compiler, Ncsu eda freepdk45*. Available: https://www.eda.ncsu.edu/wiki/FreePDK45:Contents
[108] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 31, no. 7, pp. 994-1007, 2012.
[109] A. S. Rakin, S. Angizi, Z. He, and D. Fan, "PIM-TGAN: A processing-in-memory accelerator for ternary generative adversarial networks," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 266-273: IEEE.
[110] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 236-241: IEEE.
[111] P. Kurup and T. Abbasi, *Logic synthesis using Synopsys®*. Springer Science & Business Media, 2012.
[112] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 33-38: IEEE.
[113] M. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," *IEE Proceedings-Computers Digital Techniques,* vol. 150, no. 6, pp. 403-411, 2003.
[114] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 15-24.

[115] S. Angizi, Z. He, F. Parveen, and D. Fan, "IMCE: Energy-efficient bit-wise in-memory convolution engine for deep neural network," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 111-116: IEEE.
[116] S. Angizi, Z. He, N. Bagherzadeh, and D. Fan, "Design and evaluation of a spintronic in-memory processing platform for nonvolatile data encryption," *IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems,* vol. 37, no. 9, pp. 1788-1801, 2017.
[117] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience,* vol. 5, p. 73, 2011.
[118] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a Hebbian learning rule," *Annu. Rev. Neurosci.*

, vol. 31, pp. 25-46, 2008.
[119] R. Lamprecht and J. LeDoux, "Structural plasticity and memory," *Nature Reviews Neuroscience,* vol. 5, no. 1, pp. 45-54, 2004.
[120] I. Winkler and N. Cowan, "From sensory to long-term memory: evidence from auditory memory reactivation studies," *Experimental psychology,* vol. 52, no. 1, pp. 3-20, 2005.
[121] J. L. McGaugh, "Memory-a century of consolidation," *Science,* vol. 287, no. 5451, pp. 248-251, 2000.
[122] B. L. Jackson *et al.*, "Nanoscale electronic synapses using phase change devices," *ACM Journal on Emerging Technologies in Computing Systems,* vol. 9, no. 2, pp. 1-20, 2013.
[123] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *2008 IEEE international symposium on nanoscale architectures*, 2008, pp. 85-92: Ieee.
[124] A. Sengupta, Z. Al Azim, X. Fong, and K. Roy, "Spin-orbit torque induced spike-timing dependent plasticity," *Applied Physics Letters,* vol. 106, no. 9, p. 093704, 2015.
[125] U. Cilingiroglu, "A purely capacitive synaptic matrix for fixed-weight neural networks," *IEEE transactions on circuits and systems,* vol. 38, no. 2, pp. 210-217, 1991.
[126] W. E. Engeler, "Neural net using capacitive structures connecting input lines and differentially sensed output line pairs," ed: Google Patents, 1993.
[127] D. Kwon and I.-Y. Chung, "Capacitive Neural Network Using Charge-Stored Memory Cells for Pattern Recognition Applications," *IEEE Electron Device Letters,* vol. 41, no. 3, pp. 493-496, 2020.
[128] Z. Wang *et al.*, "Capacitive neural network with neuro-transistors," *Nature communications,* vol. 9, no. 1, pp. 1-10, 2018.
[129] Q. Zheng *et al.*, "Artificial Neural Network Based on Doped HfO 2 Ferroelectric Capacitors With Multilevel Characteristics," *IEEE Electron Device Letters,* vol. 40, no. 8, pp. 1309-1312, 2019.
[130] S. D. Pyle, R. Zand, S. Sheikhfaal, and R. F. Demara, "Subthreshold Spintronic Stochastic Spiking Neural Networks With Probabilistic Hebbian Plasticity and Homeostasis," *IEEE Journal on Exploratory Solid-State Computational Devices Circuits,* vol. 5, no. 1, pp. 43-51, 2019.
[131] A. Sengupta and K. Roy, "Short-term plasticity and long-term potentiation in magnetic tunnel junctions: Towards volatile synapses," *Physical Review Applied,* vol. 5, no. 2, p. 024012, 2016.

