2023

Leveraging Signal Transfer Characteristics and Parasitics of Spintronic Circuits for Area and Energy-Optimized Hybrid Digital and Analog Arithmetic

Adrian Tatulian
University of Central Florida

LEVERAGING SIGNAL TRANSFER CHARACTERISTICS AND PARASITICS OF SPINTRONIC CIRCUITS FOR AREA AND ENERGY-OPTIMIZED HYBRD DIGITAL AND ANALOG ARITHMETIC

by

ADRIAN TATULIAN
M.S. University of Central Florida, 2020
B.S. University of Central Florida, 2013

A dissertation submitted in partial fulfillment 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

Spring Term
2023

Major Professor: Ronald F. DeMara

ABSTRACT

While Internet of Things (IoT) sensors offer numerous benefits in diverse applications, they are limited by stringent constraints in energy, processing area and memory. These constraints are especially challenging within applications such as Compressive Sensing (CS) and Machine Learning (ML) via Deep Neural Networks (DNNs), which require dot product computations on large data sets. A solution to these challenges has been offered by the development of crossbar array architectures, enabled by recent advances in spintronic devices such as Magnetic Tunnel Junctions (MTJs). Crossbar arrays offer a compact, low-energy and in-memory approach to dot product computation in the analog domain by leveraging intrinsic signal-transfer characteristics of the embedded MTJ devices. The first phase of this dissertation research seeks to build on these benefits by optimizing resource allocation within spintronic crossbar arrays. A hardware approach to non-uniform CS is developed, which dynamically configures sampling rates by deriving necessary control signals using circuit parasitics. Next, an alternate approach to non-uniform CS based on adaptive quantization is developed, which reduces circuit area in addition to energy consumption. Adaptive quantization is then applied to DNNs by developing an architecture allowing for layer-wise quantization based on relative robustness levels. The second phase of this research focuses on extension of the analog computation paradigm by development of an operational amplifier-based arithmetic unit for generalized scalar operations. This approach allows for 95% area reduction in scalar multiplications, compared to the state-of-the-art digital alternative. Moreover, analog computation of enhanced activation functions allows for significant improvement in DNN accuracy, which can be harnessed through triple modular redundancy to yield 81.2% reduction in power at the cost of only 4% accuracy loss, compared to a larger network.

Together these results substantiate promising approaches to several challenges facing the design of future IoT sensors within the targeted applications of CS and ML.

ACKNOWLEDGEMENTS

This work was supported by the NSF through ECCS-1810256.

I would like to express my gratitude to Dr. DeMara for welcoming me into the Computer Architecture Lab and supporting me throughout this journey. His teachings and helpful advice have kept me on the right course in my research, leading me to professional growth and several publications. I would also like to thank my committee members, Dr. Nazanin Rahnavard, Dr. Mingjie Lin, Dr. Rickard Ewetz, and Dr. Enrique del Barco, for taking the time to offer their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION AND MOTIVATION[1]

1.1 Research Motivation

The growing ubiquity of satellite, cellular and WiFi communications networks has led to the emergence of Internet of Things (IoT), a new computing paradigm offering challenges as well as opportunities [1]. The IoT vision consists of a global network of interconnected objects [2], serving a wide range of functions, including smart home, traffic management, vehicle safety and autonomous driving, water quality management, and health monitoring [1, 3]. Communication between IoT devices is commonly established through Wireless Sensor Networks (WSNs); in this scheme, sensor nodes communicate with each other directly, and with a centralized base station, through a multi-hop path [3].

WSN sensor nodes typically consist of a sensing unit, processing unit, transceiver unit and power supply unit [1, 3]. Power supply units commonly consist of batteries which are costly to replace once a sensor is in the field; thus, energy-efficiency is a critical feature in IoT. Area efficiency of sensors is also critical [2] to maintain costs within a feasible range. Compressive Sensing (CS) is one possible solution to these challenges: given an input signal sparse in a certain basis, CS reduces the number of samples taken per frame to attain a reduced set of measurements that enable accurate reconstruction of the original signal [3]. By delegating signal reconstruction to the base station, CS achieves a reduction in energy, storage and data transmission overheads in the IoT sensors.

[1] ©IEEE. Part of this chapter is reprinted, with permission, from [132, 136, 137].

A further challenge in IoT design is the need for adaptability: WSN sensors are often placed in dangerous, unreachable, unpredictable and dynamic environments where either no mathematical model to describe the system behavior is available, or manual re-calibration of the devices is not feasible [4]. Thus, WSNs often require smart sensors which use machine learning to adapt to changing conditions, e.g., a change in the signal's region of interest in the context of CS sampling. It is thus desirable to have multi-functional sensors, simultaneously capable of signal sampling, machine learning, data conversion and data transmission. This challenging combination of requirements and constraints necessitates innovations in hardware design within IoT devices.

1.2 Need for Adaptive Mixed-Signal Computation

While CS and machine intelligence offer logical solutions to fundamental constraints of IoT, their implementation in hardware presents additional challenges. First, both rely heavily on Vector-Matrix Multiplication (VMM), which requires memory for storing matrix elements, as well as power for writing memory and performing computations. In-memory computing approaches, such as crossbar arrays based on Non-Volatile Memory (NVM) devices, have shown promise as a potential solution to this problem. Crossbar arrays offer compact, single-cycle and energy-efficient VMM through an analog approach leveraging signal transfer characteristics of NVM circuits, allowing for parallel execution of multiply and accumulate operations. Moreover, NVM crossbar arrays allow for in-memory computing, thus eliminating significant overheads associated with data storage and transfer. Crossbar arrays can be further optimized through an adaptive approach, i.e., focusing resources on more critical subsets within the input space. Previous CMOS-only based approaches to autonomy and adaptability in FPGAs have concentrated on embedding or encapsulating the FPGA devices with a runtime reconfiguration management system [5-11], or self-aware throughput sustainment based on dynamic operating conditions [12-17].

Vector output components of VMM operations are commonly post-processed via scalar, often non-linear transformations. While this can entail significant overheads within the digital domain, an analog approach can once again reduce costs and complexity by leveraging intrinsic properties of embedded devices to perform the necessary computations. A pure analog approach to both VMM and scalar post-processing is especially beneficial for signal processing applications, where the input signal itself is analog.

After processing, IoT sensors must transmit data to the receiver; energy efficiency is especially critical for data transmission, which can contribute to 80% of power consumption in a sensor node [18]. Thus, adaptability is essential for reducing the size of data sets that must be transferred. One way of achieving this is through an adaptive quantization program, whereby subsets of data are communicated at mixed resolutions depending on relative importance levels.

A final hardware challenge is limited read margin of NVM devices. Read margin is particularly problematic in the case of Multi-Level Cells (MLCs), i.e., devices holding multiple bits. MLCs suffer not only from reduced read margin, but also incur more power and area costs than single-bit devices [19]. A layer-wise adaptive quantization approach can eliminate the need for MLCs by reducing the precision level necessary to represent a given model.

In this dissertation, we present a solution integrating these solutions by demonstrating a hybrid digital-analog, in-memory computing architecture for energy and area-efficient CS and machine learning. A Deep Neural Network (DNN) implementation of machine learning entails multiple crossbar arrays stacked in layers, with additional hardware for scalar computations at the output of each layer. The same hardware architecture can also be used for CS sampling; the approach taken in this research is to individually optimize various aspects of this architecture to reduce overheads and improve performance. The first objective is to reduce area and power costs of VMM

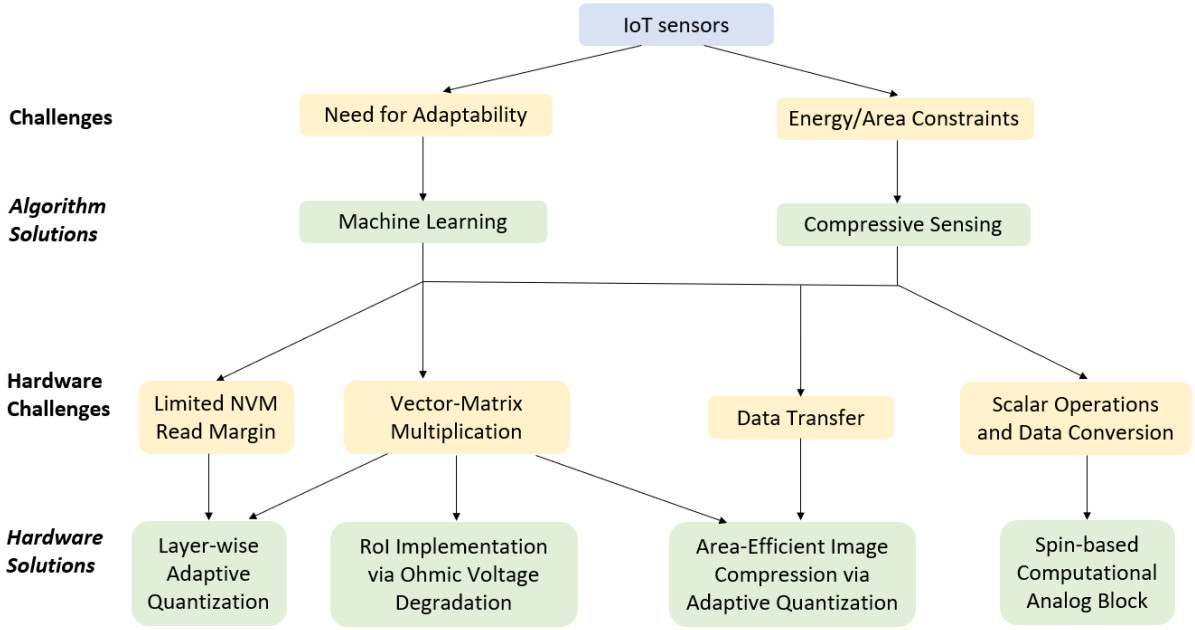


Figure 1.1: Contributions of this dissertation as solutions to challenges in IoT sensor design.

by adapting either density or precision of matrix elements to relative importance levels of corresponding input coefficients. The second objective focuses on reduction of costs associated with scalar computations by performing these computations exclusively in the analog domain. The final objective is cost reduction and performance improvement within DNNs through layer-wise adaptive quantization. These objectives and their context are illustrated in Figure 1.1 and explained in further detail in the following section.

1.3 Contributions of this Dissertation

1.3.1 Region-of-Interest Implementation via Ohmic Voltage Degradation

New data path designs that alleviate the von-Neumann bottleneck remain an intriguing and promising approach to embedded signal processing. For this design, we consider how the recent commercialization of spintronic devices such as Magnetic Tunnel Junctions (MTJs) can offer a

promising approach towards this goal. Specifically, a compact and energy-efficient architecture is sought for in-memory execution of non-uniform CS.

CS is a means of reducing the number of samples taken per frame in the transmission of spectrally-sparse and wideband data by sampling at the information rate rather than the Nyquist rate [20]. As such, CS provides a solution to unprecedented challenges associated with 5G communication, including complexity and power consumption associated with increased bandwidths [21]. In addition, CS is applicable to areas such as data collection, data recovery, distribution networks and channel estimation [22], among other fields. Through a linear transformation, CS limits the number of samples taken per frame to reduce power, storage and data transmission costs. Non-uniform CS is particularly advantageous for signals containing a Region of Interest (RoI), in which a subset of the signal may be more dense in information and thus more critical to reconstruct accurately. Besides having a single RoI, a signal of length n can consist of up to n distinct importance levels. Non-uniform CS reduces reconstruction error by differentially allocating sampling energy in accordance with the relative importance levels of the input signal [23].

Implementing CS sampling in hardware presents unique challenges. One challenge is the need for VMM operations, which can be costly when the sample size is large. Previous work [24, 25] has sought to address these challenges by assigning VMM operations to a Memristive Crossbar Array (MCA). An MCA architecture consists of bit lines running vertically and word lines running horizontally to realize a grid structure, with a memristive device providing a connection between the corresponding bit line and word line in each cell. The memristive device is a variable resistor which changes its resistive state after a certain critical current passes through in either direction.

MCAs are first programmed such that the conductance of each memristive device represents the matrix element at that location. Thus, as voltages representing the input vector components are applied to the rows of the array, currents representing the VMM resultant vector components are read along each column as a result of Kirchhoff's Current Law. The output currents are then converted to voltage levels by Trans-Impedance Amplifiers (TIAs). The MCA allows for single-cycle VMM using a dense and area-efficient architecture.

Recently, researchers have investigated methods for generating randomness without the use of Linear Feedback Shift Registers (LFSRs), e.g., through the use of probabilistic bit (p-bit) devices to write tunable random values into each column based on an analog input to the device. The ability to generate tunable randomness allows for non-uniform CS, whereby the signal sampling rate, determined by the frequency of non-zero elements in each column of the measurement matrix, is adapted to the relative importance levels present in the signal. In this approach, sampling rates are set via power gating [20], which not only reduces power but can also mitigate aging within CMOS transistors [26].

Conventionally, a dedicated hardware unit [27] would be utilized for: a) storing the mapping between signal importance levels and the corresponding configuration flow, and b) converting the output data to analog for subsequent voltage-to-frequency conversion by the p-bit devices. We reduce these hardware overheads by acquiring the necessary circuit-switched configuration settings directly from the MCA word lines.

1.3.2 Area-Efficient Image Compression via Adaptive Quantization

Image compression techniques such as Discrete Cosine Transform (DCT) and CS are feasible solutions to area and energy challenges of IoT: DCT compresses data through a change of basis to the frequency domain; given sparse and wideband data, CS reduces the number of samples taken

per frame via a linear transformation. DCT and CS can be used in conjunction to reduce data storage and transmission overheads within IoT systems; hardware solutions to such techniques have recently been a subject of active research [20, 28].

The wide disparity in significance levels between coefficients in the frequency domain allows for optimized hardware approaches concentrating resources to more important coefficients. For example, Imran *et al.* [29] proposed a dynamic mapping approach, assigning more significant coefficients to healthier processing elements within an FPGA. Moreover, Salehi *et al.* [20] demonstrated improvements using a crossbar design approach whereby more important coefficients are sampled at a higher rate. Finally, Adaptive Quantization (AQ) techniques have shown benefits for both DCT [30] and CS [31]. While implementation of AQ in hardware is intriguing considering the area constraints of IoT devices, conventional crossbar architectures are not well-suited for such an approach.

Herein, a novel Magnetoresistive Random Access Memory (MRAM)-based crossbar design [20] is presented to implement AQ in hardware by allowing for mixed-precision representation of matrix elements. Furthermore, we extend the AQ approach to CS sampling, by dynamically varying quantization levels across matrix elements to sample more important signal coefficients at a higher precision. Hardware implementations of CS as well as DCT are given on the proposed hardware.

1.3.3 Spin-Based Computational Analog Block

Multiplication and exponentiation operations are critical for a variety of applications, including computer vision [32], signal processing [27, 33] and machine learning [34, 35]. Square and square root, for example, are commonly used for normalizing vectors in signal processing applications, and square root may serve as an activation function for neural networks [34]. Despite their ubiquity, a traditional digital implementation of such functions can incur significant area and delay

overheads in the digital domain, requiring 12 or more clock cycles to execute [36] and hundreds of logic gates [37]. As a result, there has recently been renewed interest in pursuing an analog approach to operations such as multiplication, square and square root [38, 39].

Analog circuits trade off computational accuracy for reduction in overheads such as power and area; this is an attractive tradeoff for error-tolerant applications where power and area are constrained, e.g., IoT devices. The benefits offered by analog computation are magnified when used with vector-valued data, since the output data can be transferred to a crossbar array for further processing without the need for digital-to-analog conversion [27]. One example of an ideal use case is CS. CS entails compression and transmission of a spectrally-sparse signal, and then reconstruction of the signal at the receiving end. Machine learning via neural networks is another relevant application.

In recent years, Field Programmable Analog Arrays (FPAAs) have been proposed as a counterpart to traditional digital-only FPGAs, particularly for computations involving sensor interfacing and signal processing [40]. FPAAs consist of a set of analog components, such as Field-Effect Transistors (FETs), capacitors, resistors and diodes integrated into a reconfigurable fabric architecture. While a lack of software for FPAA programmability has been a challenge, recent developments including the RASP and associated high-level tools have provided a pathway for system-level analog design [41].

Analog computation can provide vast energy improvements, up to a 1000-fold improvement in computational energy efficiency [42] and thus FPAA technology has already been implemented in ultra-low power IoT sensing applications, including temperature sensors and heart rate alarms [43].

Herein, we present an analog design for performing generalized n^{th} root and power operations.

The use of Taylor series allows for computation of generalized functions as well. Area overhead is minimized by a) performing computations in the analog domain based on intrinsic properties of the embedded op-amps, and b) a reconfigurable architecture allowing for the realization of multiple functionalities within a single fabric. Our design is ideal for area and energy-limited applications and allows for computations which may not be efficient in the digital domain for such applications, such as computation of DNN activation functions for machine learning. In addition to common analog components, our fabric embeds state-of-the-art MTJ devices for added area benefits and intrinsic stochasticity.

1.3.4 Layer-wise Adaptive Quantization

A crossbar-based hardware architecture can be readily applied to Artificial Intelligence (AI) through DNNs. DNNs consist of multiple crossbar layers which perform a series of linear transformations on the input data, with a scalar activation function applied at the output of each layer. A simple use case of DNNs is classification of handwritten digits, while more complex tasks can include generalized image recognition. These use cases are applicable for IoT sensors, for example, in determination of RoI. Previous approaches to scaling up AI processing include Marker-Passing wherein the processing activity migrates to where the data resides rather than vice-versa [44] as realized on the SNAP-1 Parallel AI Prototype [45].

DNNs are first trained on a set of sample inputs to determine a model for further recognition tasks. The model consists of matrix elements, i.e., weights, assigned to each layer. Weights are commonly stored using a multi-bit representation implemented through MLC devices. Due to the increased costs and reduced reliability of MLCs [19, 46, 47], this architecture can be optimized through a layer-wise adaptive quantization program, whereby a genetic algorithm is used to optimize the bit configuration such that high precision is only assigned to layers which require it.

Moreover, the high-precision layers are implemented using a combination of single-bit MRAM devices to eliminate the need for MLCs.

Herein, we investigate the efficacy of such an approach in terms of improvements in area. Layer-wise adaptive quantization is explored as a pathway for decreasing overheads within machine learning architectures without degrading accuracy.

CHAPTER 2: BACKGROUND AND RELATED WORK[2]

2.1 Spin-Based Devices

2.1.1 Magnetic Tunnel Junction (MTJ) Fundamentals

Magnetoresistive Random Access Memory (MRAM) based on Magnetic Tunnel Junctions (MTJs) has recently been researched as a class of Non-Volatile Memory (NVM) device delivering numerous advantages, including near-zero standby power dissipation [27], high endurance [48] and vertical integration capabilities resulting in high density [49]. MTJs are composed of two ferromagnetic layers: a fixed layer and free layer, separated by a thin oxide barrier, as Figure 2.1. shows. The two stable states of the MTJ, the Parallel (P) state and Anti-Parallel (AP) state, are determined by the relative orientation of the free-layer magnetization with respect to the fixed layer. Device resistance is significantly higher in the AP state.

While various materials may be chosen for MTJ fabrication, one common choice is the use of Fe for ferromagnetic layers and MgO for the oxide barrier. This structure may be achieved using

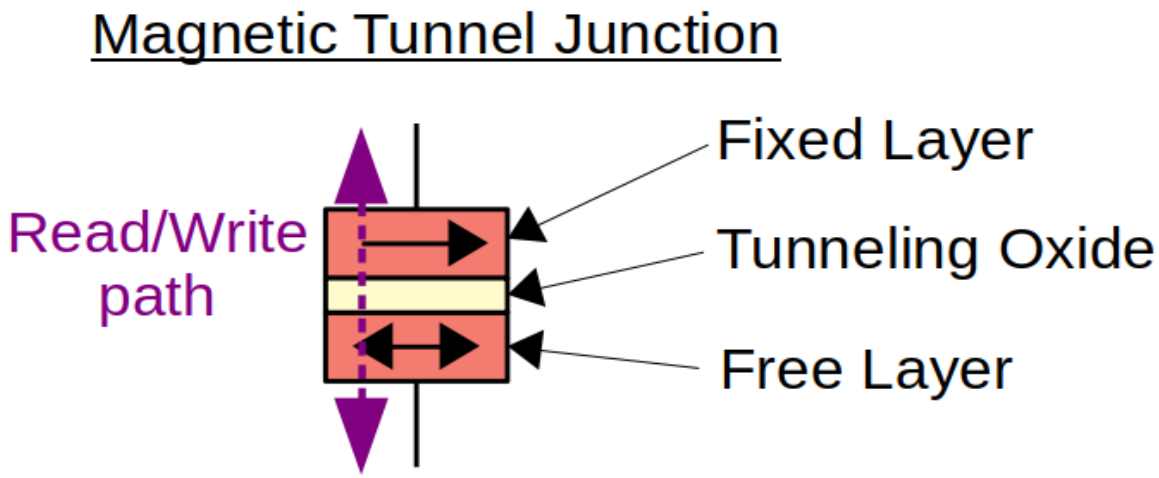


Figure 2.1: Simplified structure of two-terminal MTJ operating under STT switching.

[2] ©IEEE. Part of this chapter is reprinted, with permission, from [27, 132, 136, 137].

existing fabrication methods, e.g., molecular beam epitaxy for preparation of the Fe layer, followed by growth of the MgO layer using e-beam evaporation and patterning using photolithography [50]. The fabricated device is then placed on chip at the fourth metal line, in a CMOS backend process [51]. MTJs are commonly vertically integrated with CMOS technology using through-silicon vias in a 3D architecture, thus maximizing area efficiency and simultaneously minimizing data transfer overheads [52, 53]. As the building block of MRAM technology, MTJs have been proposed as an alternative to SRAM in cache memory [54]. Further applications benefiting from a hybrid CMOS/MRAM approach include full adders [51] and analog-to-digital converters [55].

MTJ resistance is commonly modeled [56-58] using the following equations:

$$R_p = \frac{t}{Factor \times Area \times \sqrt{\varphi}} \exp\left(1.025 t \sqrt{\varphi}\right) \tag{2.1}$$

$$R_{ap} = R_p(1 + TMR) \tag{2.2}$$

$$TMR = \frac{TMR_0}{1+(\frac{V}{V_h})^2} \tag{2.3}$$

where R_p is P-state resistance, R_{ap} is AP-state resistance, t is the oxide thickness, φ is the oxide potential, *Factor* is a material-dependent constant, *Area* is the device surface area, *TMR* is the tunnel magnetoresistance ratio, V is bias voltage and V_h is the empirically-determined bias voltage at which the TMR is half of its initial value. This model is an approximation which neither considers voltage dependence of P-state resistance nor temperature dependence of resistance.

2.1.2 MTJ Switching Characteristics

MTJ switching can occur through the Spin Transfer Torque (STT) mechanism, whereby a spin-polarized current passing through the device reverses the magnetization orientation of the free layer [59]. The MTJ is a two-terminal device when configured for STT switching, as Figure 2.1

shows. The free-layer magnetization in an MTJ is governed by the Landau-Lifshitz Gilbert (LLG) equation [58]:

$$\frac{d\boldsymbol{m}}{dt} = -\gamma \boldsymbol{m} \times \boldsymbol{H_{eff}} + \alpha \boldsymbol{m} \times \frac{d\boldsymbol{m}}{dt} - \rho_{stt} \boldsymbol{m} \times (\boldsymbol{m} \times \boldsymbol{m}_r) \tag{2.4}$$

where $\boldsymbol{H_{eff}}$ is the effective magnetic field, γ is gyromagnetic ratio, α is a damping coefficient, $\boldsymbol{m}$ and $\boldsymbol{m}_\mathbf{r}$ are magnetizations of the free and fixed layers respectively, and ρ_{stt} is the spin transfer torque coefficient and is directly proportional to the current passing through the device. MTJs are commonly composed of uniaxial ferromagnets, i.e., there is only one easy axis, and thus two directions, along which the magnetization is stable, though an energy barrier must be overcome in order to switch between the two states. The time it takes the free layer to switch between these two stable states is determined by solving the LLG equation. In spintronic circuit design, switching is typically driven by the last term in Eq. 2.4, which is directly proportional to the spin-polarized current density passing through the device.

2.1.3 MTJ I-V Characteristics

In 1963, Simmons [60] examined the problem of electron tunneling between two metals separated by an insulating film. Simmons considered linear variation of potential through the film and conducted his analysis for different ranges of bias voltage. Brinkman's 1970 publication [61] extended this work by also accounting for electronic band structure. Brinkman determined the rate of electrons tunneling through the film by integrating the density of states in the metals comprising the two electrodes, the Fermi distribution function and the tunneling probability. He thereby determined the tunneling current density, and then the conductance through the film, which he expanded to second order in voltage to get:

$$G(V) = G_0[1 - \left(\frac{A_0\Delta\varphi}{16\varphi^{\frac{3}{2}}}\right)eV + (\frac{9}{128}\frac{A_0^2}{\varphi})(eV)^2] \tag{2.5}$$

$$G_0 = \left(3.16 \times 10^{10}\frac{\sqrt{\varphi}}{t}\right)\exp\left(-1.025t\sqrt{\varphi}\right) \tag{2.6}$$

$$A_0 = \frac{4\sqrt{2m}t}{3\hbar} \tag{2.7}$$

where G is conductance, V is bias voltage, $\Delta\varphi$ is the potential barrier through the film, φ is average potential in the film, e is electron charge, t is film thickness, m is the electron mass and $\hbar$ is the reduced Planck constant. According to Brinkman's analysis, Eq. 2.5 is accurate to within 10% when the film thickness is at least 1 nm and $\frac{\Delta\varphi}{\varphi}$ is less than 1.