[132] T. Chang, S.-H. Jo, and W. Lu, "Short-term memory to long-term memory transition in a nanoscale memristor," *ACS nano*

vol. 5, no. 9, pp. 7669-7676, 2011.

[133] G. Srinivasan, A. Sengupta, and K. Roy, "Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip STDP learning," *Scientific reports,* vol. 6, p. 29545, 2016.

[134] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 288-301: IEEE.

[135] Y. Li *et al.*, "Capacitor-based cross-point array for analog neural network with record symmetry and linearity," in *2018 IEEE Symposium on VLSI Technology*, 2018, pp. 25-26: IEEE.

[136] K. Y. Camsari, S. Salahuddin, and S. Datta, "Implementing p-bits with embedded MTJ," *IEEE Electron Device Letters,* vol. 38, no. 12, pp. 1767-1770, 2017.

[137] (2010). *DRAM Power Model*. Available: https://www.rambus.com/energy/

[138] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE TCAD,* vol. 31, no. 7, pp. 994-1007, 2012.

[139] J. Sankey *et al.*, "Mechanisms limiting the coherence time of spontaneous magnetic oscillations driven by dc spin-polarized currents," *Physical review B,* vol. 72, no. 22, p. 224427, 2005.

[140] D. Fan, S. Maji, K. Yogendra, M. Sharad, and K. Roy, "Injection-locked spin hall-induced coupled-oscillators for energy efficient associative computing," *IEEE Transactions on Nanotechnology,* vol. 14, no. 6, pp. 1083-1093, 2015.

[141] S. Han *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 75-84.

[142] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 DNPU: An 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 240-241: IEEE.

[143] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:.* 2014.

[144] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim, "NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 283-295: IEEE.

[145] S. Angizi, N. A. Fahmi, W. Zhang, and D. Fan, "PANDA: Processing-in-MRAM Accelerated De Bruijn Graph based DNA Assembly," *arXiv preprint arXiv:.06177,* 2020.

[146] S. Angizi, Z. He, and D. Fan, "Energy efficient in-memory computing platform based on 4-terminal spin Hall effect-driven domain wall motion devices," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 2017, pp. 77-82.

[147] S. Angizi, J. Sun, W. Zhang, and D. Fan, "Pim-aligner: A processing-in-mram platform for biological sequence alignment," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1265-1270: IEEE.
[148] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1-6.
[149] A. F. Vincent *et al.*, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *IEEE transactions on biomedical circuits,* vol. 9, no. 2, pp. 166-174, 2015.
[150] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proceedings of the IEEE,* vol. 100, no. 6, pp. 1951-1970, 2012.
[151] S. Angizi *et al.*, "Accelerating Deep Neural Networks in Processing-in-Memory Platforms: Analog or Digital Approach?," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 197-202: IEEE.
[152] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Transactions on Very Large Scale Integration Systems,* vol. 25, no. 10, pp. 2688-2699, 2017.
[153] M. N. Bojnordi and E. Ipek, "Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 1-13: IEEE.
[154] M. Price, J. Glass, and A. P. Chandrakasan, "14.4 A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 244-245: IEEE.
[155] *Predictive Technology Model (PTM)*. Available: http://ptm.asu.edu/
[156] L. Xia *et al.*, "MNSIM: Simulation platform for memristor-based neuromorphic computing system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 37, no. 5, pp. 1009-1022, 2017.
[157] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems,* vol. 31, no. 7, pp. 994-1007, 2012.
[158] *The MNIST database of handwritten digits*. Available: http://yann.lecun.com/exdb/mnist/
[159] *Beyond the Top 1000 Names* Available: https://www.ssa.gov/oact/babynames/limits.html
[160] V. Ostwal, R. Zand, R. DeMara, and J. Appenzeller, "A novel compound synapse using probabilistic spin–orbit-torque switching for MTJ-based deep neural networks," *IEEE Journal on Exploratory Solid-State Computational Devices Circuits,* vol. 5, no. 2, pp. 182-187, 2019.