While Brinkman's model treated film conductance as a function of bias voltage, he did not consider any magnetic properties of the metals sandwiching the film and hence did not give any analysis of conductance as a function of relative magnetization orientation. The first MTJ was considered in 1975 by Julliere [62] who considered a junction composed of Fe-Ge-Co. Julliere determined that the conductance variation through the germanium layer separating the two ferromagnets was given by the relation:

$$\frac{G_p - G_{ap}}{G_p} = \frac{2PP\prime}{1+PP\prime} \tag{2.8}$$

where G is conductance and P and P' are spin polarization factors for the two ferromagnetic layers given by $P = 2a - 1$ and $P' = 2a' - 1$, where a and a' are the fractions of tunneling electrons entering iron and cobalt, respectively, whose magnetic moments are parallel to the magnetization.

Further analysis by Slonczewski in 1989 yielded the following equation for conductance [63]:

$$G(\theta) = \bar{G}(1 + PP'cos\theta). \tag{2.9}$$

where $\bar{G} = \frac{G_p + G_{ap}}{2}$ and θ is the angle between magnetization vectors in the two ferromagnetic layers.

Slonczewski's 1996 publication [64] went further to propose the STT switching mechanism for MTJs, wherein he predicted a critical current that must be reached in order for the device to switch from the P state to the AP state and vice-versa. This current, I_c, can be expressed as [65]:

$$I_c = \frac{2\alpha\gamma eE}{\mu_B g} \tag{2.10}$$

where α is the same damping constant that appears in the LLG equation, γ is the gyromagnetic ratio, e is the electron charge, E is the energy barrier between P and AP states, μ_B is the Bohr magneton, and g is the spin polarization efficiency factor. In general, the device will switch when a positive current equal to I_{c+} passes through it; when the current is removed, the device will retain its state and only switch back when a negative current I_{c-} passes through it. Thus, the state of the device follows a hysteresis curve. The presence of the hysteresis curve gives the device its memory properties.

While I_{c+} and I_{c-} are both given by Eq. 2.10, their value may be different due to the state-dependence of the g parameter in that equation. Furthermore, both critical current values may be lowered at the cost of higher switching delay, giving a tradeoff between power and performance.

Since Slonczewski's 1996 publication, many MTJ resistance models, both physical and empirical, have been explored. One common approach [56-58] is to express the P state resistance as a zero-order Brinkman model, i.e., using just the first term of Eq. 2.5 and then defining the tunnel magnetoresistance ratio as $TMR = \frac{R_{ap} - R_p}{R_p}$ so that the AP state resistance may be expressed

as: $R_{ap} = R_p(1 + TMR)$. This is the same model which was presented earlier through Eq. 2.1 – 2.3; since the only parameter in this model is the TMR, it may be referred to as the TMR model.

2.1.4 MTJ Temperature Dependence

Experimental evidence [66] has shown that MTJ conductance varies with temperature. Thus, temperature dependence is a characteristic shared between electronics and spintronics. A temperature-dependent model for MTJs, proposed by Shang [67], is:

$$G(T, \theta) = G_T(T)\{1 + P(T)^2 cos\theta\} + G_{SI}(T) \tag{2.11}$$

$$G_T(T) = \frac{G_0 CT}{\sin(CT)} \tag{2.12}$$

$$P(T) = P_0(1 - \alpha T^{\frac{3}{2}}) \tag{2.13}$$

where G is conductance, T is absolute temperature, P is spin polarization, and G_0, C, P_0 and α are fitting parameters. The theoretical basis for Eq. 2.11 is that the $1 + P^2\cos\theta$ factor comes from Slonczewski's model (Eq. 2.9), the G_T factor accounts for thermal broadening of Fermi distributions and G_{SI} is polarization-independent conductance. Polarization is assumed to follow the same temperature dependence as magnetization, which follows a $T^{3/2}$dependence to account for thermal excitation of spin waves. Due to excellent agreement of Shang's model with experimental data, the model has seen widespread use [66, 68] since its original publication.

2.1.5 Spin Hall Effect-based MTJs (SHE-MTJs)

A key challenge facing MTJs is the high energy cost of STT switching. As a solution to this challenge, researchers have investigated the Spin Hall Effect (SHE) as an alternative switching mechanism, which brings dual benefits of lower switching energy and separate read and write paths. The SHE-MTJ is a three-terminal device fabricated by connecting the free layer to a heavy

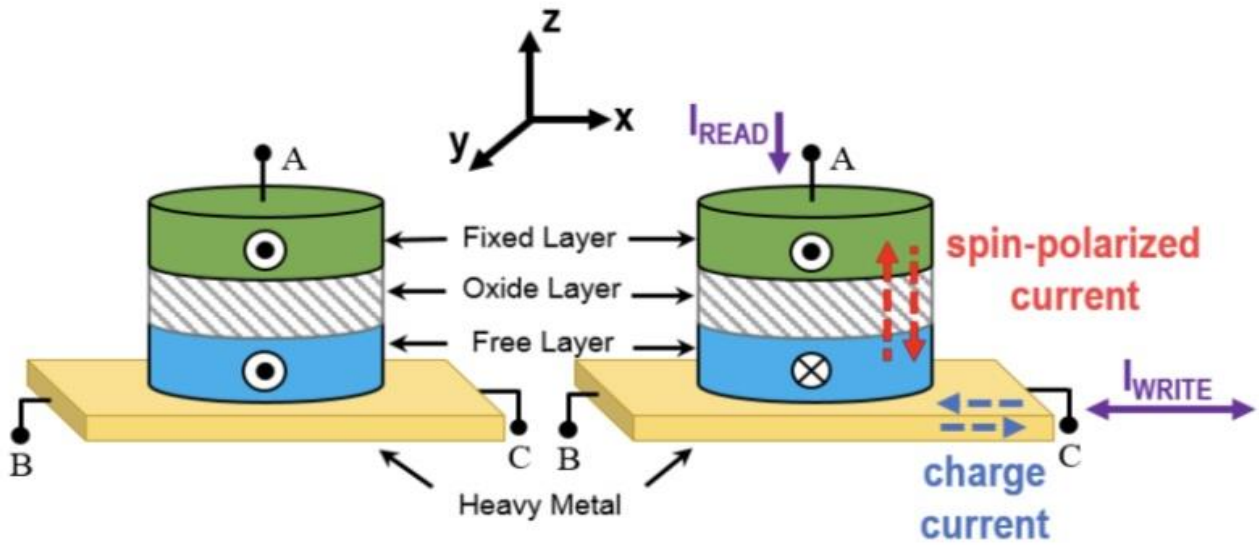


Figure 2.2: SHE-MRAM device in the P state (left) and AP state (right). The device switches states based on charge current passing through the heavy metal strip.

metal strip [69], as shown in Figure 2.2. Common materials for the heavy metal strip include β–Tantalum, β–Tungsten, and Platinum. Through the Spin Hall Effect, a bi-directional charge current passing through the heavy metal strip generates a spin-polarized current through the device; if magnitude and time duration are sufficient, this spin-polarized current then reverses the magnetization orientation of the free layer [27, 55]. Interestingly, the induced spin current can be larger in magnitude than the inducing charge current.

2.1.6 Probabilistic Spin Logic using Low-Barrier MTJs

In addition to device resistance, the energy barrier, E_B, between the P and AP states of an MTJ device can be tuned based on fabrication dimensions. The device is considered to be low-barrier under the condition $E_B \ll 40kT$, in which case thermal fluctuations at room temperature are sufficient to change the state of the device. This observation has led to construction of the probabilistic bit (p-bit) device, as shown in Figure 2.3. A p-bit [70, 71] takes analog input and yields a digital output whose probability of being logic 1 depends on the supplied input voltage.

This functionality is due to the p-bit's structure as a voltage divider between a low-barrier MTJ and NMOS transistor. A higher voltage applied to the gate of the transistor results in reduced drain-source resistance, r_{ds}, which increases the probability of delivering sufficient voltage to the input of the inverter to yield a logic 1 output.

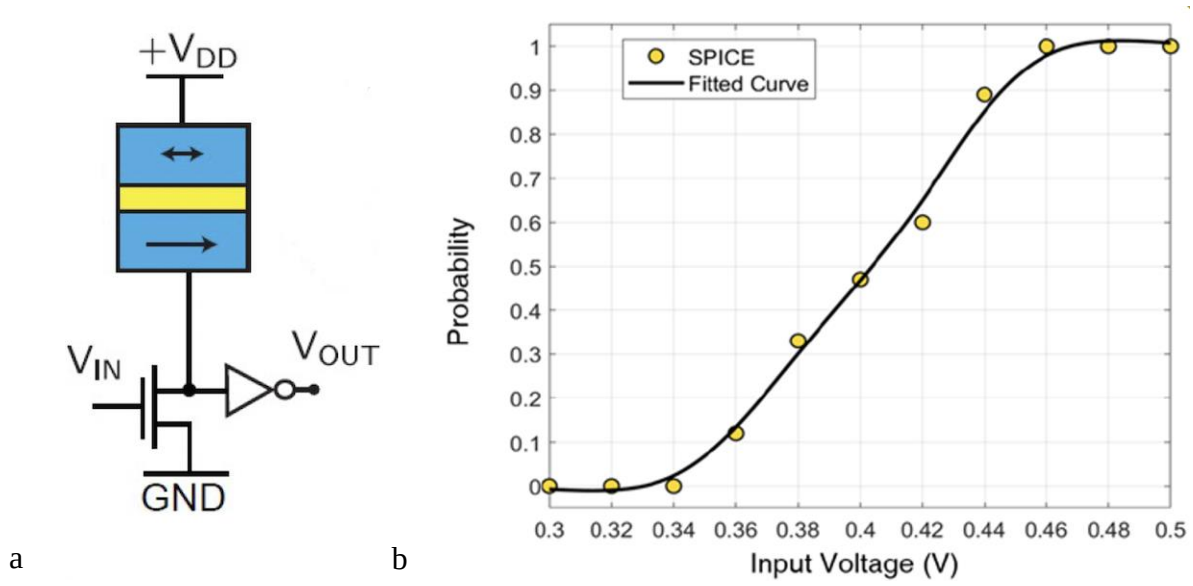
The p-bit output is described by:

$$V_{out} = V_{DD}\,\mathrm{sgn}[\tanh\left(\frac{V_b}{V_0}\right) + \mathrm{rand}(-1,1)] \tag{2.14}$$

where sgn represents the sign function, $\mathrm{rand}(-1,1)$ represents a random number in [-1,1], V_b is bias voltage and V_0 is a model parameter [70]. Thus, the probability of obtaining a logic 1 output from the p-bit device is given by:

$$\mathrm{P}(1) = \frac{1}{2}\left(1 + \tanh\left(\frac{V_b}{V_0}\right)\right). \tag{2.15}$$

Averaging p-bit outputs yields the hyperbolic tangent function through Eq. 2.15.



a b

Figure 2.3: Structure of a p-bit device consisting of a voltage divider between a low-barrier MTJ device and NMOS transistor (a); probability of a logic 1 output value (b).

2.2 Memristive Crossbar Arrays (MCAs)

2.2.1 MCA Fundamentals

Memristive Crossbar Arrays (MCAs) are a class of in-memory computing architecture consisting of word lines and bit lines, forming columns and rows, respectively, to realize a matrix structure. The word lines and bit lines are interconnected via programmable and nonvolatile memristive devices, such as MRAM or Resistive Random Access Memory (RRAM) [28]. Given a set of input voltages, v_j, applied along the MCA word lines, the current on the k^{th} bit line, i_k, is given as $i_k = \sum_j G_{jk} v_j$, where G_{jk} represents the conductance of the memristive device at the intersection of the j^{th} word line and k^{th} bit line; this relationship is a direct result of Kirchhoff's Current Law. As a result of this relationship, MCAs are intrinsically well-suited for VMM computations by setting conductance values to represent matrix elements and providing vector inputs as voltages to the array. Since negative conductance values are not attainable, VMM is commonly computed using a dual crossbar design to incorporate negative matrix elements. MCAs may also be configured to solve systems of linear equations and the eigenvalue problem [72].

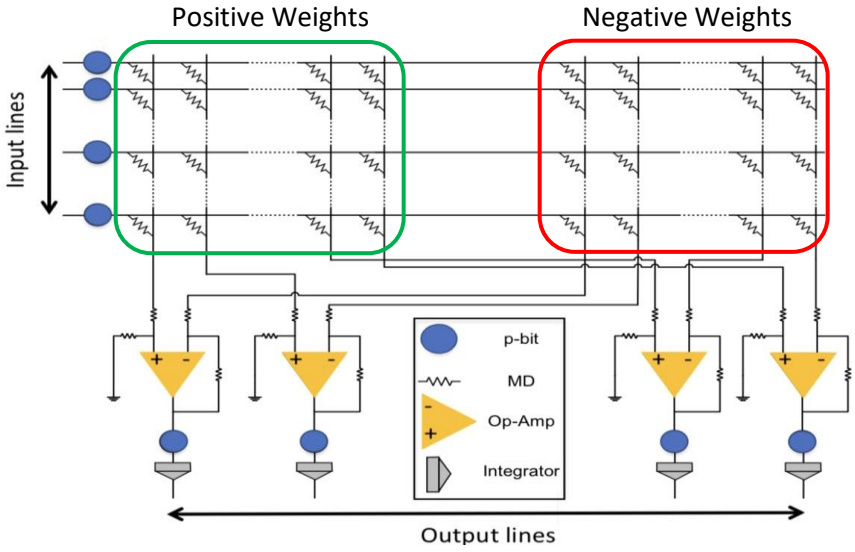


Figure 2.4: Spin-based MCA, consisting of 1) p-bit, 2) Memristive Device (MD) implemented via MRAM or RRAM, 3) op-amp and 4) integrator.

MCAs deliver a variety of advantages in VMM computation, including single-cycle VMM if the input data set is within the size limits of the array [28, 73]. Moreover, MCAs realize significant reductions in area and energy by performing computations in analog based on intrinsic signal transfer characteristics of embedded memristive devices. It was recently seen [27] that MCAs can realize up to a 5× reduction in energy and a 26× device reduction compared to a CMOS equivalent using SRAM cells together with multiply and accumulate units. Computation in the analog domain is especially advantageous for signal sensing applications, where inputs are given in analog.

As shown in Figure 2.4, MCAs are typically designed with transimpedance amplifiers at each word line output for converting currents to voltages that can interface with other units. Moreover, spin-based MCAs can include p-bit devices built into bit lines and word lines for stochastic computing applications.

2.2.2 Sneak Currents and Parasitic Voltage Degradation

One challenge to MCA functionality is the presence of sneak currents [74]. Sneak currents arise due to the difficulty of targeting a single cell in the array: in order to pass a current through a designated memristive device, either for reading or writing, the corresponding word line must be brought to V_{DD} while the bit line must be held at ground, or vice-versa. However, as seen in Figure 2.5, this configuration also introduces currents through non-targeted devices. To counteract this, all word lines and bit lines other than the ones belonging to the target device are held at $V_{DD}/2$. While this reduces the voltage difference across non-target devices from V_{DD} to $V_{DD}/2$, a certain amount of current still flows through the half-selected devices, i.e., devices sharing either a bit line or word line with the targeted device These currents are referred to as sneak currents and can result in significant energy overheads and performance degradation of the architecture.

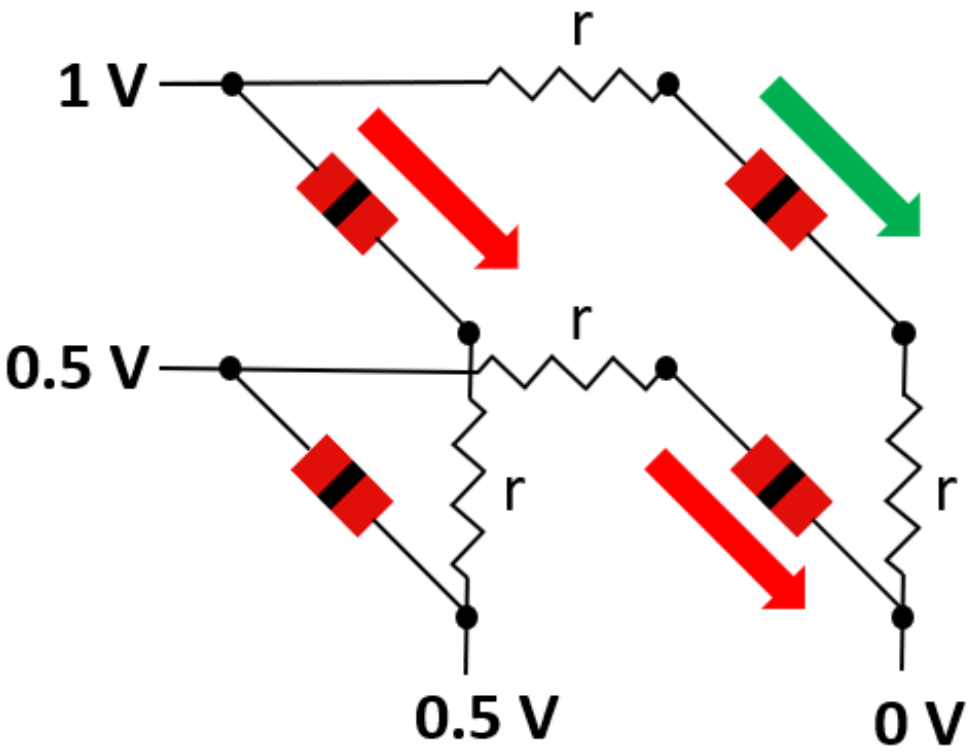


Figure 2.5: Sneak currents, represented by red arrows, introduced in writing to the top-right device in a 2×2 MCA. Also shown is the parasitic line resistance.

One common method of mitigating sneak currents is through the use of bi-directional selector diodes [75, 76]. These diodes are connected in series with the memristive device in each cell and can block current from flowing through half-selected devices given a voltage bias within a certain threshold. Another strategy is introducing a CMOS transistor in series with each device, which allows control over which cells are active at a given time. These architectures are commonly referred to as 1S1R and 1T1R, respectively [77].

A second challenge pertaining to crossbar arrays is the presence of parasitic line resistance, which results in reduced voltage drop across memristive devices due to *IR* loss within word lines and bit lines comprising the array. The *IR* loss is amplified by sneak currents and can be considerable in the absence of mitigation strategies.

Voltage degradation arising from parasitic *IR* loss leads to reduced performance in devices located far from the voltage sources in the array. During the write operation, this reduced voltage drop can significantly increase the time taken for a device to switch, e.g., a 0.4 V reduction in voltage drop across a device can increase switching delay by a factor of 10 [74]. The *IR* drop effect can also lead to read and write failures. While write failure may be avoided by using increased voltages, such an approach is not ideal since it entails increased power consumption and increases the probability of write disturbance.

A variety of approaches have been proposed to mitigate the challenges described above. Xu [74] proposed splitting writes into the crossbar into two cycles, i.e., performing half of the writes in one cycle and half in the next in order to limit the amount of current flow. Shevgoor [78] proposed using a sample and hold circuit to store charges generated by the sneak currents on a capacitor, and thus cancel out future sneak currents during the next write or read operation. Zhang [79] proposed mapping hot data, i.e., frequently accessed data, to fast regions in the MCA while mapping cold data into slower regions, having greater voltage degradation and thus greater performance loss. Zhang [80] further proposed splitting bit lines using a separation transistor and using this approach to differentially map data based on access frequency.

2.2.3 Multi-Bit Crossbar Arrays

Multi-bit representation of matrix elements is often necessary for precise computations. Previous works include a variety of methods for achieving this in hardware. One approach is through the use of Multi-Level Cell (MLC) devices implemented using MRAM [47, 81] or RRAM [82]. Authors in [47] discuss challenges facing MLC devices, i.e., frequent read errors, incomplete writes, and write disturbance as a result of thermal fluctuations and process variation. This is particularly true for technologies such as MRAM that are especially limited in read margin.

Despite their limited reliability, authors in [81] discuss the feasibility of incorporating MLCs for approximate computing applications.

In addition, authors in [83] discuss the use of SOT (Spin Orbit Torque)-MRAM devices for construction of multi-bit arrays. Each SOT-MRAM device consists of a fixed number of magnetic domains, which switch based on magnitude of current passing through the device. The device exhibits intrinsic stochasticity, i.e., the number of magnetic domains that switch for a given current is given by a sigmoidal probability distribution. However, SOT-MRAM devices also exhibit significant area overhead as a result of holding values through a thermometer code and require significant energy overheads for write operations.

Multi-bit quantization can alternatively be achieved through a combination of single-bit devices, either spread across columns of a single crossbar [84, 85], or across multiple crossbar arrays [86]. In both cases, intermediate results are combined using shift-and-add operations to yield the final vector product. Zou *et al.* [86] discuss the idea of representing an *n*-bit weight, *w*, via 2*n* crossbar arrays, containing the values g_0^+, g_0^-, ..., g_n^+, g_n^-. During a read operation, the weight is computed using the formula: $(g_0^+ \times 2^0 + \cdots + g_n^+ \times 2^n) - (g_0^- \times 2^0 + \cdots + g_n^- \times 2^n)$. In addition to simplicity, this method provides added security since an adversary is not able to infer information from the memory, given access to only one crossbar array. Matrix splitting is commonly necessary for physical mapping of matrices due to the size of the matrix exceeding currently manufacturable crossbar arrays, as well as non-linear voltage degradation effects affecting computational accuracy [87, 88].

To reduce area in a multi-crossbar memory, Chen *et al.* [89] show that individual crossbars can be stacked in a 3D architecture; by reversing the deposition order between layers, the authors reduce hardware overheads by sharing electrodes and interconnection wires. Furthermore, for

further memory optimization, Khan *et al.* [90] demonstrate a technique for reading multiple bits simultaneously in a single MRAM array by modifying the bias voltage applied to unselected cells to maintain a feasible sensing margin. Furthermore, the authors achieve multi-bit write operations by applying a pulsed bias voltage to unselected cells to prevent write disturbance.

Multi-bit arrays allow for adaptive quantization techniques for optimal allocation of resources. Proposed techniques have included dynamic reconfiguration of bit-widths in Analog to Digital Converters (ADCs) at the outputs of the array [28]; this approach reduces computational power in signal conversion, as well as reducing memory and bandwidth requirements for data storage and transmission. Furthermore, Khan *et al.* [91] gave an MCA-based approach for dynamic layer-wise weight quantization in neural networks by using multiple memristive devices spread across MCA columns to hold a single weight; the authors used a power gating technique to switch off certain columns for low-precision computations. Other approaches to MCA-based adaptive quantization [92, 93] have targeted the weight mapping distribution for a fixed precision level, rather than varying levels of precision.

2.3 Mixed-Signal Computing

2.3.1 Analog Computing: Motivation and Related Works

Analog computation relies on intrinsic signal transfer characteristics of circuit components to conduct computations. Analog computations tend to be approximate, but may be superior to digital counterparts in latency, power consumption, and area albeit at times subject to significant precision, noise, temperature, and operating voltage challenges.

Implementation of analog computation has taken a wide variety of forms within AI applications. A recent work [94] discusses analog computation in the context of one type of neural network, i.e., the multilayer perceptron (MLP). The MLP hardware utilizes MOSFET-based

current mirrors along with operational transconductance amplifiers to perform VMM. Two operations are performed: multiplication and addition utilizing Kirchhoff's Current Law in order to sum the current signals at a specific node. Authors in [95] propose a generalized non-linear function synthesizer through Taylor series approximation. The hardware implementation relies on successive application of a squaring unit (SU) based on a class AB current mirror architecture and yields a maximum error of 10% for a 5^{th} order polynomial. Meanwhile, authors in [96] show how to construct a reliable nonlinear circuit to exploit nonideal properties within a cascaded array of analog multipliers for simulation of mathematical chaos. In [97], a mixed analog and digital hybrid solution is introduced that seeks to alleviate challenges, e.g., lack of programmability associated with a fully analog nonlinear computation stack. The architecture developed by the authors is applied to obtain solutions to non-linear ordinary differential equations. In [98], root and power computations are implemented using time-mode circuits. Its hardware relies on the translinear principle, i.e., exponential I-V characteristics of CMOS transistors, which allow computations by cascading exponential and logarithmic units.

Several authors have sought automated hardware synthesis and optimization through the use of genetic algorithms. In [99], which is one of the pioneering works in this field, the authors present a wide variety of analog circuits produced via genetic algorithms, including a cube root implementation. Subsequently, [100] expands upon the work of [99] to synthesize a wider variety of evolutionary computation circuits, with improved output error. Meanwhile, [101] conducts iterative refinement on computational circuits including squaring, square root, and cubing circuits. From this iterative process, certain circuits that are created through genetic programming are able to be refined through the error produced from the previous best-of-run from the same circuit. Authors in [102] extend this approach by combining analog and digital computation, whereby out-

Table 2.1: Summary of analog computation architectures, including hardware overhead.

Work	Functionality	Mode of operation	No. of elements	Highlighted Contributions
[95]	n^{th} power via Squaring Unit	Class AB current mirror	22	Arbitrary nonlinear functions in terms of Taylor series expansion
[98]	Square, cube, 4^{th} power	Translinear time-to-voltage and voltage-to-time convertors	~100	Nonlinear operations through the time-mode translinear principle
[99]	Cube root	Evolved computational circuit	48	Pioneer in evolutionary circuit design
[100]	Square, square root, cube, cube root	Evolved computational circuit	≤44	Genetic algorithms for optimizing analog circuits for non-conventional applications

puts are refined digitally to improve computational accuracy for less error-prone applications. Finally, [103] explores synthesis of arbitrary functions through Puiseux series, using genetic algorithms to minimize error. Table 2.1 summarizes a selection of these works, highlighting the hardware overheads in comparison to an approximate digital multiplier requiring ~1000 components [37].

2.3.2 Mixed-Signal Field Programmable Array (MFPA)

Due to the significant benefits offered by analog computation to certain use cases, there has been a renewed interest in extending the scope of reconfigurable computing to the analog domain. The Reconfigurable Analog Signal Processor (RASP) [41] introduced in 2012 was an attempt to overcome two of the main challenges preventing widespread adaptation of analog processing: lack of a programmable interface and lack of robust design tools. The RASP provided a set of high-level design tools for system-level analog design.

Concurrently with the introduction of the RASP, researchers [40] developed a Field Programmable Mixed-Signal Array (FPMA) consisting of both analog and digital elements arranged in a Manhattan-style fabric. Their design consisted of Computational Logic Blocks comprised of LUTs and D Flip-flops, and Computational Analog Blocks consisting of analog elements such as op-amps, capacitors, and transistors. Their design used a global interconnect to integrate transistors (FETs), capacitors, resistors and diodes into a reconfigurable fabric architecture. Further innovations have included introduction of a 16-bit microprocessor into the FPMA fabric [104], and Time-Domain Configurable Blocks for dynamic reconfigurability of analog functions [105].

The Mixed-Signal Field Programmable Array (MFPA) [27] advances a device-level-to-architecture-level approach to integrate front-end signal processing within a low-footprint reconfigurable fabric that enables mixed-signal computing for high-throughput on-chip CS. Mixed-signal techniques combined with in-memory computation geared to the demands of CS are integrated in a field-programmable and run-time adaptable platform.

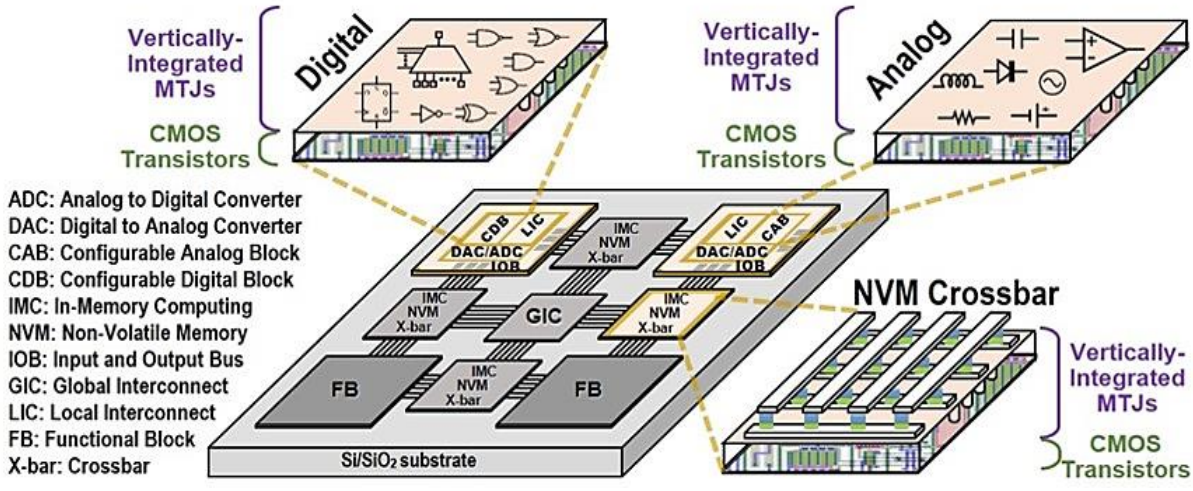


Figure 2.6: Hybrid spin/charge device realization as configurable blocks within the MFPA fabric.

As shown in Figure 2.6, the MFPA architecture entails a circuit and register-level design so that an MFPA slice acquires analog signals and then performs CS sampling and reconstruction via in-memory computing using reduced precision/dynamic range. In-memory computing approaches extend related works, such as Rabah's architecture [106] consisting of separate processing elements (PEs) and memory elements (MEs). The architecture develops analog computable memories, or analog computing arrays, where instead of storing the analog values to be used by external computing elements, in-memory computing is utilized. This cross-cutting beyond-von-Neumann architecture explores the use of dense MRAM-based crossbar arrays to perform VMM necessary for execution of CS sampling and signal reconstruction algorithms.

MTJs having low energy barrier are used as compact True Random Number Generators (TRNGs) for generation of the CS measurement matrix, as justified within previously-published work [20]. The MFPA is composed of two types of Functional Blocks (FBs): Configurable Digital Blocks (CDBs) and Configurable Analog Blocks (CABs), similar to CABs and CLBs used in previous CMOS-based FPMAs [40, 104]. These FBs are connected via the embedded NVM Crossbar Arrays which perform VMM. The recently-published MTJ-based Look-Up Table (LUT) [107] is used within CDBs to implement Boolean functions via in-memory computing. Additionally, hybrid spin-CMOS ADCs [55] are used within CABs.

Thus, MTJs are investigated for selected processing roles to simultaneously reduce area and energy requirements while providing stochasticity and non-volatility needed for execution of CS. MFPAs can advance a unified platform on a single die accommodating a continuum of information conversion losses and costs targeting CS applications. Design of such a mixed-signal reconfigurable fabric can enable feasible hardware approaches that execute CS algorithms more efficiently than digital FPGA-based or CPU-based implementations, which can then be extended

to low-energy miniaturization for IoT sensing applications. The parallelism enabled by the fabric is readily applicable to other applications relying on VMM, such as artificial intelligence.

2.4 Compressive Sensing (CS)

2.4.1 Sparse Representation of Signals

The sparsity of a signal having length n and k non-zero coefficients is defined as $S = k/n$; a signal is characterized as sparse if the value of S is sufficiently smaller than 1. Many real-world signals are comprised of only a small number of significant coefficients when expressed in the frequency domain and can therefore be approximated using a sparse representation [108]. One way of achieving this is through the Discrete Cosine Transform (DCT), which entails transforming an image, $\mathbf{I} \in \mathbb{R}^{N\times N}$, to its frequency domain representation, $\mathbf{X} \in \mathbb{R}^{N\times N}$, through the operation $\mathbf{X} = \mathbf{DID^T}$, where $\mathbf{D} \in \mathbb{R}^{N\times N}$ is the DCT matrix defined as [109]:

$$D_{ij} = \begin{cases} \frac{1}{\sqrt{N}}, & i = 0 \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(2j+1)i}{2N}\right), & i > 0 \end{cases} \tag{2.16}$$

The original image can be recovered from $\mathbf{X}$ using the inverse operation, $\hat{\mathbf{I}} = \mathbf{D^T X D}$.

DCT is widely used as part of the JPEG standard. JPEG processing involves the following steps: 1) subdivision of an image into blocks of 8×8 pixels, 2) application of DCT to each block, 3) quantization of the coefficients in each block to reduce magnitudes, 4) transmission of the compressed image, and 5) application of the inverse transform to recover the original image [109]. Due to the wide disparity in significance levels of DCT coefficients, only a few coefficients require a high-precision representation; thus, Step 3 is achieved by performing a Hadamard product between the frequency-domain image, $\mathbf{X}$, and a quantization table, $\mathbf{Q}$, which results in less significant coefficients being assigned a progressively coarser quantization [30]. The end result is

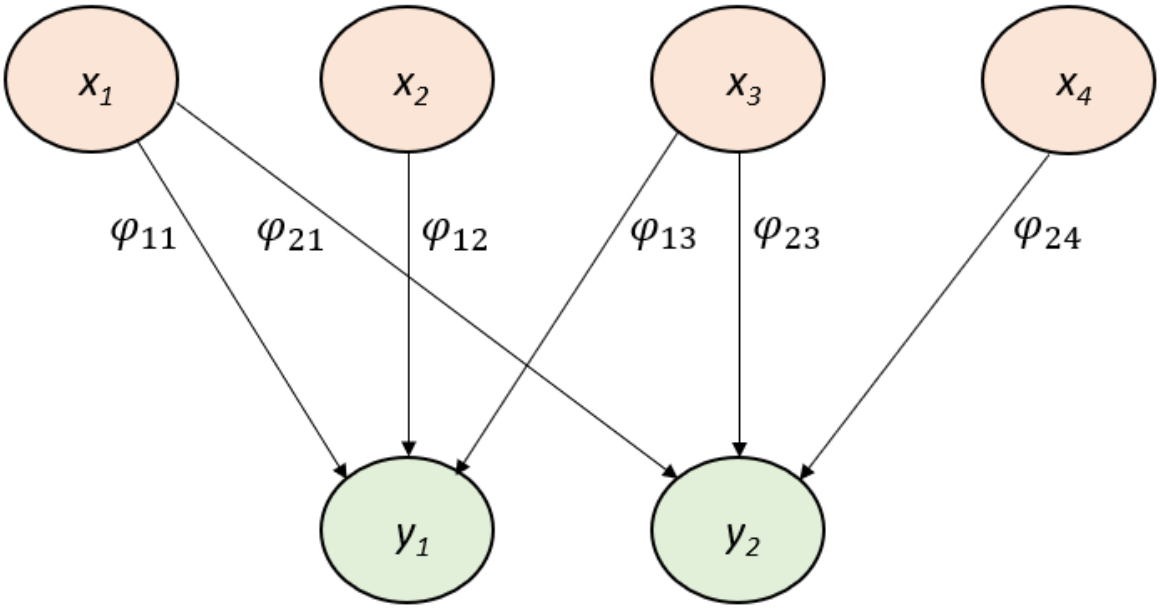


Figure 2.7: Bipartite graph representation of sampling phase of CS.

a sparse representation of the image that can be compressed to yield significant reduction in computational resources with little impact on quality.

2.4.2 Undersampling Sparse Signals

CS is an emerging signal processing technique which allows for undersampling, i.e., sampling at a sub-Nyquist rate, of spectrally-sparse and wideband data. Applications of CS include reduction of power consumption and complexity in 5G communication networks [21], and reduction of sampling duration in time-critical applications such as MRI [43]. CS consists of a sampling phase, followed by a reconstruction phase. In the sampling phase, a linear transformation is applied to a sparse signal $\boldsymbol{x}\epsilon\mathbb{R}^n$ via the measurement matrix, $\boldsymbol{\Phi}\epsilon\mathbb{R}^{m\times m}$, to obtain a compressed measurement vector, $\boldsymbol{y}\epsilon\mathbb{R}^m$, with $m \ll n$. This mapping may be represented using a bipartite graph where each signal coefficient, x_i, is connected to measurement y_j via the edge φ_{ji}, as Figure 2.7 shows.

After sampling and transmission of measurements, the receiver must then solve an undetermined system of linear equations to reconstruct the original signal. It has been shown that in order to exactly reconstruct the signal with a probability $1-\delta$, the minimum number of measurements is given by:

$$m \geq C\mu^2(\boldsymbol{\Phi}, \boldsymbol{\Psi})k \log(n/\delta) \tag{2.17}$$

where C is a constant and μ is the coherence between the measurement matrix and the basis, $\boldsymbol{\Psi}$, given by:

$$\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = \sqrt{n} \max_{1\leq i,j\leq n} \left|\langle \boldsymbol{\varphi_i}, \boldsymbol{\psi_j} \rangle\right| \tag{2.18}$$

where $\boldsymbol{\varphi_i}$ and $\boldsymbol{\psi_j}$ are row and column vectors of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, respectively. It follows from Eq. 2.17 that $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi})$ should be minimized in order to minimize *m*. If rows of $\boldsymbol{\Phi}$ are normalized, then $\sum_{j=1}^{n} \left|\langle \boldsymbol{\varphi_i}, \boldsymbol{\psi_j} \rangle\right|^2 = 1$ for all *i*, since the change of basis is a unitary transformation. Thus, $\max_{1\leq i,j\leq n} \left|\langle \boldsymbol{\varphi_i}, \boldsymbol{\psi_j} \rangle\right|$ attains a minimum value of $1/\sqrt{n}$ if $\left|\langle \boldsymbol{\varphi_i}, \boldsymbol{\psi_j} \rangle\right| = 1/\sqrt{n}$ for all *i* and *j*. Hence, it is desirable to have a dense measurement matrix in the same basis that gives a sparse signal [108].

One possible approach to the CS reconstruction problem is to choose the solution with the lowest sparsity. This translates directly to the minimization problem:

$$\hat{\mathbf{x}} = \text{argmin}\|\boldsymbol{x}\|_0 \text{ s.t. } \boldsymbol{y} = \boldsymbol{\Phi x}. \tag{2.19}$$

Due to this problem being NP-hard [110], it is often reformulated as the basis pursuit problem:

$$\hat{\mathbf{x}} = \text{argmin}\|\boldsymbol{x}\|_1 \text{ s.t. } \boldsymbol{y} = \boldsymbol{\Phi x}, \tag{2.20}$$

which can be solved using convex optimization techniques. It is shown that $\hat{\mathbf{x}}$ is an accurate reconstruction of the original signal vector if the measurement matrix satisfies the *Restricted Isometry Property (RIP)* [108], i.e., that for any 2*k*-sparse vector $\boldsymbol{x}$,

$$\|\boldsymbol{x}\|_2^2(1-\delta_{2k}) \leq \|\boldsymbol{\Phi x}\|_2^2 \leq \|\boldsymbol{x}\|_2^2(1+\delta_{2k}), \tag{2.21}$$

such that $\delta_{2k} < 1$. RIP prevents any vector of sparsity $2k$ from being in the null space of $\boldsymbol{\Phi}$. Thus, given two *k*-sparse vectors, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, such that $\boldsymbol{\Phi x}_1 = \boldsymbol{\Phi x}_2$, it follows that $\boldsymbol{x}_1 - \boldsymbol{x}_2 = 0$ since the difference of k-sparse vectors must be $2k$ sparse. Thus, RIP ensures unique solutions to Eq. 2.19.

Moreover, it has been shown [108] that if $\delta_{2k} < \sqrt{2} - 1$, then:

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_1 \leq C_0 \|\boldsymbol{x} - \boldsymbol{x_k}\|_1 \tag{2.22}$$

where C_0 is a constant, $\hat{\boldsymbol{x}}$ is the solution to Eq. 2.20 and $\boldsymbol{x_k}$ is equal to $\boldsymbol{x}$ with all components, except for the largest *k* components, set to zero. Thus, Eq. 2.22 guarantees a unique solution to the basis pursuit problem if $\boldsymbol{x}$ is *k*-sparse and RIP is satisfied with the given condition.

In light of the above considerations, it has been seen [20, 108] that random measurement matrices, e.g., having column vectors randomly chosen from the unit sphere in $\mathbb{R}^m$, or having elements chosen from a Gaussian or Bernoulli distribution, are ideal in that they are largely incoherent with any given basis while simultaneously obeying RIP.

Figure 2.8 illustrates a common IoT image compression flow, which consists of the following steps: 1) The image is partitioned into N×N blocks, 2) DCT is performed on each block, 3) quantization is performed via a Hadamard product with quantization matrix, **Q**, 4) the image matrix, **X***, is mapped to a vector representation, $\boldsymbol{x}$, and compressively sampled, 5) the compressed measurement vector is transmitted to the receiver, which reconstructs the signal, and 6) the image

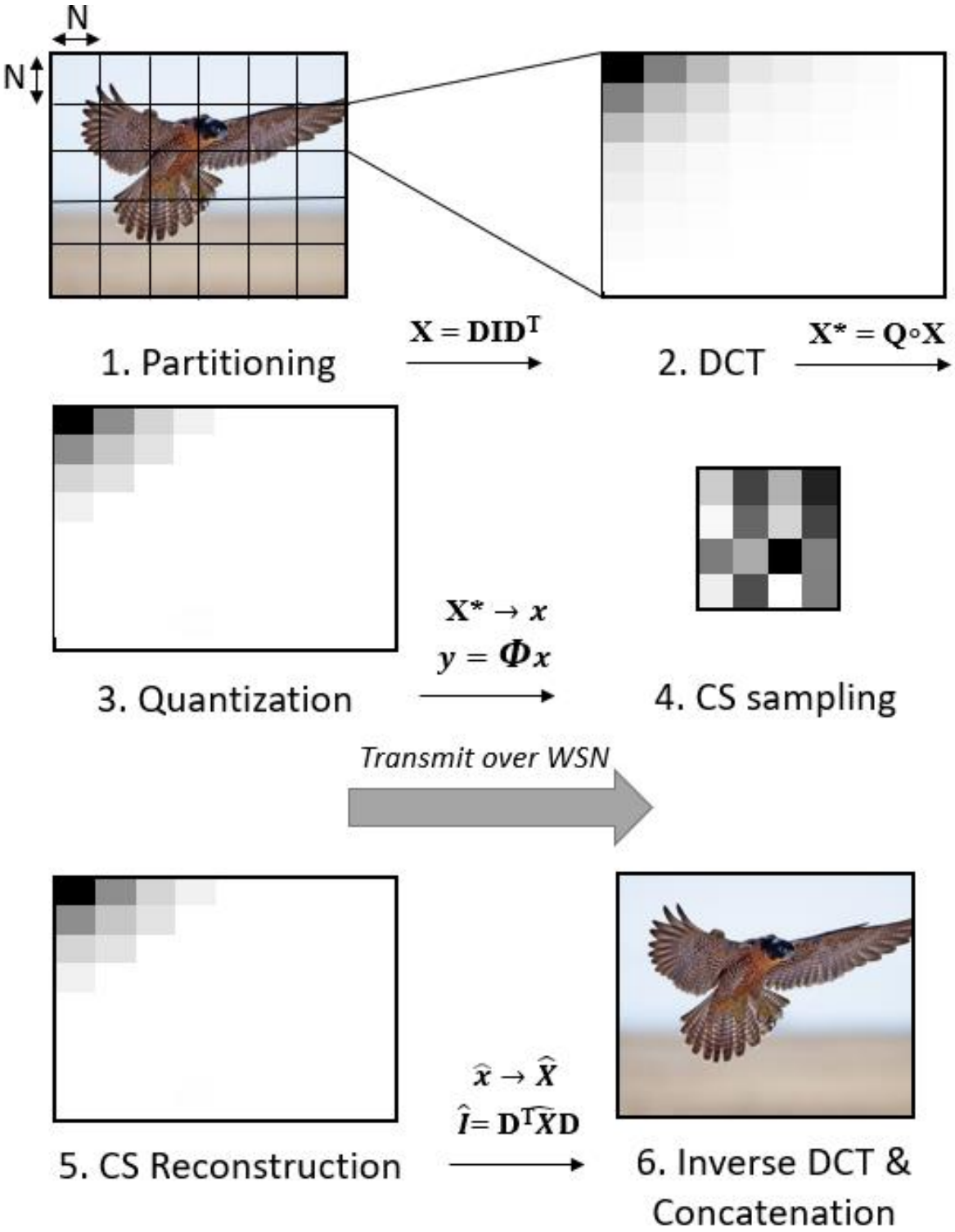


Figure 2.8: IoT signal compression flow, consisting of compressive sampling in the DCT domain, transmission, and reconstruction by the receiver.

is recovered by converting each block back to spatial domain via inverse DCT and concatenating the blocks. Due to space constraints, vectors in Steps 4 and 5 are represented as matrices in the figure.

2.4.3 Non-uniform Sampling

Real-world signals often exhibit Regions of Interest (RoIs), i.e., subsets of the signal that may be more critical to accurately reconstruct than others. One approach is to partition the measurement matrix into *t* sub-matrices, with each sub-matrix having dimensions $m \times n\alpha_t$ where α_t is the fraction of columns of $\boldsymbol{\Phi}$ occupied by sub-matrix *t*, as Figure 2.9 illustrates. The density of non-zero elements in each sub-matrix is determined by the importance level of the subset of $\boldsymbol{x}$ that the sub-matrix maps to through the VMM operation. A higher density of non-zero elements corresponds to a higher sampling rate for more important signal coefficients. In the bipartite graph model, there are more edges connected to nodes that represent these coefficients. This translates to a disproportionate amount of sensing energy being allocated to the RoI [23].

Key parameters characterizing $\boldsymbol{\Phi}$ are [23]: 1) the matrix dimensions, *m* and *n*, 2) the number of non-zero elements per row, *L*, 3) the set of relative widths of the *t* sub-matrices, $\alpha_1, \ldots, \alpha_t$, and 4)

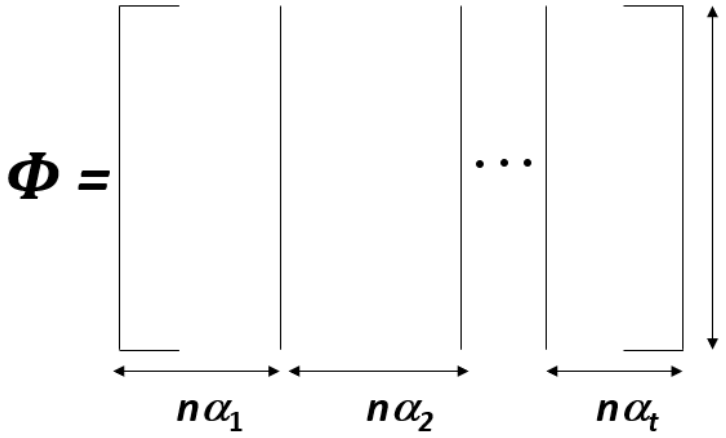


Figure 2.9: Measurement matrix partitioned into *t* sub-matrices, where sub-matrix densities are determined by signal importance levels.

the set of values $p_1, \ldots, p_t$, where p_i represents the probability that a randomly chosen nonzero element in $\boldsymbol{\Phi}$ belongs to a particular column in sub-matrix *i*. These parameters are not all independent: they must satisfy the constraints $\sum_{i=1}^{t} \alpha_i = 1$ and $\sum_{i=1}^{t} p_i \alpha_i n = 1$. In the latter equation, the expression $p_i \alpha_i n$ represents the fraction of non-zero elements occurring in sub-matrix *i*, compared to the measurement matrix as a whole.

Explorations in partitioned CS were first motivated by speed ups available through parallel processing [111] and [112]. This same approach was quickly applied to non-uniform CS by authors motivated by observations including varying pixel saliency levels in images [113] and heterogeneity in WSNs [114]. Various methodologies have been proposed for RoI detection, including Bayesian inference [115] and reinforcement learning [116].

2.4.4 An Overview of Reconstruction Algorithms

Convex optimization using basis pursuit is one of several methods of reconstructing a signal from its measurements. An alternate convex optimization approach is Least Absolute Shrinkage and Selection Operator (LASSO), which solves the following optimization problem [117]:

$$\hat{\mathbf{x}} = \text{argmin}\left(\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1\right) \tag{2.23}$$

where λ is a Lagrange multiplier which specifies the balance between low-error and low-sparsity solutions. LASSO is useful for optimal recovery of signals transmitted over noisy channels.

Approximate Message Passing (AMP) is a reconstruction algorithm with low complexity, allowing for fast convergence. Derived from the Iterative Soft Thresholding (IST) technique, AMP approximates the signal in Iteration *i* as [118]:

$$\hat{\mathbf{x}}_{\boldsymbol{i}} = \eta(\hat{\mathbf{x}}_{\boldsymbol{i-1}} + \boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{r}_{\boldsymbol{i-1}}) \tag{2.24}$$

where $\eta(x)$ is a soft-thresholding function and the residual, $\boldsymbol{r}_{\boldsymbol{i}}$ is defined as:

$$\boldsymbol{r_i} = \boldsymbol{y} - \boldsymbol{\Phi}\hat{\mathbf{x}}_{\boldsymbol{i}} + b_i \boldsymbol{r_{i-1}} \tag{2.25}$$

and $b_i = \frac{1}{m} \|\hat{\boldsymbol{x}}_{\boldsymbol{i}}\|_0$. The key difference between IST and AMP is the last term in Eq. 2.25, which allows for improvement in convergence rate without increasing complexity. A more complete treatment of AMP is given in [119].

Greedy algorithms provide an alternate class of reconstruction techniques which can provide for computational benefits. Orthogonal Matching Pursuit (OMP) is one example of a greedy reconstruction algorithm which selects the column of $\boldsymbol{\Phi}$ most correlated with the residual $\boldsymbol{r}_{i-1}$ in each iteration. The signal can then be approximated in Iteration *i* by solving the least squares problem [110]:

$$\hat{\mathbf{x}}_{\boldsymbol{i}} = \text{argmin}(\|\boldsymbol{y} - \boldsymbol{\Phi}_{\boldsymbol{i}}\hat{\boldsymbol{x}}\|_2) \tag{2.26}$$

where $\boldsymbol{\Phi}_{\boldsymbol{i}}$ is initially set to zero and is augmented by the selected column of $\boldsymbol{\Phi}$ at each iteration.

Regularized Orthogonal Matching Pursuit (ROMP) is similar to OMP but beneficial in noisy applications [117]. ROMP also starts each iteration by computing correlations between columns of $\boldsymbol{\Phi}$ and the residual. Instead of selecting a single column of $\boldsymbol{\Phi}$ at each iteration, ROMP selects multiple columns to allow for the possibility of corruption due to noise.

Finally, Compressive Sampling Matching Pursuit (CoSAMP) seeks to reconstruct $\boldsymbol{x}$ by identifying its support set, i.e., set of indices having non-zero coefficients [117]. CoSAMP begins by computing the signal proxy given by $\boldsymbol{c}_i = \boldsymbol{\Phi}^T \boldsymbol{r}_{i-1}$ where $\boldsymbol{r}_{i-1}$ is the residual of the previous iteration. Next, the $2k$ coefficients of $\boldsymbol{c}_i$ with highest magnitude are used, together with the estimate from the previous iteration, to estimate the support set of $\boldsymbol{x}$. As in OMP and ROMP, the corresponding columns of $\boldsymbol{\Phi}$ are then combined to form $\boldsymbol{\Phi}_{\boldsymbol{i}}$ and the least squares problem given in

Eq. 2.26 is solved to estimate the signal, $\hat{\mathbf{x}}_{\boldsymbol{i}}$. At the conclusion of each iteration, the estimated support set is pruned by keeping only indices of the *k* largest coefficients in $\hat{\mathbf{x}}_{\boldsymbol{i}}$.

While the above provides a brief overview of selected CS reconstruction algorithms that commonly appear in literature, a more complete treatment of reconstruction algorithms can be found in [117].

2.4.5 Hardware Implementation of CS

Several challenges must be overcome in hardware implementation of CS sampling and reconstruction. Sampling requires a random number generator for populating the measurement matrix as well as computationally intensive operations such as VMM. Reconstruction requires multiple iterations involving VMM and matrix inversion. Thus, hardware and software optimizations are necessary to accommodate CS in resource-limited applications such as IoT.

Hardware implementations of sampling have included Massoud's crossbar design [24] leveraging RRAM devices randomly programmed using Linear Feedback Shift Registers (LFSRs). Qian [25] eliminated the need for LFSRs by achieving randomness through the intrinsic process variation present within RRAM devices. Moreover, Salehi [20] proposed an MRAM-based crossbar array consisting of SHE-MTJs for non-uniform sampling, as shown in Figure 2.10. In this approach, stochastic properties of low-barrier MTJs are leveraged by using p-bit devices to populate the matrix. P-bits are especially useful for non-uniform sampling since they allow for tunable stochasticity, i.e., the capability to adjust the probability of non-zero outputs by means of an input voltage. While the above approach using single-bit devices is useful for Bernoulli matrices, Salehi extended this approach to accommodate for Gaussian matrices as well, by using SOT-MRAM devices which exhibit multi-bit precision and intrinsic stochasticity at the cost of increased area.

Hardware implementation of reconstruction algorithms has focused on maximization of parallelism and minimization of complexity. Septimus and Steinberg [110] presented an FPGA implementation of OMP, where the Moore-Penrose pseudoinverse, $\boldsymbol{\Phi_i}^{\dagger} = (\boldsymbol{\Phi_i}^{\boldsymbol{T}}\boldsymbol{\Phi_i})^{-\mathbf{1}}\boldsymbol{\Phi_i}^{\boldsymbol{T}}$, was used

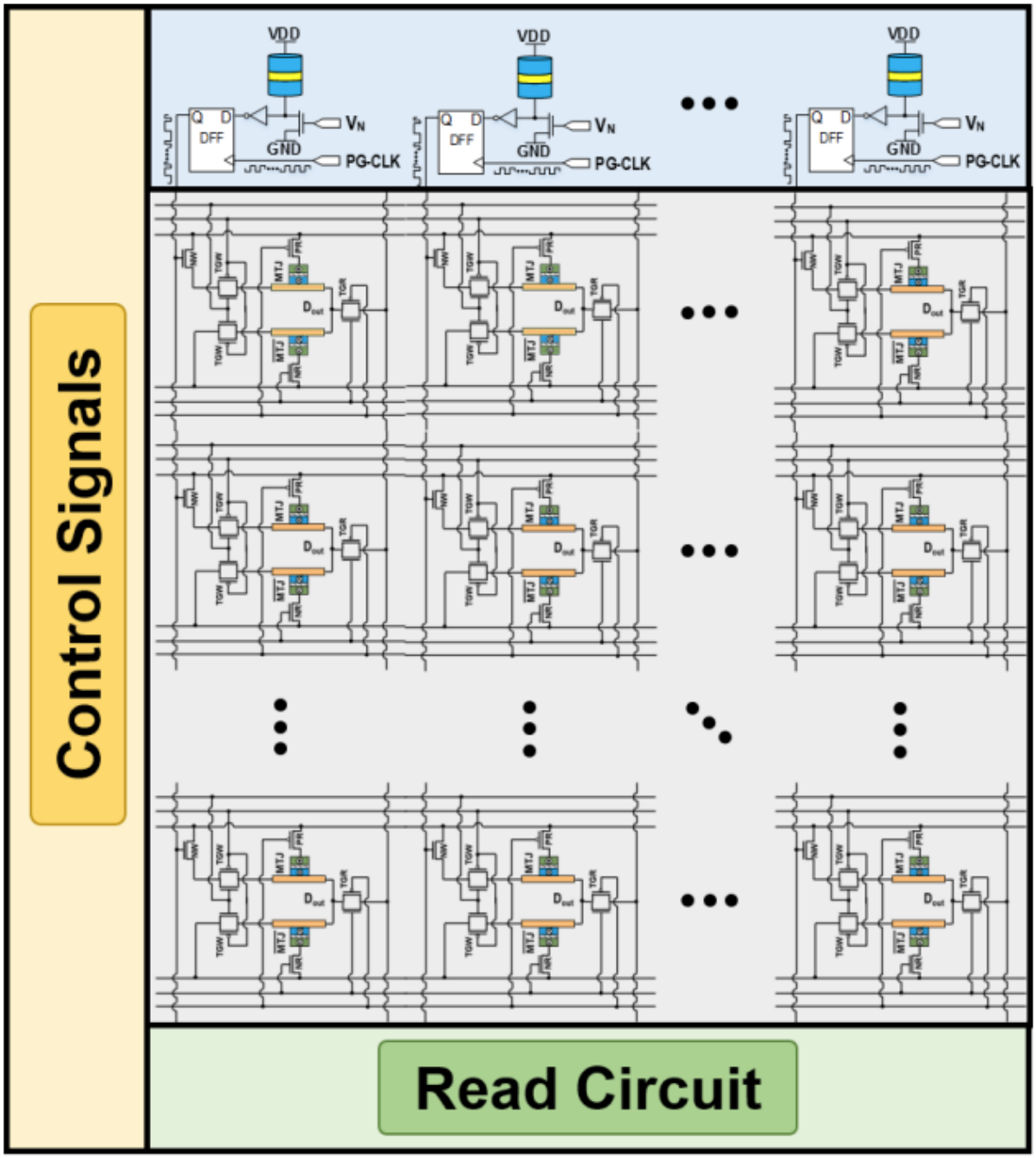


Figure 2.10: Salehi's implementation of the non-uniform measurement matrix using an MRAM-based crossbar populated by p-bit devices in each column [20].

to reduce the complexity of matrix inversion operations needed for least squares minimization. The authors were able to reduce this computation to that of inverting the symmetric matrix, $\boldsymbol{C} = \boldsymbol{\Phi_i}^{\boldsymbol{T}}\boldsymbol{\Phi_i}$, via the Alternate Cholesky Decomposition, i.e., $\boldsymbol{C} = \boldsymbol{LDL}^{\boldsymbol{T}}$ where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{D}$ is a diagonal matrix. Alternate FPGA implementations have been given for algorithms such as OMP and AMP [120]. Liu [121] proposed an MCA-based approach to basis pursuit, and Le Gallo presented a similar approach to AMP [122]. An implementation of OMP using the MFPA fabric was presented in [27].

2.5 Deep Belief Network (DBN)

2.5.1 Restricted Boltzmann Machine (RBM)

Restricted Boltzmann Machines (RBMs) are a class of recurrent stochastic neural network [123] in which the energy of the network in state *k* is determined by:

$$E(k) = -\sum_i s_i^k b_i - \sum_{i<j} s_i^k s_j^k w_{ij} \tag{2.27}$$

where s_i^k refers to the state of node *i* while the network is in state *k*, and w_{ij} represents the weight between nodes *i* and *j*. Each node in an RBM has a probability of being in state 1 given by:

$$P(s_i = 1) = \sigma\left(b_i + \sum_j w_{ij} s_j\right) \tag{2.28}$$

where σ represents the sigmoid function. Over time, a Boltzmann distribution is reached where the probability of finding the system in state *k* is defined as:

$$P(k) = \frac{e^{-E(k)}}{\sum_u e^{-E(u)}} \tag{2.29}$$

where the summation in the denominator is taken over all possible states of the system. An RBM is a two-layer neural network consisting of a visible layer and hidden layer which can be trained using the Contrastive Divergence algorithm [123], consisting of the following steps:

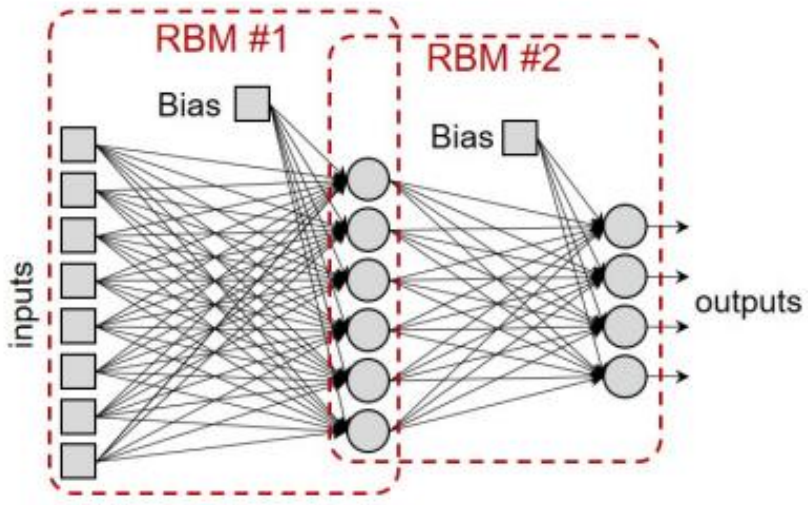


Figure 2.11: DBN structure consisting of one visible layer and two hidden layers [124].

1) Apply a training vector, $\boldsymbol{v}$, to the visible layer and sample hidden layer outputs, $\boldsymbol{h}$.
2) Feed back the hidden layer outputs and sample the resulting input, $\boldsymbol{v}$'.
3) Apply $\boldsymbol{v}$' as an input to the visible layer and sample the resulting hidden layer outputs, $\boldsymbol{h}$'.
4) Update the weights according to the equation, $\Delta W = \eta(\boldsymbol{v}\boldsymbol{h}^T - \boldsymbol{v}'\boldsymbol{h}'^T)$ where η is the learning rate.

Deep Belief Networks (DBNs) can be realized by stacking RBMs, as Figure 2.11 shows. Training is achieved through an iterative application of the Contrastive Divergence algorithm.

As shown in Figure 2.12, DBNs may be implemented using crossbar arrays for computationally efficient VMM. Weights are represented by the state of memristive devices in the array; since device conductance value cannot be negative, two devices are commonly used to represent a single weight, using an architecture such as that shown in Figure 2.4. DBNs require a stochastic neuron at each output for computation of the activation function, which can be achieved in MRAM via embedded p-bit devices as illustrated in Figure 2.4.

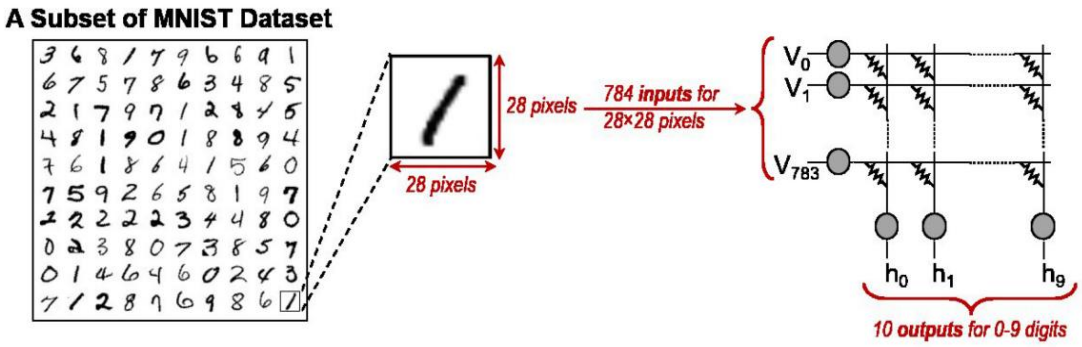


Figure 2.12: 784 × 10 DBN implemented using crossbar for MNIST digit recognition [123].

2.5.2 Probabilistic Inference Network Simulator (Pin-Sim)

DBN simulations on the MNIST dataset can be readily performed at both the software and hardware level, using the Probabilistic Inference Network Simulator (PIN-Sim) [123]. PIN-Sim consists of five modules: first, *trainDBN* reads the training images in MATLAB and outputs weight and bias matrices using the Contrastive Divergence algorithm. A second MATLAB module, *mapWeight*, converts the weight and bias data into device conductance values. First, *mapWeight* splits weights and biases into positive and negative values, i.e.,

$$w_{ij}^{+} = \begin{cases} w_{ij}, & w_{ij} \geq 0 \\ 0, & w_{ij} < 0 \end{cases} \qquad w_{ij}^{-} = \begin{cases} 0, & w_{ij} \geq 0 \\ -w_{ij}, & w_{ij} < 0 \end{cases} \tag{2.30}$$

$$b_{j}^{+} = \begin{cases} b_{j}, & b_{j} \geq 0 \\ 0, & b_{j} < 0 \end{cases} \qquad b_{j}^{-} = \begin{cases} 0, & b_{j} \geq 0 \\ -b_{j}, & b_{j} < 0 \end{cases} \tag{2.31}$$

Next, *mapWeight* uses the following equations to set conductance values based on weights and biases:

$$\forall w_{ij} \, \epsilon \, (W^{+}, W^{-}) : gw_{ij} = \frac{(g_{max} - g_{min}) \times (w_{ij} - w_{min})}{w_{max} - w_{min}} + g_{min} \tag{2.32}$$

$$\forall b_{ij} \, \epsilon \, (B^{+}, B^{-}) : gb_{ij} = \frac{(g_{max} - g_{min}) \times (b_{ij} - b_{min})}{b_{max} - b_{min}} + g_{min} \tag{2.33}$$

where W^+, W^- represent weight matrices, B^+, B^- represent bias matrices, g_{max} and g_{min} are maximum and minimum conductance values of all weighted connections in the array, w_{max} and w_{min} represent maximum and minimum elements in weight matrices, and b_{max} and b_{min} represent maximum and minimum elements in bias matrices.

Finally, *mapWeight* converts the conductance values to resistance values and quantizes these values based on device capabilities of the hardware, using the following equation:

$$\forall g_{ij} \epsilon (GW^+, GW^-, GB^+, GB^-): r_{ij} = \frac{\text{round}(Q \times 1/g_{ij})}{Q} \tag{2.34}$$

where GW^+, GW^-, GB^+, GB^- represent positive and negative weight and bias conductance matrices, and Q is the quantization factor.

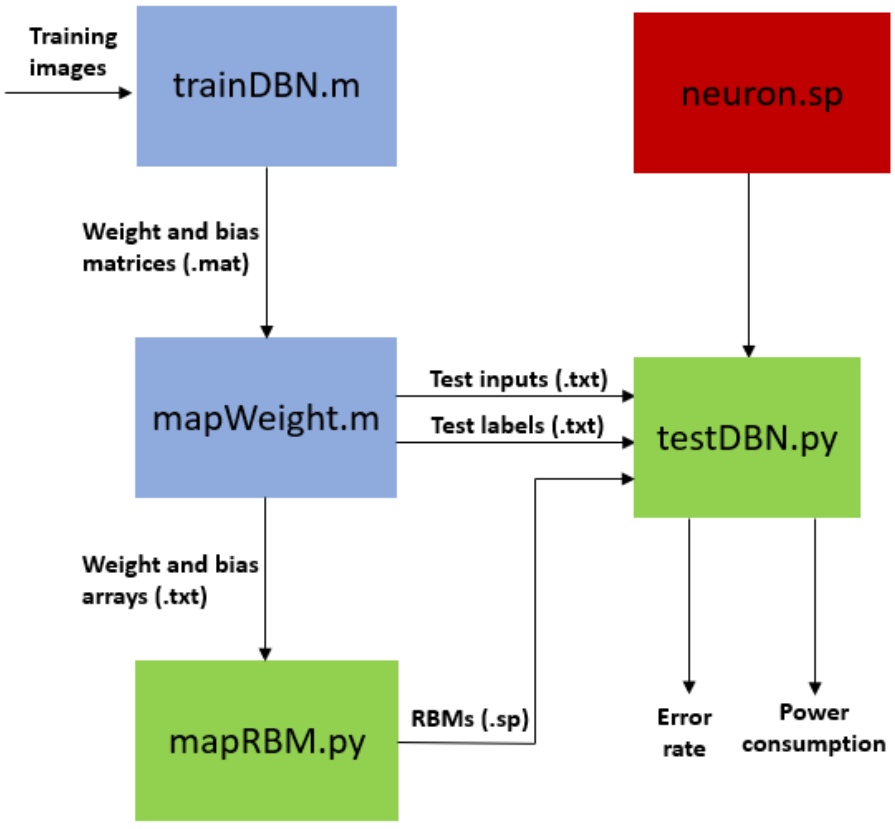


Figure 2.13: Logical flow of PIN-Sim, including the five main modules involved in DBN simulation.

The third PIN-Sim module is *mapRBM*, a Python script which generates SPICE representations of multiple crossbar weighted arrays based on the outputs of *mapWeight* and the given network topology. A final Python module, *testDBN*, executes a SPICE circuit simulation of the DBN to determine classification error rate as well as power consumption. The inputs to the *testDBN* module consist of the outputs of *mapWeight* and *mapRBM* as well as the module, *neuron*, which is a SPICE representation of the circuit used for computing the activation function. A visual description of PIN-Sim is given in Figure 2.13.

2.5.3 Probabilistic Interpolation Recoder (PIR)

The stochastic analog outputs of probabilistic neurons in each DBN layer must be integrated and converted to digital for a fully operational architecture. This is conventionally achieved using a resistor-capacitor (RC) circuit followed by ADC, as shown in Figure 2.14a. However, such an approach is not ideal for resource limited applications due to the power and area demands of the ADC. The Probabilistic Interpolation Recoder (PIR) [124] provides an alternative approach to stochastic output digitization with improved resource utilization.

Sample and Count-based PIR (SC-PIR) integrates the neuron outputs using an RC circuit and samples the resulting outputs, $Neuron_{OUT}$, at each positive edge of the clock (CLK). A counter is then used to accumulate the sampled outputs by incrementing by one whenever $Neuron_{OUT}$ is greater than $V_{DD}/2$. The final outputs are returned as an *n*-bit digital value, given by $OUT_{n-1}…OUT_0$, which can be reset to zero through a CTRL signal, as shown in Figure 2.14b.

Sample and Shift-based PIR (SS-PIR) is an alternative design which is further optimized for energy consumption. SS-PIR also samples integrated neuron outputs at each positive edge of the clock. SS-PIR outputs are stored using a bidirectional shift register which shifts right or left if the

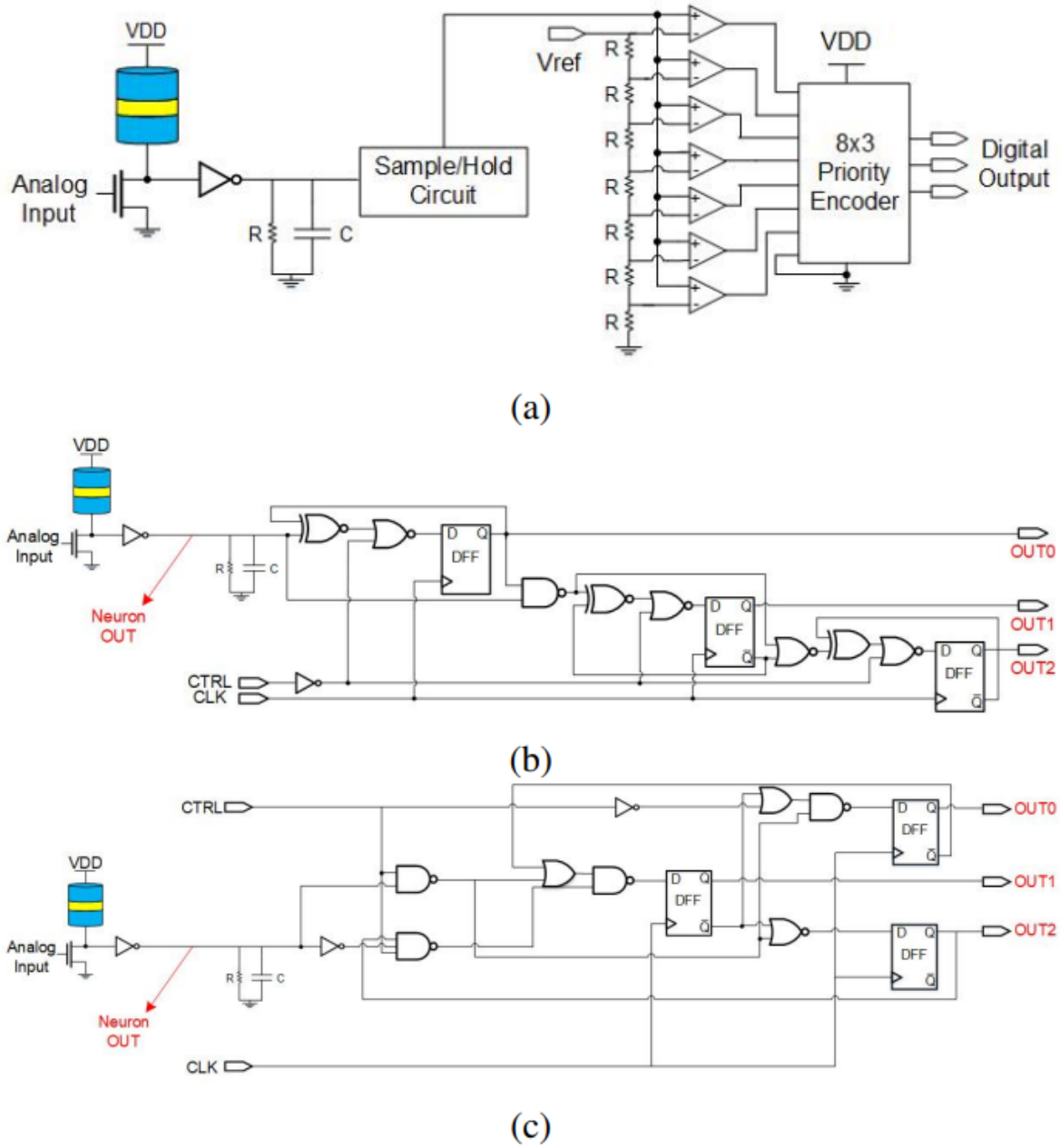


Figure 2.14: Interpolation of neuron outputs using a) ADC, b) 3-bit SC-PIR circuit, and c) 3-bit SS-PIR circuit [124].

integrated neuron outputs are less than or greater than $V_{DD}/2$, respectively. The shift register can be reset using a CTRL signal, as shown in Figure 2.14c.

SC-PIR and SS-PIR both show slightly increased error, compared to a conventional ADC, when set to 3-bit precision in MNIST digit classification applications. However, both variations of PIR yield significant improvements in resource utilization as measured by the Energy-Error-Product (EEP). PIR can be scaled to greater precision levels at reduced energy cost: 5-bit SS-PIR yields lower error rate and significantly lower energy consumption than a 3-bit ADC [124].

CHAPTER 3: NON-UNIFORM CS VIA OHMIC VOLTAGE ATTENUATION[3]

3.1 Voltage Degradation in MRAM-based Crossbars

As seen in Section 2.2.2, voltage attenuation due to parasitic line resistance is a well-known effect in crossbar array architectures. This effect is more significant in MRAM-based arrays due to the comparatively low resistance of MRAM devices compared to alternatives such as RRAM. In this chapter, simulation results are performed in MATLAB using the Modified Nodal Analysis technique [125, 126] to determine the significance of the voltage attenuation in 64×64 and 128×128 arrays.

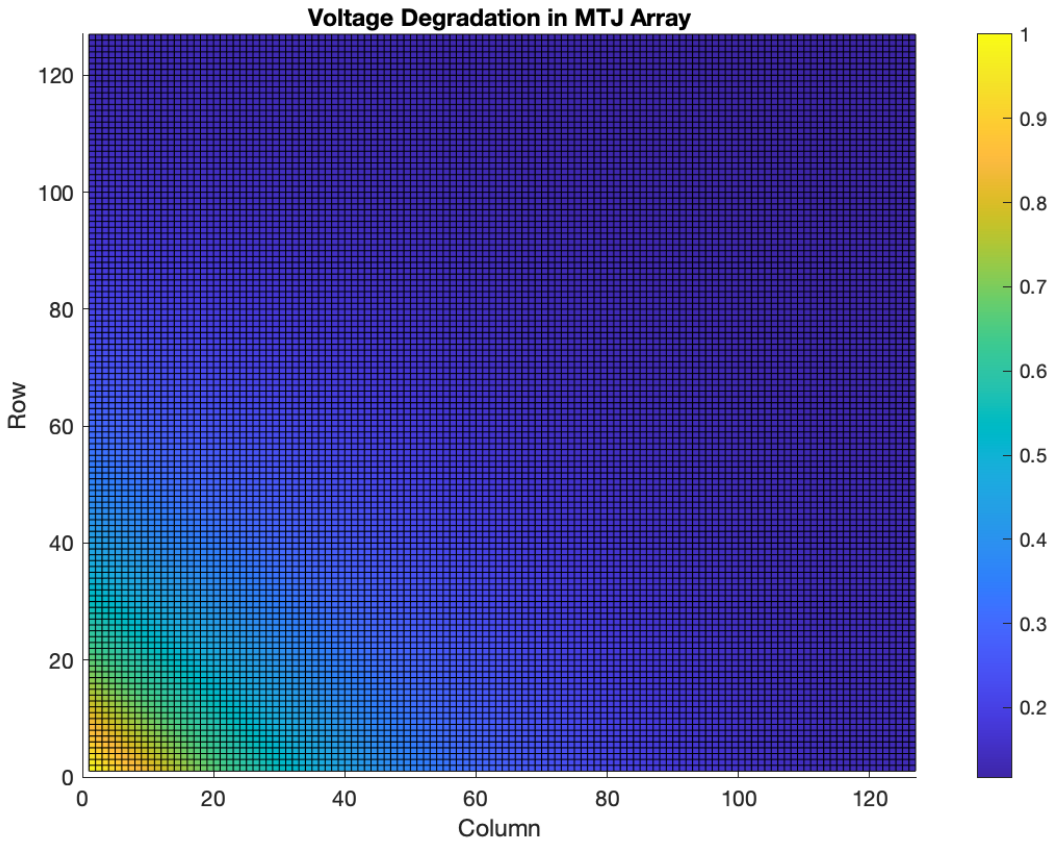


Figure 3.1: Voltage difference across elements of 128×128 MRAM-based crossbar with each element set to a resistance of 5600Ω.

[3] ©IEEE. Part of this chapter is reprinted, with permission, from [132].

Figure 3.1 is a heat map showing voltage drop across each element of an MRAM-based MCA with each device set to a P-state resistance value of 5600Ω and 1V applied to each row from the left side. The parasitic line resistance is 2.5Ω per cell and no mitigation strategies such as selector diodes or pass transistors are used. The figure illustrates the severity of the voltage degradation effect, which is >50% for the majority of the array and approaching a 90% loss in the top right corner.

Figure 3.2 shows the severity of voltage degradation along elements adjacent to the first 64 columns in the top row of an MCA having a full array of MRAM devices carrying a resistance of 5600Ω. The figure demonstrates that even for relatively small array dimensions of 64×64, an attenuation exceeding 50% can be attained past a certain threshold of line resistance.

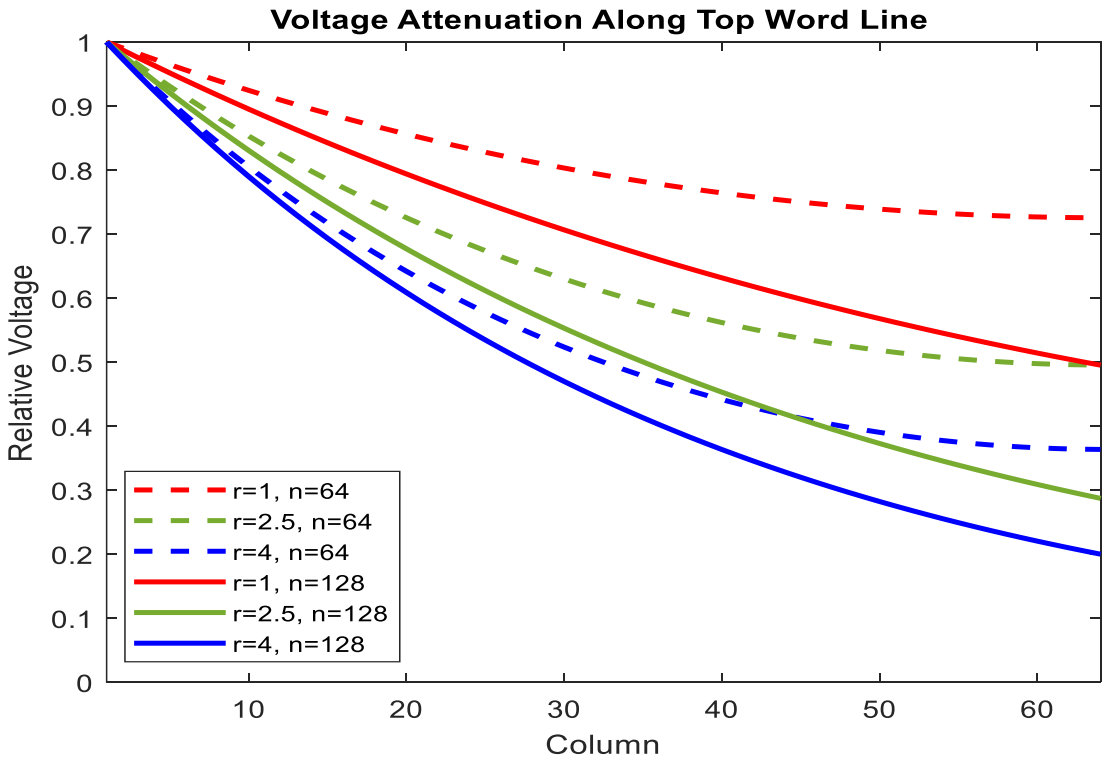


Figure 3.2: Relative voltage attenuation along the top word line of an MCA, for a variety of array sizes, n, and line resistance values, r.

3.2 Non-Uniform Measurement Matrix Implementation

Previous work [20] has demonstrated the possibility of non-uniform CS matrix implementation by MRAM-based MCAs. One approach to writing the matrix is by including a p-bit for each column to generate *m* outputs over *m* clock cycles, writing the crossbar one row at a time through a power-gated D flip-flop. Following Eq. 2.15, p-bits corresponding to columns requiring higher densities of non-zero elements are simply given a higher input voltage.

The proposed architecture supplies appropriate inputs for each p-bit device. A Stochastic Bitstream Generator (SBG), shown in Figure 3.3, is connected to each column of an MCA. During the control cycle, a voltage V_{input} is applied to WL_1 with every other word line and bit line in the array held at ground. CTRL is high and $\overline{\text{CTRL}}$ is low. Thus, capacitor C is charged to a voltage $V_{in} + V_{bias}$, where V_{in} is determined by a multiplexer output from *t* possible inputs, each one being sourced from a different location along the first word line in the MCA. V_{bias} is a DC voltage offset necessary to reach the p-bit operational range and is the same in every column. Thus, the proposed architecture generates a spectrum of voltages from a single DC voltage source.

After the control cycle ends, CTRL and $\overline{\text{CTRL}}$ are switched low and high, respectively, to lock in the capacitor voltage. At this time, the power-gated clock connected to the D flip-flop, DFF, begins to count *m* clock cycles, with the output connected to the access transistors of the devices in that column. During each clock cycle, a voltage V_w is applied to the corresponding word line in the array with all other word_lines and bit lines held at zero. V_w is chosen such that it is sufficient to write to an SHE-MRAM device. Thus, in each column, the device in row *i* is written to if and only if the i^{th} output of that column's D flip-flop is a logic 1.

The proposed architecture makes use of the Ohmic voltage degradation effect in MCAs to intrinsically realize non-uniform CS. The resistance along both the bit lines and word lines of the

MCA is modeled as r Ω per cell. Due to this resistance, the voltage along the word lines decays as one proceeds away from the source.

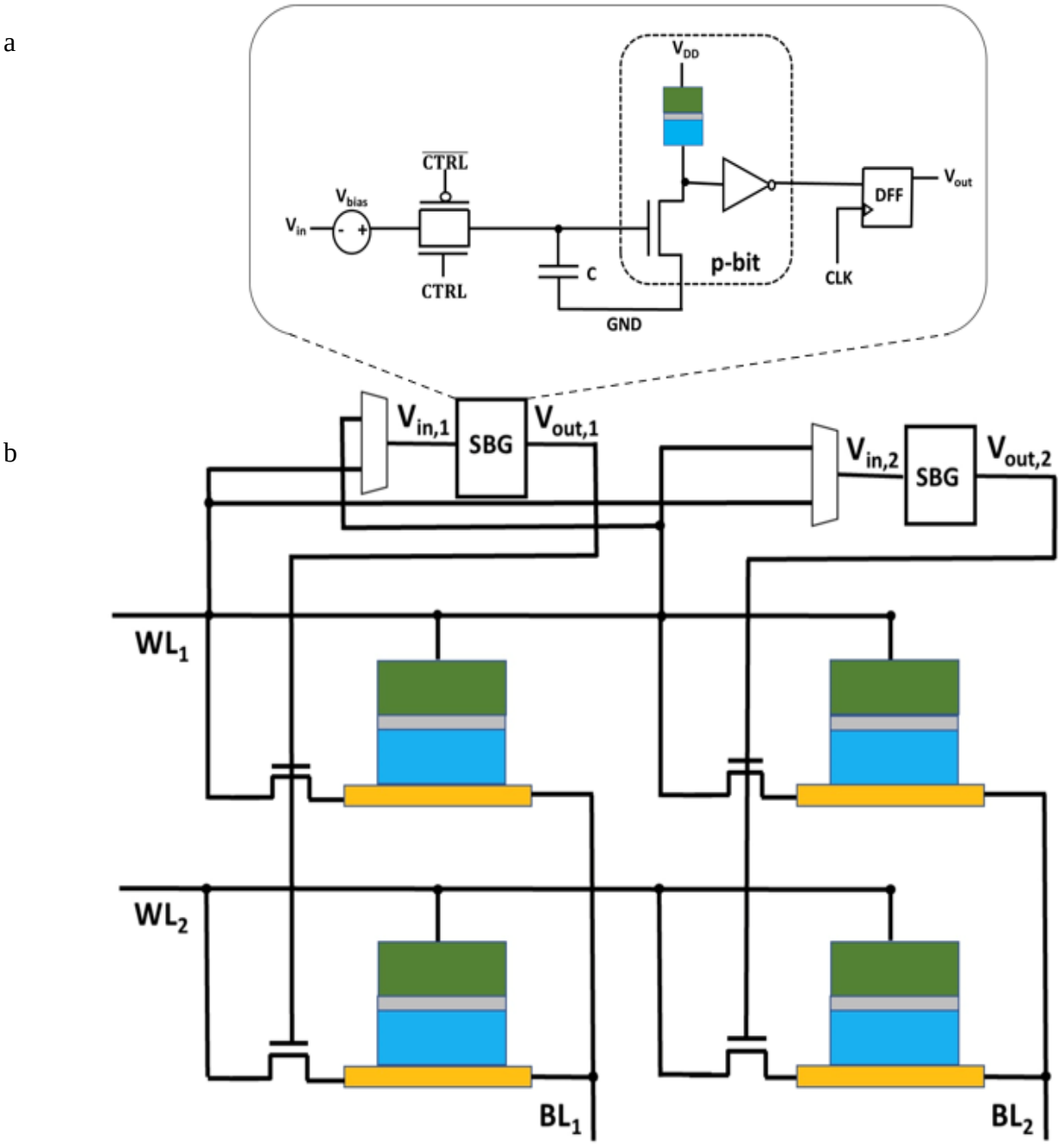


Figure 3.3: a) Stochastic Bitstream Generator (SBG) providing m output bits, with the fraction of 1's determined by the input voltage; one SBG is present per MCA column. b) Implementation of a 2×2 MRAM-based MCA.

Modified Nodal Analysis is implemented using MATLAB and verified in HSPICE to determine MCA node voltages and power consumption, with MTJs being modeled as linear with resistance R. SPICE and MATLAB are used to characterize the p-bit, after which results are fed into the p-bit device model, which yields a probability of obtaining a logic 1 output given by:

$$\mathrm{P}(1) = \frac{1}{2}\left(1 + \tanh\left(\frac{V_{in}-V_c}{V_0}\right)\right) \tag{3.1}$$

which is a horizontal translation of Eq. 2.15. In the presented research, model parameters of V_c = 0.4V and V_0 = 0.04V are used to attain agreement with experimental data [20].

Simulations are performed using the 14nm HP-FinFET Predictive Technology Model (PTM) library [127], with V_{DD} = 0.8V. MCA dimensions of 64×64 and 128×128 are considered. The sizes chosen in our simulations are representative of those commonly found in the literature [125], where larger networks are often mapped onto a grid of smaller arrays such as the ones considered herein. Moreover, since the severity of voltage degradation correlates with array size, it follows that the proposed design is also applicable to larger-sized crossbar arrays.

The case of two sub-matrices corresponding to a single RoI is considered. In this scenario, p-bit inputs, V_{in}, are sourced from the first word line in the array: from Column 4 for columns within the RoI, and from Column 48 for all other columns. To determine the robustness of the proposed architecture, a variety of line resistance values, r, corresponding to values found in the literature [125], [128, 129], are tested to determine the necessary voltage parameters to maintain the measurement matrix characterizations listed in Table 3.1. Parameters of the embedded three-terminal MTJs [56] are provided in Table 3.2.

3.3 Simulation Results

Figure 3.4 shows that target measurement matrix parameters indicated in Table 3.1 are achieved under feasible input and bias voltages for a variety of line resistances. The necessary input voltage

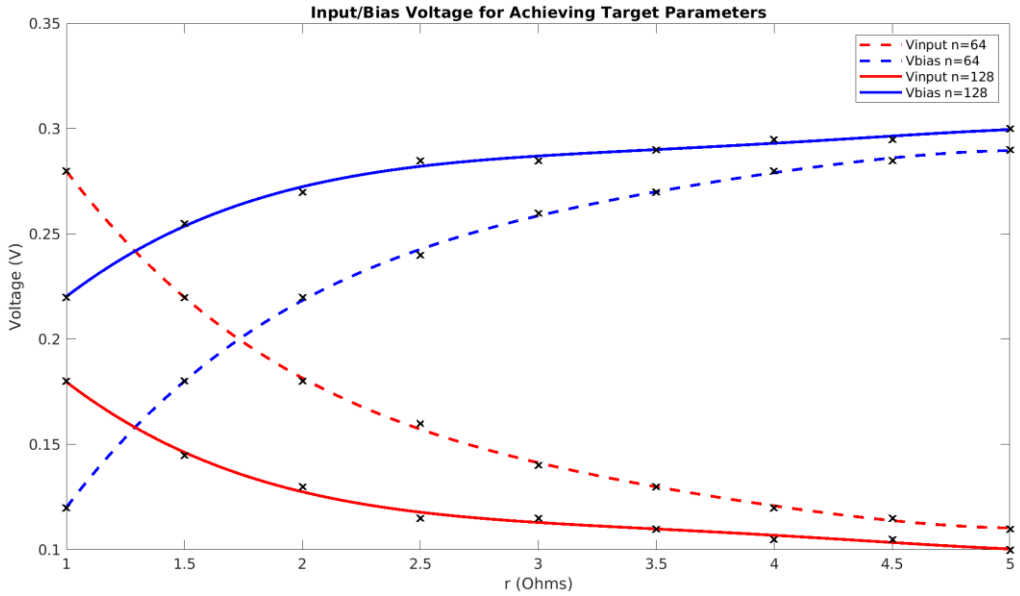


Figure 3.4: Input and bias voltage necessary to maintain constant measurement matrix parameters for line resistance values in the range from 1Ω per cell to 5Ω per cell, for a 64×64 and 128×128 array.

decreases for larger line resistance values as well as larger array size to counteract a higher rate of voltage degradation by reducing the current in the array. A higher bias voltage is then required to compensate for the reduced input voltage.

Figure 3.5a and Figure 3.5b show results for a 64×64 MCA and 128×128 MCA, respectively, with on-state and off-state devices represented in yellow and blue, respectively. As expected, a sharp reduction in densities is observed in both panels due to a change in multiplexer configuration, yielding a reduced input voltage to the SBGs.

Table 3.3 lists the delay for configuration of a 64×64 and 128×128 array. In this simulation, a parasitic resistance of 2.5Ω is modeled as a series combination with the capacitor. A clock period of 1.6ns is adequate for each capacitor to attain 99.9% of the target voltage. Figure 3.6 provides timing diagrams to illustrate the transient response of the SBGs. The top two panels in Figure 3.6

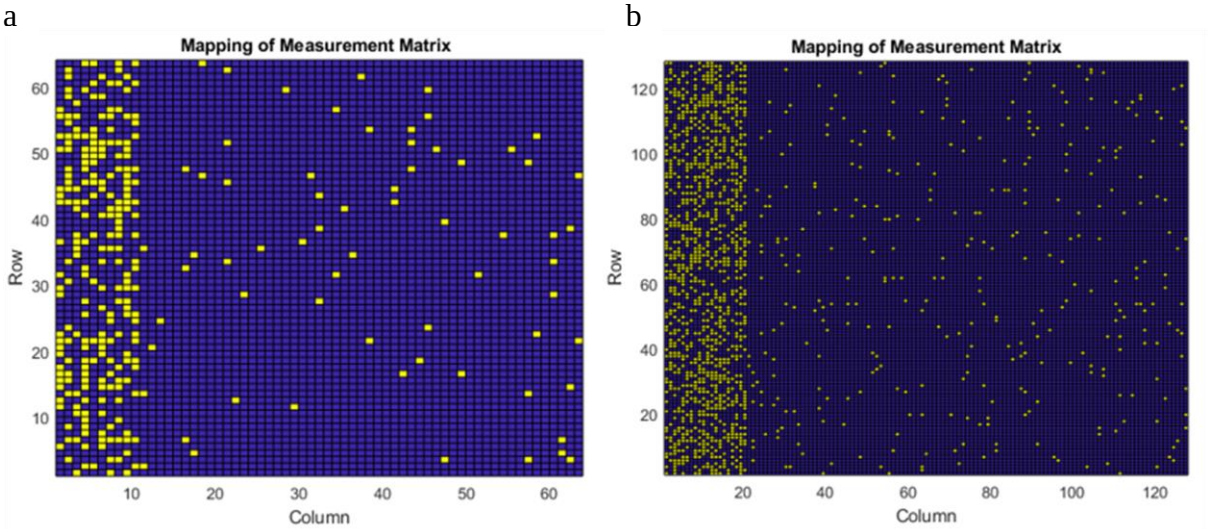


Figure 3.5: CS measurement matrix mapped to MCA crossbar array for a) a 64×64 and b) a 128×128 array size, demonstrating achievement of target parameters given in Table 3.1. Yellow and blue cells represent on-state and off-state devices, respectively.

show the inputs to the transmission gate within the SBG, indicating that the transmission gate is activated at t = 1.0ns. The next panel shows the transient voltage on the SBG capacitor within Column 1, which is inside the RoI and charged to a target voltage of around 0.39V. Finally, the bottom panel shows the transient voltage on the SBG capacitor within Column 128, which is outside of the RoI and hence charged to a reduced voltage of approximately 0.32V. These data demonstrate the completion of the control cycle within the 1.6ns period listed in Table 3.3.

Table 3.3 also gives energy results, giving the counterintuitive result that the maximum configuration energy of 333fJ occurs in the case of the smaller 64×64 MCA. Further analysis on this result is provided in Section 3.4. Table 3.4 provides a comparison between the presented approach and the alternative of using a 4-bit lookup table (LUT) together with Digital to Analog Converter (DAC) for acquiring the configuration data. Based on prior work [107], a 6-input LUT consumes 8.58fJ of read energy per bit, together with 1,547 transistors. Thus, supplying 4 bits to each column of a 64×64 array would require 256 LUTs, each being read simultaneously; the cost

Table 3.1: Simulation parameters for a crossbar representing two sub-matrices, including measurement matrix parameters n, m, L, α_1, p_1; MTJ P- state resistance, R; line resistance, r; capacitance, C; initial word line input voltage, V_{input}; and bias voltage, V_{bias}.

	Case 1	**Case 2**
$n \times m$	64×64	128×128
L/n	0.1	0.1
α_1	0.15	0.15
p_1	5/64	5/128
R	5600Ω	5600Ω
r	2.5Ω/cell	2.5Ω/cell
C	20fF	20fF
V_{input}	0.16V	0.12V
V_{bias}	0.24V	0.28V

Table 3.2: Parameters of the three-terminal MTJ device.

Parameters	**Value**
MTJ area	60nm×30nm×π/4
Heavy metal volume	100nm×60nm×3nm
Oxide thickness	0.85nm
P-state resistance	5600Ω

Table 3.3: Simulation results for writing a CS measurement matrix with RoI.

Array size	**Power**	**Time**	**Total energy**
64×64	208μW	1.6ns	333fJ
128×128	169μW	1.6ns	270fJ

Table 3.4: Comparison of our presented architecture with the alternative of using a 4-bit lookup table and digital-to-analog converter for signal acquisition in a 64×64 array.

Architecture	**Energy**	**Transistor count**
Herein	333fJ	17,088
LUT+DAC [28], [84]	194pJ	>396,032

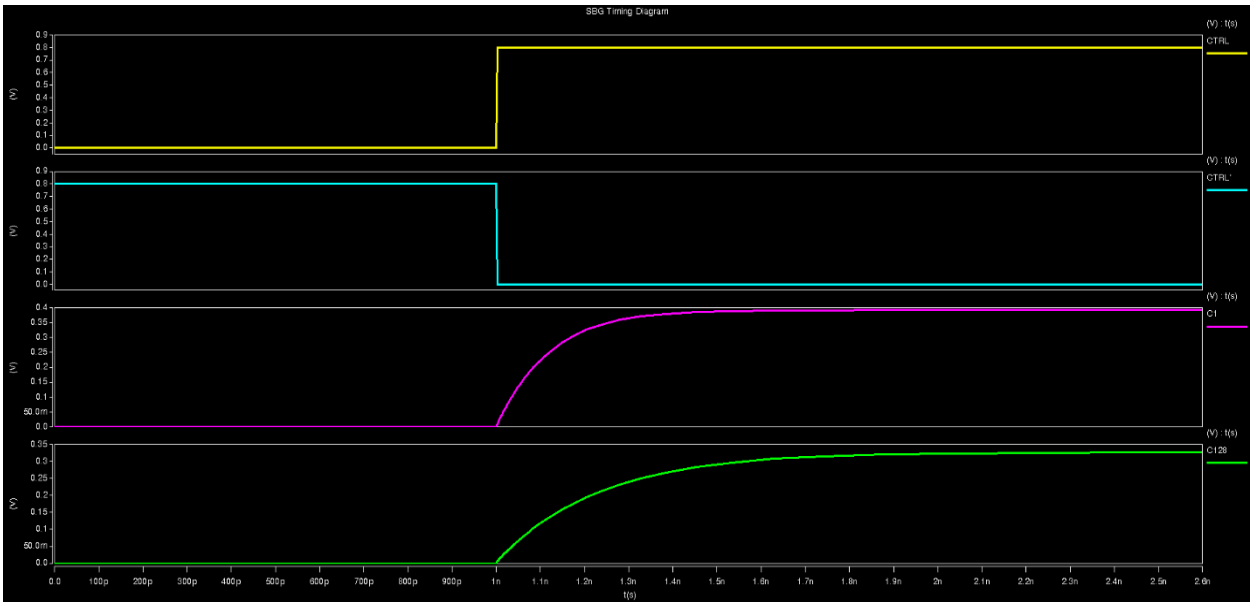


Figure 3.6: Timing diagram showing (from top to bottom): CTRL signal, $\overline{\text{CTRL}}$ signal, voltage on SBG capacitor in 1st column, and voltage on SBG capacitor in 128th column. Data shown is for a 128×128 array with 0.12V applied along the uppermost word line, and a bias voltage of 0.28V.

would be 2,196fJ of energy in addition to 396,032 transistors. Moreover, a 4-bit DAC consumes 3pJ per conversion [130]; the total energy would thus be 192pJ if one DAC were included for each column of the array. This would bring the total overhead to 194pJ plus 396,032 transistors. In contrast, our design consumes 333fJ in total energy for charging the capacitors in each of the 64 SBGs to the correct input voltage. Moreover, the hardware overhead is limited to 64, 6-input multiplexers; at a cost of 267 transistors per multiplexer, this brings the total number of transistors to 17,088. Thus, significant reductions in hardware and energy costs are attained by relying on the array itself to store and supply the necessary configuration signals. Due to the simplicity of our design, we obtain a 583-fold reduction in energy together with a 23-fold reduction in hardware resources.

3.4 Analysis of Size Dependence of Energy Consumption

Table 3.3 gives the unexpected result that greater energy consumption occurs for a smaller crossbar array. This section gives a theoretical analysis of this result. First, we note the following observations:

1) Our design uses a control cycle to set the capacitor voltage of each SBG, where SBGs serving columns part of the Region of Interest require a higher voltage. Since SBGs acquire voltages only from the uppermost word line in the array, only this specific word line receives nonzero input voltage.
2) The 128×128 array receives a lower input voltage than the 64×64 array. Specifically, an input of 0.12V is applied to the 128×128 array while an input of 0.16V is applied to the 64×64 array, as indicated in Table 3.1. This is necessary since both a higher input voltage, and larger crossbar size, result in a greater rate of voltage degradation. Thus, a reduced input voltage is applied to the larger array in order to compensate for the array size, as illustrated in Figure 3.4.
3) The 128×128 array experiences continual degradation in voltage past the 64^{th} column, which can lead to a reduced average top-row voltage compared to the 64×64 array.

For simplicity, we consider the total power consumption of a 64×64 and 128×128 array under steady-state conditions, i.e., treating capacitors as open circuits. The MRAM device resistance within the array is assumed to be $R = 5600\Omega$ with a line resistance of $r = 2.5\ \Omega$ per cell, which are the same conditions used in previous simulations. Since voltage is applied only to the uppermost word line, roughly 98.6% of the power consumption occurs along the top word line and the devices connected to this word line. Hence, the total power consumption can be approximated by:

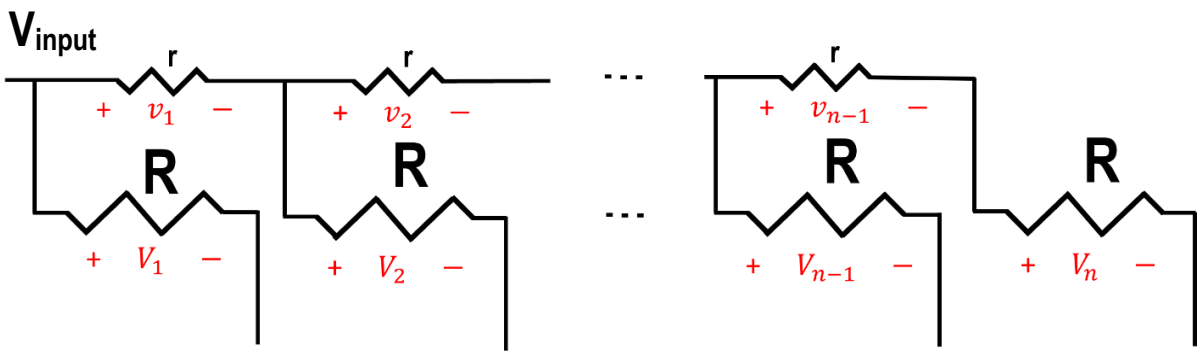


Figure 3.7: Model of top row of an n×n crossbar array, with parasitic resistance along the top word line labeled as r, and memristive devices labeled as R.

$$P = \frac{1}{R}\sum_{i=1}^{n} V_i^2 + \frac{1}{r}\sum_{i=1}^{n-1} v_i^2 \tag{3.2}$$

where V_i refers to the voltage difference across the devices connected to the top word line, v_i refers to the voltage difference across the parasitic resistors in the top word line, and n refers to the number of bit lines in the array, as illustrated in Figure 3.7. Eq. 3.2 can also be written as:

$$P = \frac{n}{R}\overline{V^2} + \frac{(n-1)}{r}\overline{v^2} \tag{3.3}$$

where bars are used to denote average values. Table 3.5 lists the values of $\overline{V^2}$ and $\overline{v^2}$ for the 64×64 and 128×128 in steady-state conditions. Table 3.5 shows that $\overline{v^2}$, which is a reflection of the average rate of voltage degradation along the uppermost word line, is reduced by 52% for the 128×128 array. This is largely due to a continual degradation in voltage past the 64th column, at a reduced rate. $\overline{V^2}$ for the 128×128 array is reduced by 76% compared to the 64×64 array, which makes sense in light of the second and third observations listed above. By substituting the data in

Table 3.5: Mean square voltage values for 64×64 and 128×128 arrays. Units are V^2.

	$\overline{v^2}$	$\overline{V^2}$
64×64	2.30×10^{-6}	0.0112
128×128	1.10×10^{-6}	0.00273

Table 3.5 into Eq. 3.3, we obtain P = 186μW and P = 118μW for the smaller and larger arrays, respectively. These values are within 2% of the total steady-state power consumption for the arrays and less than the transient average power data listed in Table 3.3. A similar albeit more complex analysis could be performed to explain the differences in the transient data.

In short, the power consumption for the 128×128 array is lower due to a reduced input voltage and greater voltage degradation beyond the 64th column: a quadratic relationship between power and voltage counteracts a linear increase in row size.

3.5 Summary

It has been shown that parasitic voltage degradation in a crossbar array can be used to implement control logic for applications in non-uniform CS, whereby a range of input signals is intrinsically derived from a single bias voltage source. Target inputs are generated within a single clock cycle at a maximum energy overhead of 333fJ. Moreover, the results are shown to be robust to array size and magnitude of parasitic resistance. While the primary focus has been CS, there are a myriad of additional applications requiring non-uniform voltage that could benefit from this approach.

CHAPTER 4: AREA-EFFICIENT IMAGE COMPRESSION VIA MEMRISTIVE CROSSBARS LEVERAGING ADAPTIVE QUANTIZATION

4.1 Crossbar Memory Allocation via Adaptive Quantization

MCAs have been employed for image compression via DCT as well as CS. Li *et al.* [131] demonstrated 2D DCT on a fabricated 64×128 array. The authors were able to represent each matrix element to 6-bit precision via 64 levels of conductance in the memristors being used. Two memristors were used to represent a single matrix element in order to accommodate both positive and negative elements without having access to negative conductance values. Zhang [28] demonstrated an optimized CAD approach to DCT using memristive crossbars, where the computation was restructured to include only a single VMM operation. In addition to DCT, crossbar approaches to image compression by CS sampling have been demonstrated by Le Gallo [122] and Salehi [20], as discussed in Section 2.4.5.

Due to the sparsity of images in the DCT domain, Adaptive Quantization (AQ) is a useful memory allocation strategy for image processing applications. In the context of the research presented herein, AQ consists of mapping data to memory at variable levels of precision, i.e., assigning a greater number of bits to certain subsets of data than others. While the benefits of AQ have been previously demonstrated [30, 31], its implementation within a crossbar memory array is challenging since conventional crossbar design approaches [84-86] do not readily allow for mixed-precision elements within a single array. Previous MCA-based AQ techniques, including reconfiguration of ADC bit-widths [28] and power gating parts of the array [91], reduce computational energy costs but are still expensive in terms of memory area. To the best of the author's knowledge, there have not been any designs published thus far leveraging adaptive precision levels for crossbar memory optimization.

4.2 AQ for Area-Optimized Image Compression

Herein we propose the Area-Conserving Crossbar Leveraging Adaptive Information Mapping (ACCLAIM). The objective of this architecture is to minimize the hardware resources necessary to perform image compression without compromising reconstruction accuracy; this objective is achieved through an AQ approach involving representation of matrix elements at variable precision levels based on relative importance levels of corresponding input coefficients. In contrast to the previous approach which performs non-uniform compression using variable sampling rates [132], AQ allows for reduction in circuit area in addition to energy consumption. AQ is especially effective when working with spectrally-sparse images in the DCT domain.

ACCLAIM is similar to previous crossbar designs [84, 85] which use multiple memristive devices across adjacent bit lines to represent a single matrix element. In these works, bit line outputs are combined using shift and add operations to yield final dot product results. ACCLAIM represents multi-bit elements using memristive devices spread across word lines in the array, which allows for AQ by varying the number of word lines assigned to each input coefficient.

Figure 4.1 illustrates the ACCLAIM architecture. The crossbar is shown with word lines running vertically and bit lines running horizontally. Inputs are delivered via sequential voltage pulses such that only a part of the array is active at any given time; memristive devices active during each cycle are indicated in the figure using a grayscale coding. The analog input vector, $\mathbf{x}$, is passed to word lines of a crossbar array. In the example shown, $\mathbf{x}$ consists of four coefficients and the array contains a total of 8 word lines: 4 word lines are allocated to coefficient x_0, 2 to x_1, and 1 each to x_2 and x_3. The tradeoff to this approach is that dot products must be performed sequentially rather than in parallel since precision levels are mapped to word lines rather than bit lines. In the example shown, inputs are delivered in sequential voltage pulses such that dot products

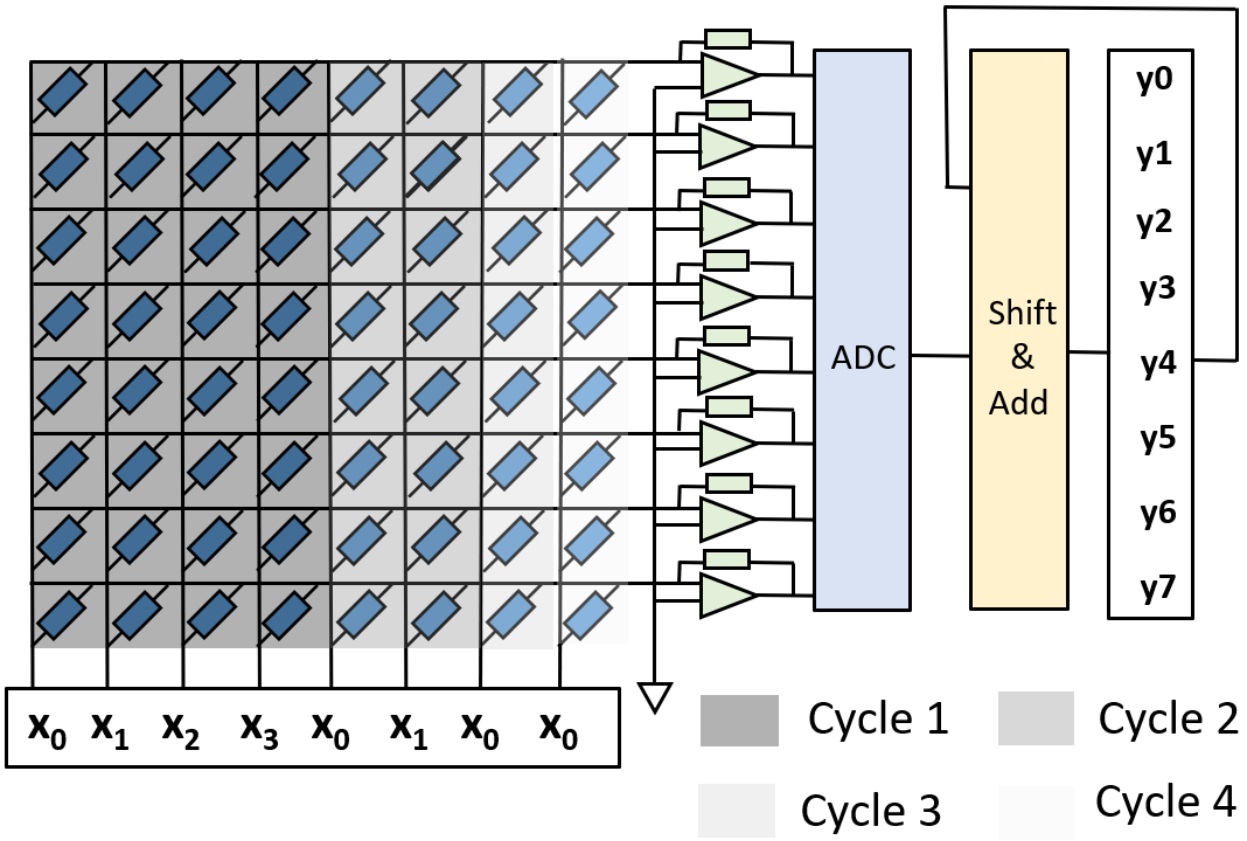


Figure 4.1: ACCLAIM architecture, including transimedance amplifiers shown in green, ADCs shown in blue and Shift and Add units shown in yellow.

are computed in four cycles; in Cycle k, only devices representing the k^{th} most significant bit of their respective matrix element are active. During each cycle, each intermediate dot product is converted to digital, and progressively combined with the previous cycle's result via shift and add. Thus, ACCLAIM does not require duplication of peripheral circuits, regardless of computational precision level.

4.3 Application to DCT

Adaptive Quantization via ACCLAIM allows for enhanced DCT transmission and reconstruction, given a specified bit budget. IoT sensors can significantly reduce image transmission overheads by applying row-wise quantization to the frequency-domain image, **X**,

prior to transmission. The resulting matrix, **X***, is used by the receiver to reconstruct the original image, **I**, using the inverse DCT transform: $\hat{\mathbf{I}} = (\mathbf{D}^T)(\mathbf{X}^*)(\mathbf{D})$. The first matrix product, $(\mathbf{D}^T)(\mathbf{X}^*)$, is performed row-by-row by providing subsequent rows of $\mathbf{D}^T$ as vector inputs to the mixed-precision matrix **X***; each operation serves as a mixed-precision vector-matrix multiplication which is optimally performed by the receiver using ACCLAIM. Products involving the DCT coefficient may result in an overflow, which is corrected through a scalar addition performed by the Shift and Add unit.

For purposes of evaluation, a 400×400 monochrome *Lena* image is partitioned into 8×8 blocks, and DCT is applied to each block. Row-wise quantization is performed based on a bit budget, *B*, which represents the number of bits available to each column of **X***. Two methods of quantization are compared: AQ, which intelligently allocates the available bit budget across rows of **X*** based on a gradient descent optimization approach, and uniform quantization, which assigns a constant number of bits to each row regardless of importance level. Each AQ configuration is specified by a vector, $\boldsymbol{b}$, of 8 integers, b_i, representing the number of bits allocated to each element of row *i*. The objective is to minimize reconstruction error, defined as:

$$error = \left\|\hat{\mathbf{I}} - \mathbf{I}\right\| / \|\mathbf{I}\|, \tag{4.1}$$

under the constraint $\sum b_i \leq B$.

Simulations are conducted in MATLAB to assess reconstruction accuracies using both adaptive and uniform image quantization. Table 4.1 lists results for a variety of bit budgets, showing consistent and significant improvement as a result of AQ. Figure 4.2 shows the reconstructed *Lena* image using both uniform and adaptive quantization, with a fixed bit budget

a  b 

Figure 4.2: DCT reconstruction of compressed *Lena* image attained using 24 bits per column allocated a) uniformly, and b) adaptively among rows of 8×8 blocks in the frequency domain.

Table 4.1: Impact of AQ on DCT Reconstruction.

B	**Non-AQ Error (dB)**	**AQ Error (dB)**	**AQ Configuration**
12	-7.2	-30.2	[5,1,1,1,1,1,1,1]
16	-12.2	-36.0	[6,4,1,1,1,1,1,1]
20	-12.2	-40.8	[7,5,3,1,1,1,1,1]
24	-24.8	-43.0	[8,5,4,3,1,1,1,1]
28	-24.8	-45.0	[8,6,5,5,1,1,1,1]

of $B = 24$, to illustrate the 18dB improvement resulting from AQ. For reference, uniform quantization requires $B = 56$ to attain the same reconstruction accuracy.

4.4 Application to CS

ACCLAIM is next evaluated for CS sampling and reconstruction. Following the procedure outlined in Figure 1, the *Lena* image is partitioned into 10×10 blocks, which are then transformed via DCT and compressed via CS sampling; the image is subsequently reconstructed using the Approximate Message Passing (AMP) algorithm [122] and inverse DCT. For evaluation purposes,

a  b 

Figure 4.3: CS reconstruction of compressed *Lena* image partitioned into 10×10 blocks, each sampled using a 40×100 measurement matrix with 200 bits per row allocated a) uniformly and b) adaptively.

Table 4.2: Impact of AQ on CS Reconstruction.

B	***m***	**Non-AQ Error (dB)**	**AQ Error (dB)**	**AQ Configuration**
120	40	-15.2	-37.8	[3,1,3]
200	40	-31.3	-40.3	[5,1,5]
280	40	-31.3	-40.0	[6,1,6]
360	40	-35.2	-40.7	[8,1,6]
120	60	-10.3	-39.1	[3,1,3]
200	60	-31.8	-43.1	[5,1,5]
280	60	-31.8	-43.5	[6,1,6]
360	60	-39.1	-43.5	[6,1,7]

the measurement matrix, $\boldsymbol{A} \in \mathbb{R}^{m\times 100}$, is quantized based on the memory budget, *B*, representing the available number of bits per row. AQ allocates a greater number of bits to critical columns of ***A***, corresponding to more significant DCT coefficients. AQ configurations for CS are represented by a vector, ***b,*** consisting of 3 integer elements: b_1 is the number of bits assigned to elements of critical columns within the measurement matrix, b_2 is the number of bits assigned to elements of

non-critical columns, and the square of b_3 is the number of columns considered critical; the number of critical columns is chosen as the square of an integer due to the structure of the DCT matrix. AQ seeks to minimize reconstruction error under the constraint $b_1 b_3^2 + b_2(100 - b_3^2) \leq B$.

Simulations are conducted in MATLAB to compare CS reconstruction accuracy between adaptive and uniform quantization as a function of bit budget, *B*, and number of measurements, *m*. Results listed in Table 4.2 demonstrate consistent improvement resulting from AQ in the case of m = 40 as well as m = 60. Figure 4.3 shows the reconstructed *Lena* image sampled using 40 measurements and a bit budget of 200 bits per row, illustrating the 9dB improvement resulting from AQ.

In addition, hardware simulations in HSPICE are performed to assess per-block energy and area requirements of ACCLAIM to achieve a set reconstruction error under CS sampling using 40 measurements. Energy is computed using the formula $E = \sum_i P_i t_s$ where P_i is average power per cycle and t_s=100ns is the sampling time for one cycle. Simulations are performed on MRAM crossbar arrays using SHE-MTJ technology [20] with parameters given in Table 4.3; the CS measurement matrix is simulated by choosing random states for MRAM devices, and input voltages represent a selection from the *Lena* image in the frequency domain. The size of the crossbar is B×40, where B is the bit budget necessary to achieve a set reconstruction accuracy.

Hardware simulation results, shown in Figure 4.4, confirm the consistent area benefits seen in Table 4.3. For a minimum error of -35dB, AQ reduces the memory word line count from 400 to 118, thus achieving a 70.5% benefit in area. Moreover, AQ achieves reconstruction errors below -40dB, which is never achieved by uniform quantization regardless of array size. This result is consistent with prior works demonstrating outperformance of non-uniform CS [20, 23]. Energy consumption is reduced for the AQ approach in all but two cases; the reason for the higher energy

Table 4.3: Hardware Simulation Parameters.

Tech Node	14nm
V_{DD}	0.8V
Parallel MTJ Resistance (logic 0)	2800Ω
Anti-parallel MTJ Resistance (logic 1)	5600Ω
MTJ Polarization	0.52
MTJ Area	60nm×30nm×π/4

despite reduced area is that the average voltage input to the array is higher in the case of AQ since more significant components of the input are duplicated as per Figure 4.1. Energy consumption for the array is reduced by 30.2% at an error threshold of -35dB.

Herein, we define the *Error-Energy-Area Product (EEAP)* metric as:

$$EEAP = E_s \times A_{CB} \times error \tag{4.2}$$

where E_s is sampling energy per block, A_{CB} is crossbar memory area per block and $error$ refers to the image reconstruction error as defined in Eq. 4.1. EEAP is a metric which measures the efficacy of resource utilization in approximate computing, considering the tradeoff between hardware costs and computational accuracy. As such, EEAP is a convenient metric for assessing the efficacy of image compression hardware in resource-limited applications such as IoT. ACCLAIM achieves a 79.4% reduction in EEAP compared to the conventional uniform quantization approach.

Finally, simulations are performed to assess the latency of a 164×40 array operating over 5 cycles, which is necessary to achieve a -40dB error using AQ. Results indicate a crossbar latency within 100s per cycle, and latency of peripherals within 7ns. Thus, 40, 5-bit measurements are produced within 535ns. This latency is equivalent to the delay of transmitting the data at 373Mbps, and hence not a bottleneck due to the limited bandwidths of IoT sensors.

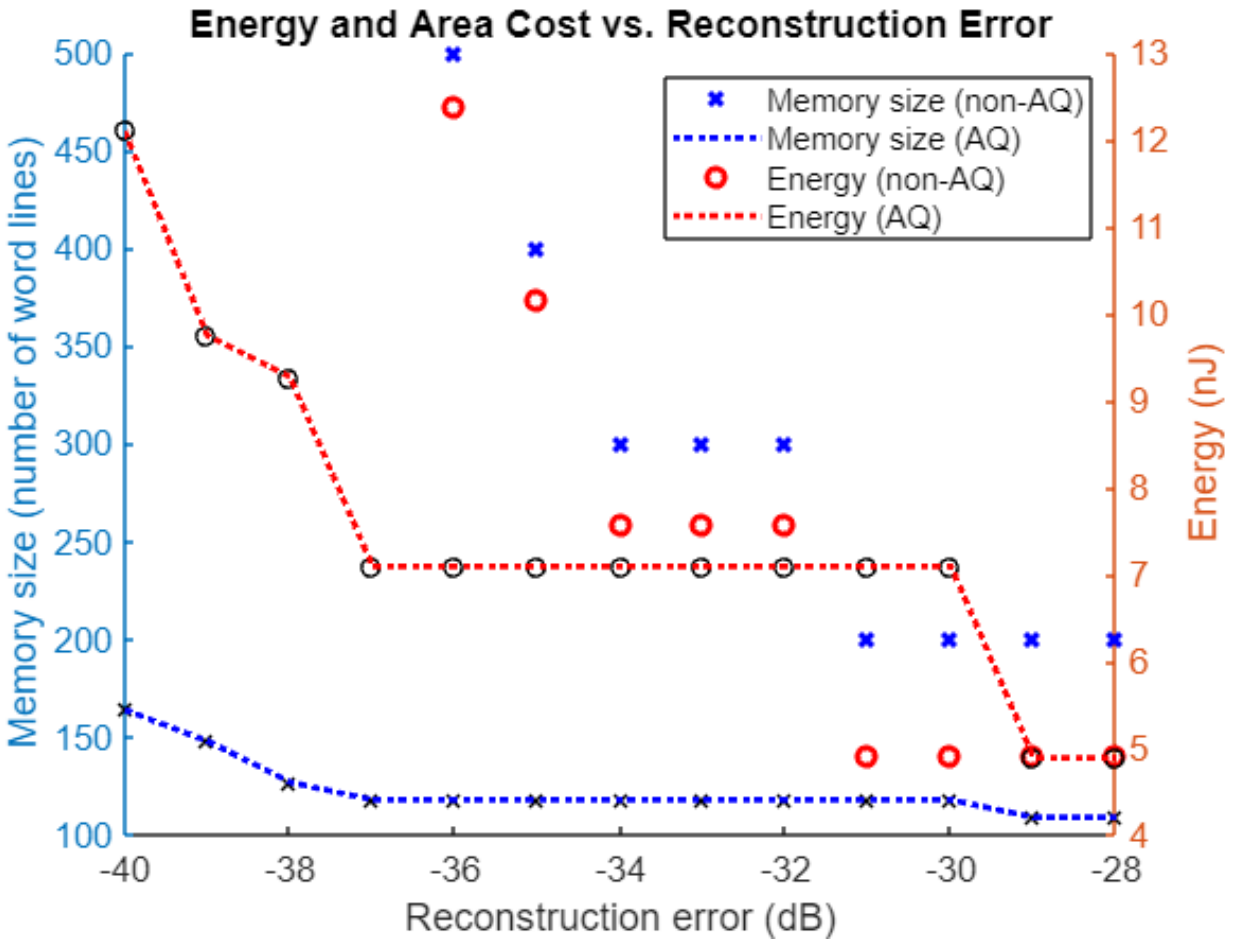


Figure 4.4: Sampling energy and area per block necessary to achieve a set CS reconstruction accuracy for the *Lena* image, partitioned into 10×10 blocks and sampled using a 40×100 matrix.

4.5 Summary

Herein, we have developed the Area Conserving Crossbar Leveraging Adaptive Information Mapping (ACCLAIM), a novel memristive crossbar design which intelligently allocates crossbar memory by assigning greater precision levels to matrix elements corresponding to more significant subsets of the input space. Such an Adaptive Quantization (AQ) technique is particularly useful for image compression applications such as Discrete Cosine Transform (DCT) and Compressive Sensing (CS), where the input often consists of sparse signals with specific regions of interest.

ACCLAIM reduces storage and data transmission overheads by reducing the size of the image, in the case of DCT, and the size of the measurement matrix, in the case of CS, without compromising reconstruction accuracy. Moreover, given a fixed accuracy standard, ACCLAIM allows for a 70.5% reduction in area and 30.2% reduction in energy. We define the Energy-Error-Area Product (EEAP) as a useful metric for expressing the efficacy of resource utilization in approximate computing applications. AQ implemented via the presented architecture achieves a 79.4% reduction in EEAP for CS sampling computations.

CHAPTER 5: EXPONENTIATION USING STT MAGNETIC TUNNEL JUNCTIONS[4]

5.1 Analog Circuit Design

5.1.1 Op-Amp Design

The proposed reconfigurable analog multiplier is based on the op-amp design presented in Figure 5.1. The op-amp consists of two cascaded stages: an input stage consisting of a differential amplifier, followed by a gain stage. A simple design consisting of only 10 CMOS transistors is chosen to optimize for power consumption as well as area. The op-amp is simulated using models from the PTM 14nm LSTP library [127], at V_{DD} = 0.8V.

Figure 5.2a then presents a layout of the proposed op-amp design. The layout indicates dimensions of 43F×23F, for a total area of 989F^2. This layout is contrasted with a CMOS NAND gate in Figure 5.2b, which has dimensions of 18F×14.5F, for a total area of 261F^2. The op-amp and NAND gate form an interesting comparison as common building blocks of analog and digital multipliers, respectively.

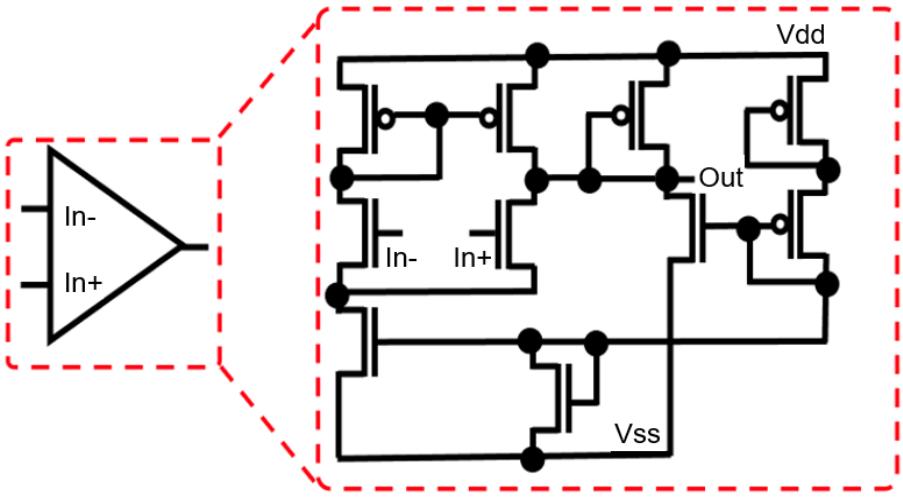


Figure 5.1: Op-amp comprised of 10 MOSFETs offering high speed and compact area.

[4] ©IEEE. Part of this chapter is reprinted, with permission, from [136, 137].

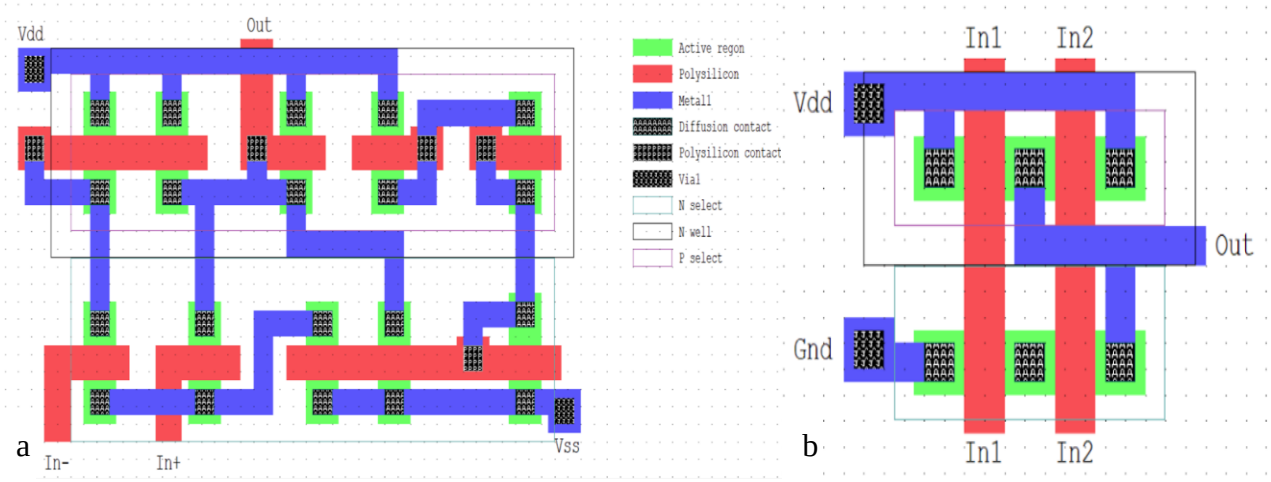


Figure 5.2: a) Layout of op-amp used in this dissertation versus b) a CMOS NAND gate.

5.1.2 Three-Stage Analog Circuit

Similarly to [98], the translinear principle is applied to attain exponentiation of the input signal. We propose a reconfigurable design, embedded within the FPAA fabric shown in Figure 5.3. The design consists of a three-stage circuit, shown in Figure 5.4, which accepts a single input for performing exponentiation operations; the design can also be reconfigured to accept two inputs for performing analog multiplication.

The first stage, outlined in red in Figure 5.4, is a logarithmic amplifier with output given by:

$$V_1 = -A_{OL}V_0 \tag{5.1}$$

$$-\frac{V_0 - V_{in}}{R_1} = I_{S1}\left[\exp\left(\frac{V_0 - V_1}{V_T}\right) - 1\right] \tag{5.2}$$

where A_{OL} represents open-loop gain and I_{S1} represents the saturation current of diode D_1. Eq. 5.1 is from general op-amp theory and Eq. 5.2 results from applying Kirchhoff's Current Law at the negative input terminal. Thus, solving Eq. 5.1 and Eq. 5.2 simultaneously yields:

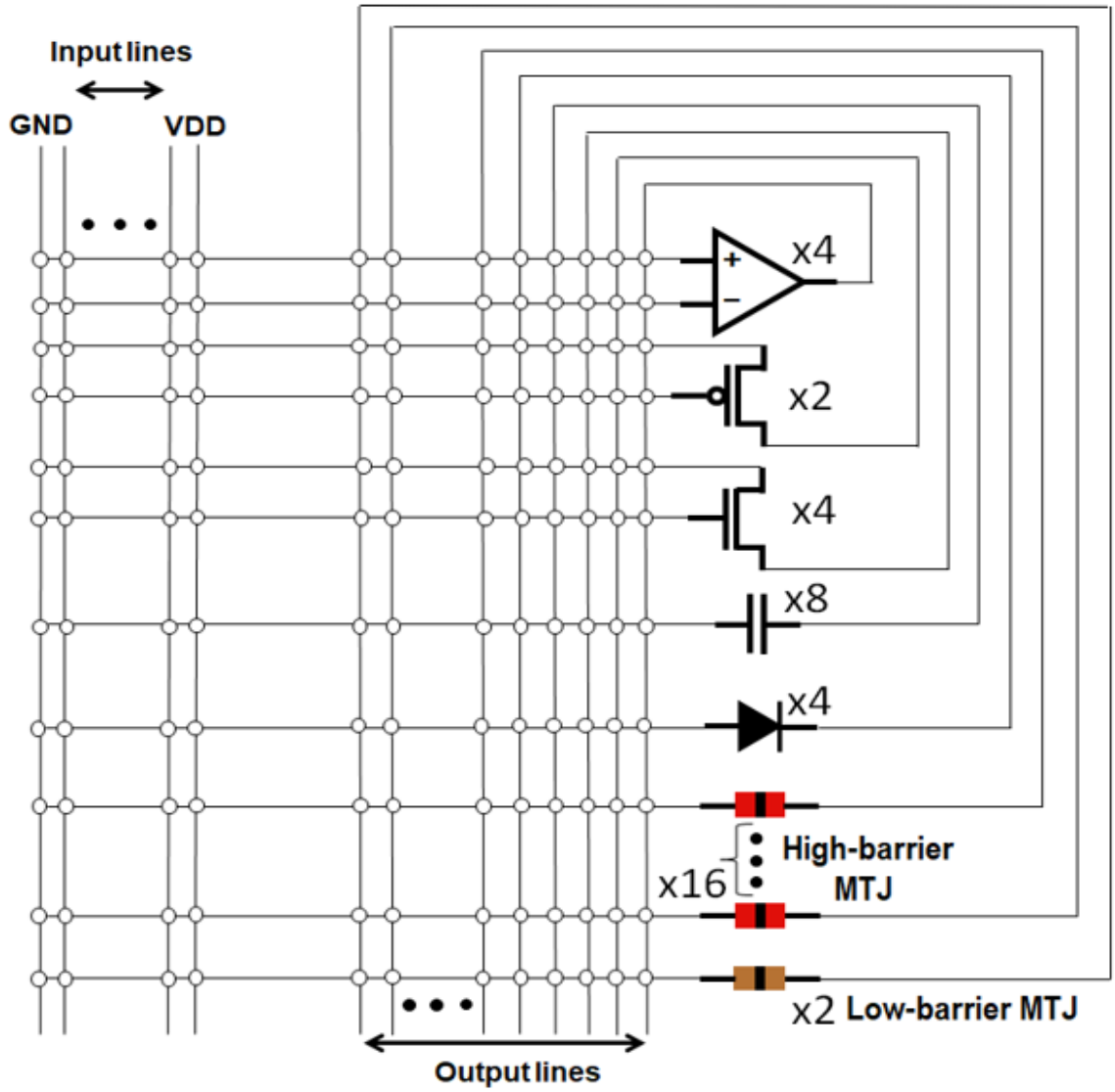


Figure 5.3: FPAA fabric comprised of active and passive analog devices such as NMOS/PMOS transistors, capacitors and diodes, along with spin-based Magnetic Tunnel Junction (MTJ) devices.

$$V_1\left(1+\frac{1}{A_{OL}}\right) = -V_T ln\left(\frac{V_{in}+\frac{V_1}{A_{OL}}}{R_1 I_{s1}}+1\right) \tag{5.3}$$

In the limit of infinite open-loop gain and high input voltage, Eq. 5.3 is approximated as:

$$V_1 = -V_T ln\left(\frac{V_{in}}{R_1 I_{s1}}\right). \tag{5.4}$$

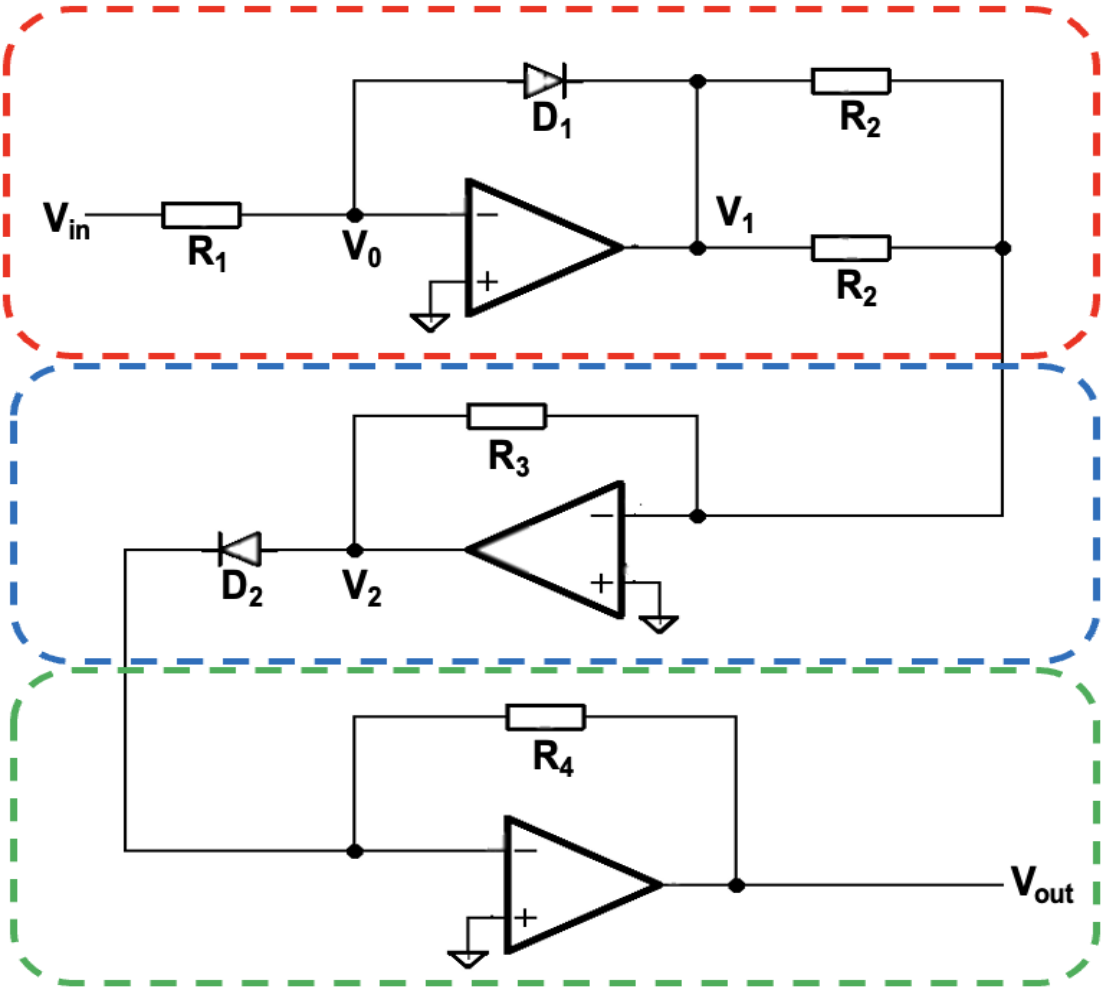


Figure 5.4: Analog circuit for generalized exponentiation. The first, second, and third stages are outlined in red, blue, and green, respectively.

The second stage is an analog adder, whereby a similar analysis yields $V_2 = -\frac{2V_1R_3}{R_2}$. Finally, the third stage is an anti-log amplifier with output approximately given by:

$$V_{out} = -R_4 I_{s2} e^{\frac{V_2}{V_T}}. \tag{5.5}$$

Overall, it is seen that the output of this circuit is given by:

$$V_{out} = -\frac{R_4 I_{s2}}{(R_1 I_{s1})^a} (V_{in})^a \tag{5.6}$$

where $a = 2R_3/R_2$.

The above analysis indicates the ability to implement any positive power function of the input via the design shown in Figure 5.4. In addition, a dual-input stage consisting of two logarithmic amplifiers can be inserted to attain an analog multiplier. Finally, an inverting amplifier can be inserted between the second and third stages to realize inverse power functions as well. Each mode can be implemented using the elements included in the fabric presented in Figure 5.3. To minimize area, MTJs in the P state are used to implement the resistors shown in Figure 5.4; MTJs in the P state have roughly linear I-V characteristics in accordance with experimental data [133].

Eq. 5.4 – 5.6 hold precisely only for infinite open-loop gain, which is not attainable in practice. Thus, the equations provide a starting point for the design, after which parameters must be adjusted to minimize output errors. Final parameters are: $R_1 = 3500k\Omega$, $R_2 = 50k\Omega$, $R_3 = 150k\Omega$, $R_4 = 75k\Omega$, $I_{s1} = 50nA$, and $I_{s2} = 5.4nA$. In addition, a load capacitance of 100fF and load resistance of $1000k\Omega$ are included at the output stage of each op-amp.

5.2 Analog Multiplication

First, the performance of the analog circuit is evaluated as a multiplier. In this mode, two separate logarithmic amplifiers serve as the input stage, receiving inputs V_{in1} and V_{in2}. The circuit is evaluated in terms of DC transfer characteristics, frequency response, and Total Harmonic Distortion (THD), for various DC amplitudes of V_{in2} within the operational range between 0.3V and 0.7V.

DC transfer characteristics are presented in Figure 5.5. In each trial, V_{in1} is swept across the operational range and the average non-linearity error is determined based on the percentage

deviation from a linear regression line. As listed in Table 5.1, the maximum non-linearity error of 0.55% occurs at $V_{in2} = 0.7\text{V}$.

Figure 5.6 shows the frequency response of the multiplier, evaluated in the range from 100MHz to 1GHz. In this case, V_{in1} is a sinusoidal signal with offset of 0.45V and amplitude of 0.25V and V_{in2} is fixed. The -3-dB bandwidth, listed in Table 5.1, is in the 100MHz range in each case. While this bandwidth may be high for applications with limited signal-to-noise ratio, the circuit can be reconfigured to attain various bandwidths depending on the RC time constant at the op-amp output.

Table 5.1 also lists the delay in reaching 90% of the target voltage in the case when $V_{in1} = V_{in2}$; the delays are in the nanosecond range, consistent with the circuit bandwidth.

Next, Table 5.2 provides THD in the case where one input is 0.45V DC, and the second input is sinusoidal with amplitudes of 0.05V and 0.25V. THD is within 1% up to a frequency of approximately 1MHz, indicating practical functionality of the system.

5.3 Generalized Exponentiation

5.3.1 Circuit Performance

The circuit is next evaluated in its ability to compute square and square root functions. Simulation results demonstrate high-accuracy implementation of n^{th}-root functions; power functions beyond squaring introduce challenges related to voltage saturation. It is, however, possible to obtain these functions via a squaring unit by iteratively applying the mathematical identity: $(A+B)^2 - (A-B)^2 = 4AB$. For example, in the case of the cubing function, x^2 is substituted for A and x for B; any n^{th} power function, $n \geq 2$, can thus be computed [95]. The authors of [95] were hence able to compute a 5^{th} order polynomial function within 10% error.

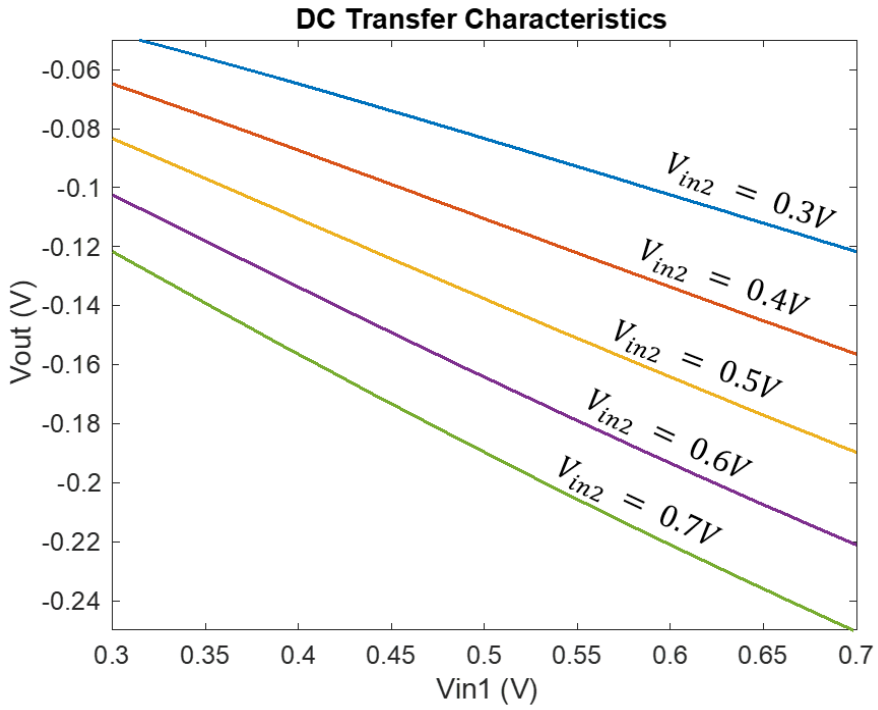


Figure 5.5: DC transfer characteristics for the proposed multiplier, with one input fixed and the second input varying across the operational range.

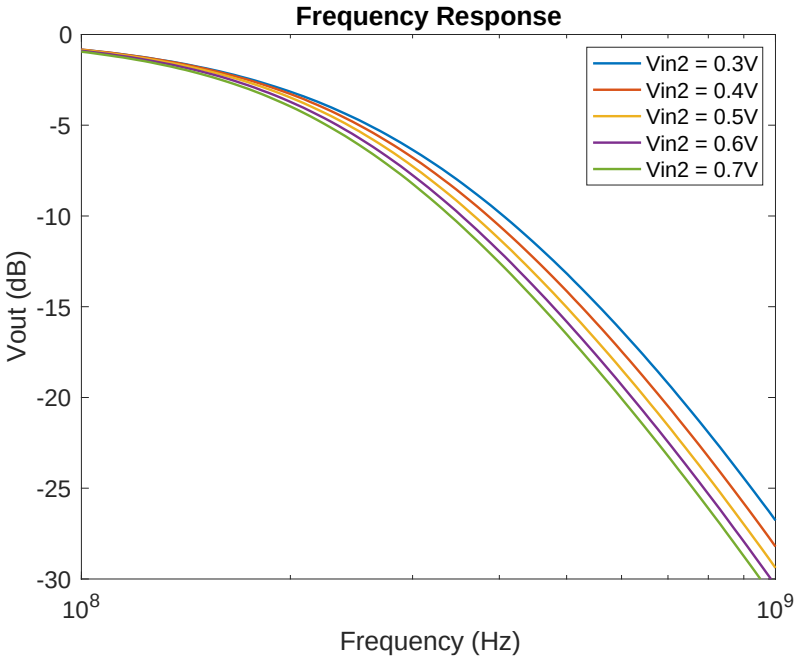


Figure 5.6: Frequency response, with one input fixed and the other input sinusoidal with offset of 0.45V and amplitude of 0.25V.

Table 5.1: Error, bandwidth, and delay data.

V_{in2}	Error	-3-dB bandwidth	Delay
0.3V	0.48%	195MHz	3.8ns
0.4V	0.11%	191MHz	3.9ns
0.5V	0.25%	186MHz	4.1ns
0.6V	0.43%	178MHz	4.4ns
0.7V	0.55%	174MHz	5.0ns

Table 5.2: THD with one DC and one sinusoidal input.

Frequency	THD@Amplitude = 0.25V	THD@Amplitude = 0.05V
10KHz	0.80%	0.76%
100KHz	0.81%	0.77%
1MHz	0.81%	0.75%
2MHz	1.08%	1.10%
3MHz	1.82%	1.61%

Performance of cube root, square root, and squaring circuits implemented using the proposed design are given in Table 5.3, including technology node, supply voltage, total number of elementary components, power dissipation, and mean error over an input range of 0.2V – 0.6V.

Comparing to the approximate digital multiplier described in [37], at the design point giving nearly identical power consumption, the analog circuit described herein yields slightly improved mean error across the operational range. Furthermore, the approximate digital design requires an area equivalent to 245 CMOS NAND gates, i.e., 980 CMOS transistors. Thus, our design achieves

Table 5.3: Comparison of area, power, and accuracy of STT-MTJ based generalized exponentiation with alternate recent approaches.

	Herein	**Herein**	**Herein**	**[37]**	**[39]**	**[95]**	**[98]**
Mode	Analog	Analog	Analog	Digital	Analog	Analog	Analog
Operation	Cube root	Sq. root	Square	Multiplier	Multiplier	Square	Square
Tech node	14nm	14nm	14nm	28nm	130nm	500nm	180nm
V_{DD}	0.8V	0.8V	0.8V	1V	0.6V	1.5V	1.3V
No. comp.	43	43	43	~1000	35	12	~100
Power	123μW	122μW	126μW	126μW	23μW	600μW	149μW
Mean Error	0.50%	0.66%	1.30%	1.87%	9.1%*	N/A	0.24%**

*RMS noise vs. max. output

**At $V_{in} = 0.4$V

a 97% reduction in transistor count. In addition, the layout presented in Figure 5.2 indicates that the area of an op-amp is approximately 3.79× the area of a NAND gate; this indicates an approximately 95% reduction in area if three op-amps are used for squaring.

The design in [39] demonstrates reduced power consumption but significantly higher error, and a relatively limited bandwidth of 51.2KHz. The approach developed in [98] introduces a similar design to the one described herein, relying on the translinear principle to implement n^{th} power functions by combining hardware with logarithmic and exponential output characteristics; a limitation of this design is that its reliance on time-mode circuitry intrinsically leads to significant time delays, on the order of microseconds.

5.3.2 Process Variation of MTJ Devices

A Monte Carlo simulation is performed to determine the effects of process variation in MTJ devices. For this simulation, 100 trials are conducted considering a 1.5% standard deviation in

Table 5.4: Error rate due to MTJ process variation.

V_{in}	Square	Square root
0.3V	5.96%	3.81%
0.4V	6.36%	3.88%
0.5V	6.13%	3.92%
0.6V	5.72%	3.95%
0.7V	5.30%	3.96%

the resistance of each MTJ device; this value is consistent with the variation seen in a 4-Mb MRAM array [134]. The resulting standard deviations in the circuit outputs are listed in Table 5.4. While the maximum standard deviation due to PV is 6.36%, the presence of only 16 high-barrier MTJ devices in the reconfigurable fabric may allow for improved device tolerances and thus improved computational accuracy in the fabricated design.

5.3.3 Variation in Diode Saturation Voltage

A further analysis is performed in this section to assess the impact of diode characteristics on DC transfer characteristics and frequency response of the presented design. Since inputs are delivered via voltages, the saturation current of the diodes is not a significant limitation on the input range. The diodes are necessary to regulate the voltage at each stage to attain an approximation for the desired computational function. According to the theory presented in Section 5.1.2, the performance of the circuit depends on the product $R_1 I_{s1}$, where R_1 is the resistance of the first-stage resistance and I_{s1} is the saturation current of the diode in the first stage. Thus, variation of I_{s1} should not make a difference if R_1 is adjusted to compensate. An analysis of DC transfer characteristics and frequency response is performed for the input combinations (R_1, I_{s1}) = (3500kW, 50nA), (35,000kW, 5nA) and (350,000kW, 0.5nA).

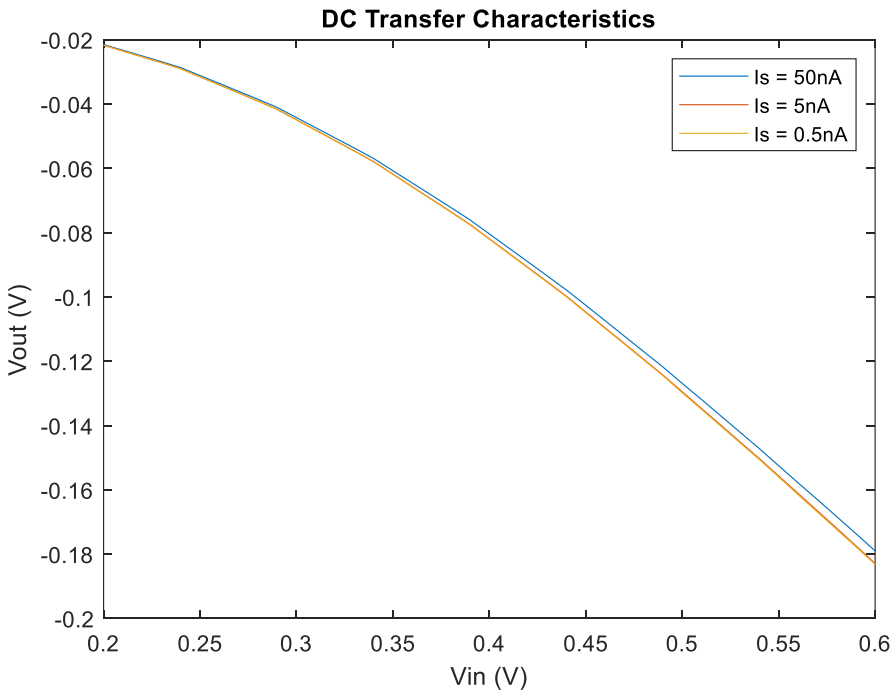


Figure 5.7: DC transfer characteristics for analog squaring circuit, considering three different parameters for the first-stage diode saturation current, Is.

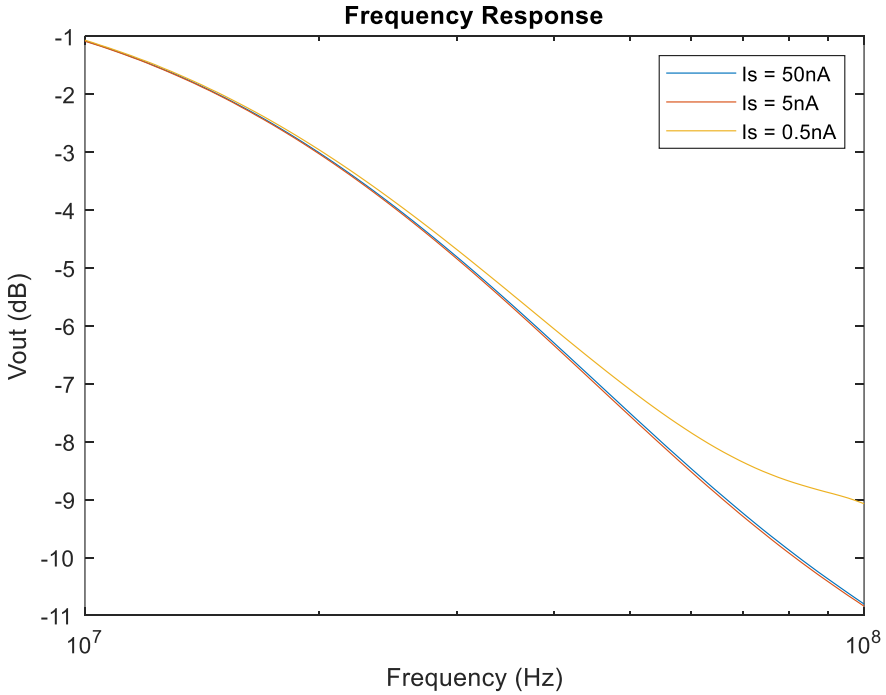


Figure 5.8: Frequency response for analog squaring circuit, considering three different parameters for the first-stage diode saturation current, Is, with an input voltage magnitude of 0.4V.

Table 5.5: Effect of diode saturation current on error and bandwidth of analog squaring circuit.

I_s	Error	-3-dB bandwidth
50nA	1.30%	19.50MHz
5nA	1.36%	19.50MHz
0.5nA	1.37%	19.95MHz

Figure 5.7 and Figure 5.8 show results of the analysis which are summarized in Table 5.5. The results demonstrate a negligible variation in error and bandwidth, in agreement with the theory. It may be noted that since both inputs are treated as sinusoidal, a smaller bandwidth is attained than that presented in Section 5.3.1, which treats only one input as sinusoidal. An average power consumption of 126μW is attained for all three cases.

5.3.4 Temperature Dependence

Given the temperature dependence of diodes as well as MTJ devices, a brief temperature analysis of the circuit performance is conducted to complement the previous data attained at a temperature of 25ºC. This analysis focuses on the temperature dependency of mean error of analog squaring using the presented design. While HSPICE includes temperature dependent models for diodes, MTJ temperature dependence is modeled using Eq. 2.11 – 2.13. The following substitutions may be made into Eq. 2.11 – 2.13:

$$P_0 = \sqrt{\frac{TMR_0}{2-TMR_0}} \tag{5.7}$$

$$G_{SI}(T) = ST^{4/3} \tag{5.8}$$

where Eq. 5.7 follows from Julliere's model (Eq. 2.8) and Eq. 5.8 is taken from [66].

In these equations, G_{SI} is a component of MTJ device conductance as defined in Eq. 2.11, S is a constant, T represents absolute temperature, P represents polarization and TMR represents tunneling magnetoresistance. The following parameters are substituted into the temperature model consisting of Eq. 2.11 – 2.13 together with Eq. 5.7 – 5.8: $TMR_0 = 1$, $\alpha = 2 \times 10^{-5}$ $K^{-3/2}$, C = 0.0015 K^{-1} and $S = 10^{-12}$ $\Omega^{-1} K^{-4/3}$ are used; G_0 is chosen based on the target MTJ conductance value. Furthermore, the presented design uses P-state MTJs so $\theta = 0$ is chosen. TMR_0 is consistent with [55]; moreover, α and S are consistent in order of magnitude with [66]. C is obtained using [67]:

$$C = 1.387 \times 10^{-4} t/\sqrt{\varphi} \tag{5.9}$$

where t represents oxide barrier height in Angstroms and φ is the oxide barrier potential in electron-volts. C = 0.0015 K^{-1} is derived assuming an oxide barrier thickness of 10 Angstroms [55] and a barrier potential on the order of 1eV [66].

Based on the above model, the mean computational error of the squaring unit across an input voltage range between 0.2V and 0.6V is determined, for temperatures ranging from -20°C to 70°C, with results summarized in Table 5.6. An error below 10% is observed between temperatures of -10°C and 40°C. This operational interval includes common environmental temperatures as well as physiological temperature but does not fully cover the commercial temperature range. Thus, further work is necessary to improve the resilience of the design in more extreme environments.

Table 5.6: Mean error of analog squaring circuit as a function of temperature, T.

T	-20°C	-10°C	0°C	10°C	20°C	30°C	40°C	50°C	60°C	70°C
err	14.1%	3.17%	0.748%	0.352%	0.551%	1.70%	5.71%	14.2%	22.9%	27.3%

5.4 Generalized Functions

The proposed hardware may also be used to implement generalized functions. For one, the analog circuit can function in a third mode where inverse power and root functions are computed by adding an inverting amplifier before the final stage; a $1/\sqrt{x}$ function yields an average error of 0.4%. Furthermore, exponential and logarithmic functions can be computed using only one op-amp stage. Other generalized functions can be implemented using a Taylor series approximation.

Figure 5.9 shows an approximation of the function $f(x) = x - x^2 - x^3 - x^4 - x^5$ based on the proposed analog squaring unit. This simulation includes squaring errors, but neglects errors in

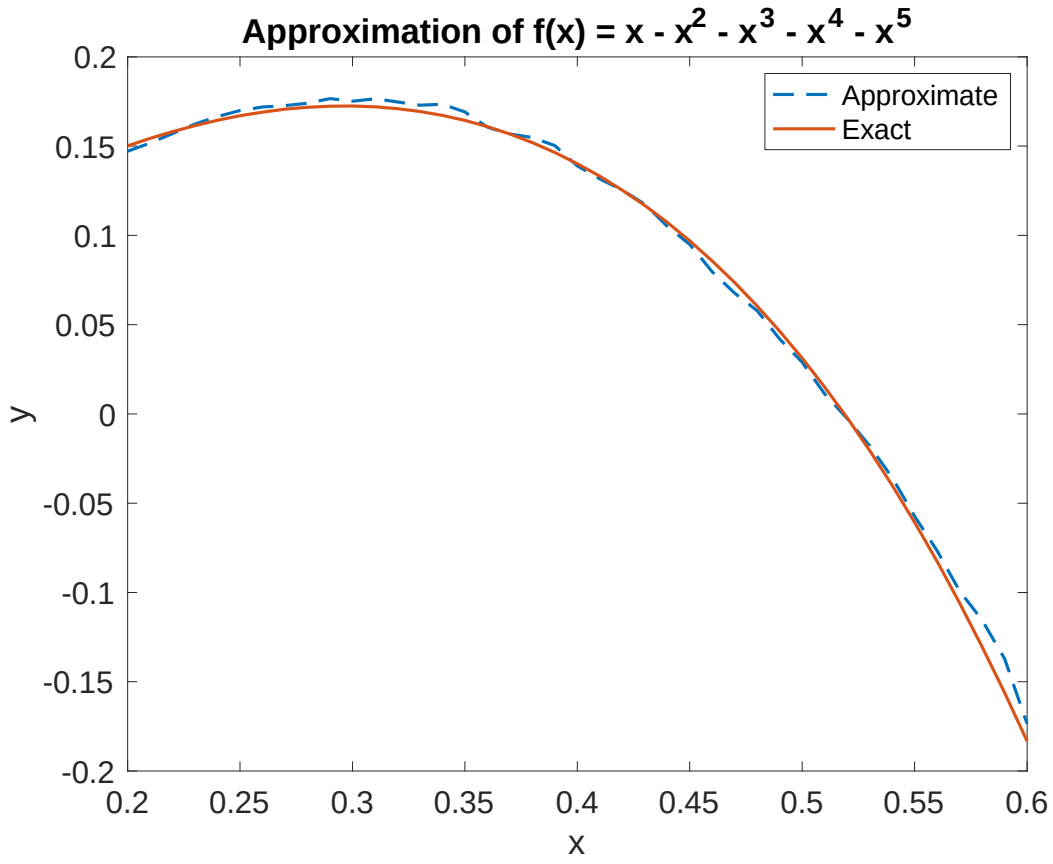


Figure 5.9: Approximation of a 5th order polynomial function using the proposed hardware, showing agreement with an error-free implementation.

addition, subtraction, and voltage rescaling; the resulting output is a fair approximation of the target function, with an average error of 4.83% over the tested range. This demonstrates the feasibility of generating generalized functions through Taylor series using our analog approach.

5.5 Summary

Herein, we have presented an analog circuit capable of multiplication and general exponentiation operations. The circuit is based on a reconfigurable fabric which allows for versatility in the mode of operation as well as tunability in bandwidth, allowing for adaptation to diverse signal processing and machine learning applications. Simulation results on circuit performance indicate reduction in error and 95% reduction in area when compared to a state-of-the-art approximate digital multiplier. Furthermore, a significant reduction in execution time in addition to reduction in complexity is attained in comparison to a time-mode analog exponentiation circuit operating on similar principles. The presented circuit may be used for computation of generalized functions via Taylor series approximation: simulation results indicate less than 5% error in computation of a fifth-order polynomial.

CHAPTER 6: APPLICATIONS OF SPIN-BASED ANALOG COMPUTATION[5]

6.1 Spintronically Configurable Adaptive in-memory Processing Environment (SCAPE)

The Spintronically-Configurable Adaptive in-memory Processing Environment (SCAPE) [135] is designed to reduce overheads at the cost of precision by performing computations intrinsically in the analog domain. As such, SCAPE is well-suited for error-tolerant applications such as CS and image classification. SCAPE consists of three stages: a Vector Matrix Multiplication Stage (VMMS) consisting of a SHE-MRAM crossbar, an Analog Activation Stage (AAS) consisting of the analog computation circuit described previously [136, 137] and finally an Analog to Digital Conversion (ADC) stage. The ADC stage uses the SS-PIR design, as shown in Figure 2.14, to reduce area and energy costs of data conversion.

In contrast to the generally reconfigurable fabric shown in Figure 5.3, AAS provides a partially reconfigurable fabric consisting of 1) the analog computational circuit shown in Figure 5.4, having a reconfigurable resistance, R_3, 2) p-bit devices for stochastic computation, and 3) an op-amp and digital inverters for threshold computations. The switch to partial reconfigurability reduces area overheads while still maintaining the necessary functionality for target applications.

6.2 Application to CS Signal Reconstruction

6.2.1 Implementation of AMP

Approximate Message Passing (AMP), as introduced in Section 2.4.4 and provided below as Algorithm 1, is a CS reconstruction algorithm designed for fast convergence [120]. As an error tolerant application requiring VMM together with square and square root computations in each iteration, AMP is a viable target for SCAPE. AMP begins by initializing the residual vector, $\mathbf{r_0}$, to

[5] ©IEEE. Part of this chapter is reprinted, with permission, from [135, 137].

the measurement vector **y**, as well as initializing the estimate of the signal vector **x** to zero and the counter to 1 (Lines 1 - 3). Next, the threshold θ is computed as the root mean square error of the residual (Line 5). Lines 6 and 7 provide an estimation of the reconstructed signal vector as a function of the thresholding parameter, in accordance with the Iterative Soft Thresholding technique. In the notation, sign(**a**)max(|**a**|-θ, 0) refers to element-wise vector operations. The function sign(x) is defined to be +1 for $x > 0$ and -1 for $x < 0$. Finally, Lines 8 and 9 update the residual similarly as in Iterative Soft Thresholding, with the key difference being the last term in Line 9. The counter is then incremented in Line 10 before the loop repeats.

The AMP algorithm is implemented using the SCAPE hardware architecture. VMM is executed using the VMMS and scalar operations, including multiplication, square, square root, and inverse square root, are executed on the AAS. The AAS is also used to compute the thresholding functions

Algorithm 1 Approximate Message Passing

Inputs: The measurement matrix, $\mathbf{\Phi}$
The measurement vector, $\mathbf{y}$
The number of measurements, m
The number of iterations, k

Output: The approximate signal vector, $\hat{\mathbf{x}}$

Procedure:

1: $\mathbf{r_0} \leftarrow \mathbf{y}$
2: $\hat{\mathbf{x}}_0 \leftarrow \mathbf{0}$
3: $i \leftarrow 1$
4: **while** $i < k$ **do**
5: $\quad \theta \leftarrow \|\mathbf{r}_{i-1}\|/\sqrt{m}$
6: $\quad \mathbf{a} \leftarrow \hat{\mathbf{x}}_{\mathbf{i-1}} + \mathbf{\Phi}^{\mathbf{T}}\mathbf{r}_{\mathbf{i-1}}$
7: $\quad \hat{\mathbf{x}}_{\mathbf{i}} \leftarrow \text{sign}(\mathbf{a})\max(|\ \mathbf{a}\ | - \theta, 0)$
8: $\quad b_i \leftarrow \|\hat{\mathbf{x}}_i\|_0/m$
9: $\quad \mathbf{r}_i \leftarrow \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}_i + b_i\mathbf{r}_{i-1}$
10: $\quad i \leftarrow i + 1$
11: **end while**

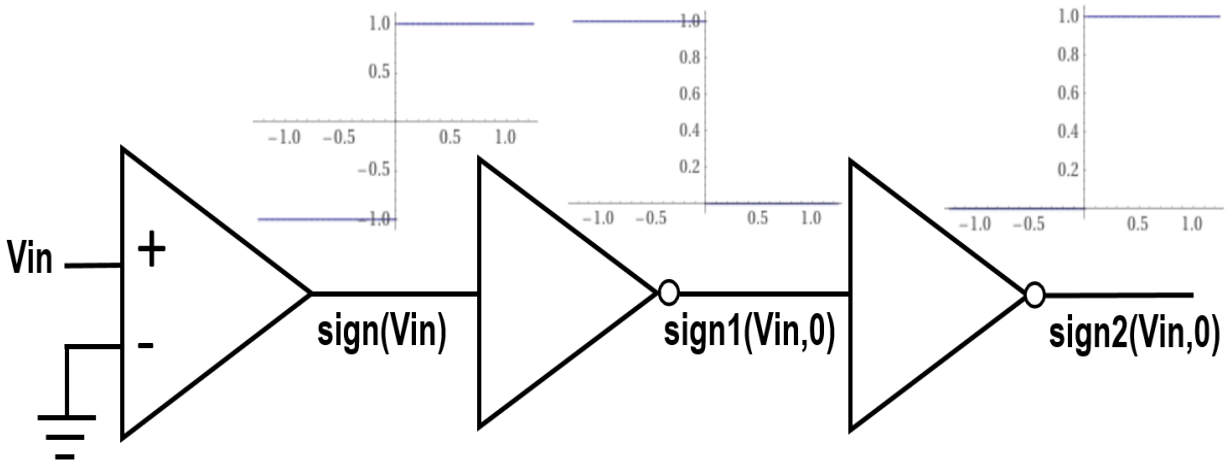


Figure 6.1: An analog design for thresholding operations. The functions y = sign(x), y = sign1(x,0) and y = sign2(x,0) are illustrated in the top panel by the leftmost, middle and rightmost graphs, respectively.

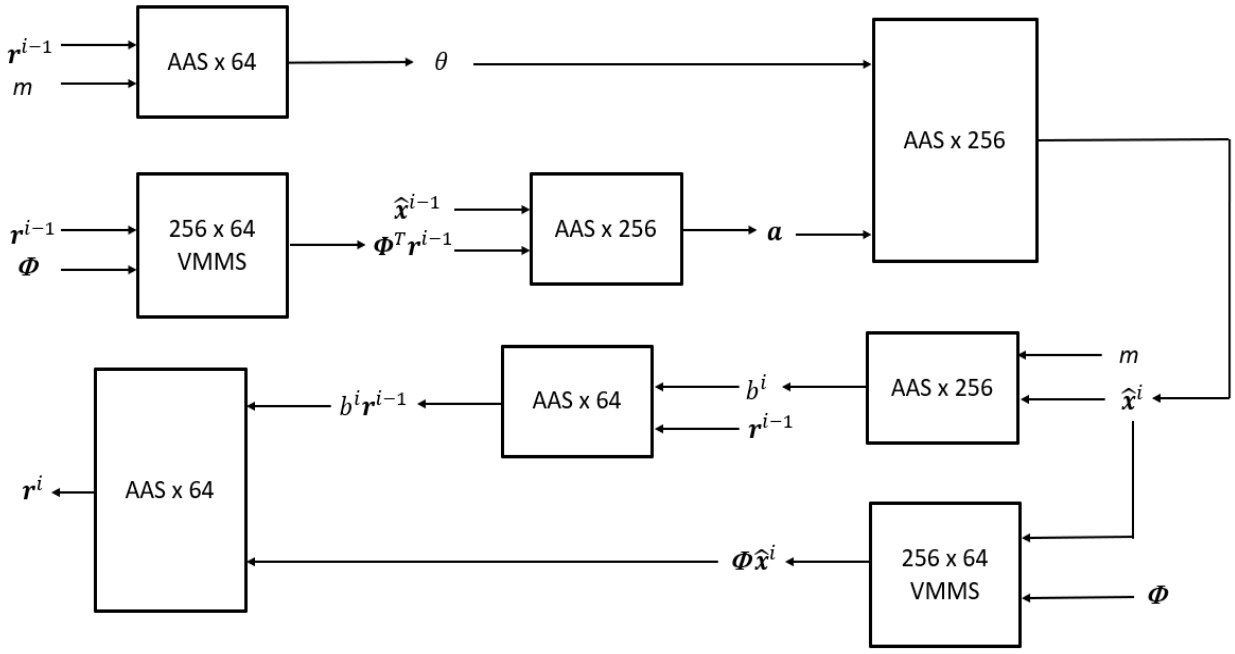


Figure 6.2: Hardware implementation of AMP algorithm.

necessary for AMP via the simple analog design shown in Figure 6.1. In this design, an analog comparator circuit with $V_{ref} = 0$ computes the function y = sign(x). A three-stage design based on a chain of inverters is used for the computation of two-additional functions: y = sign1(x,ref),

defined as 1 when x < ref and as 0 when x > ref, and y = sign2(x,ref), defined as 1 when x > ref and as 0 when x < ref. Based on this hardware, the remaining three functions necessary for AMP may be computed. First, $y = |x|$ is rewritten as $y = x\text{sign}(x)$. Next, $y = \max(x,0)$ is equivalent to $y = x\text{sign2}(x,0)$. Finally, $y = \|\boldsymbol{x}\|_0$ is roughly equivalent to $y = \sum(\text{sign1}(x,-0.05) + \text{sign2}(x,0.05))$, assuming any input with an absolute value greater than 0.05 is considered as "nonzero."

Figure 6.2 demonstrates a hardware implementation of one loop of the AMP algorithm, based on the architecture presented herein. Reconstruction based on a signal size n = 256 and m = 64 requires a 256 × 64 VMMS array to execute the VMM in Line 6 and Line 9 and 256 AAS functional units for scalar operations.

6.2.2 Performance of AMP

The performance of AMP is evaluated in MATLAB based on signals of length n=1000, with sparsity rate k/n = 0.1. The number of measurements, m, is varied from 200 to 500 to determine the magnitude of the reconstruction error in decibels, defined as:

$$error(dB) = 20log\left(\frac{\|\hat{\boldsymbol{x}}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\right). \quad (6.1)$$

Figure 6.3 shows AMP performance considering an exact implementation (blue circles), approximation errors intrinsic to the analog hardware as detailed in Table 5.3 (red circles) and finally approximation errors considering process variation errors detailed in Table 5.4 (yellow circles).

The results demonstrate a negligible impact of the intrinsic circuit error on AMP performance; certain data points such as m=500 demonstrate a lower error with the approximate approach, indicating statistical insignificance of the error. Even the increased computational error resulting

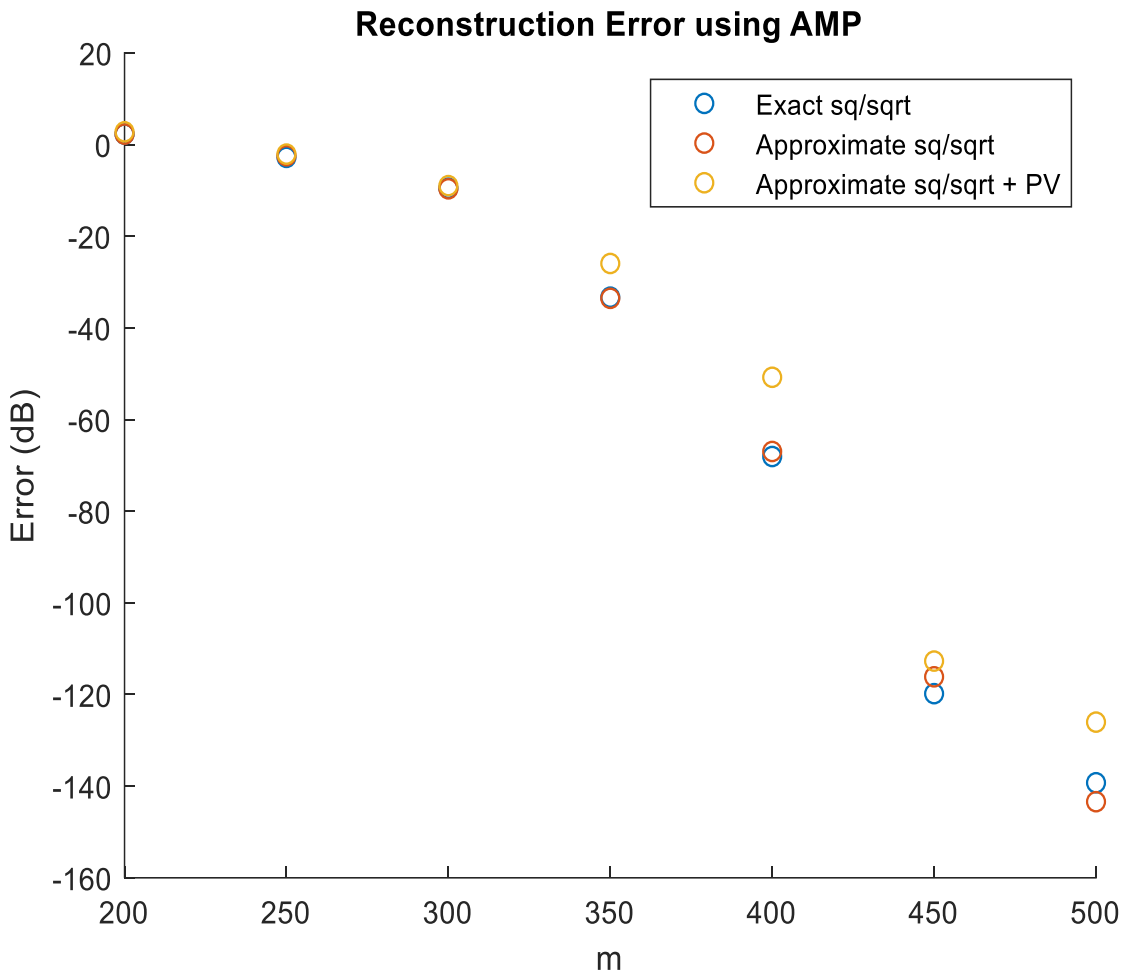


Figure 6.3: Signal reconstruction error of the AMP algorithm as a function of number of measurements, where square and square root operations are performed exactly (blue circles), with approximation error of the presented hardware (red circles), and with approximation error including process variation (yellow circles).

from process variation amounts to only a slight degradation in performance and consistently requires less than 50 additional measurements to regain the reconstruction accuracy of the AMP algorithm.

To determine the total energy cost of AMP, SPICE simulations are performed to determine the per-cell energy cost of the VMMS, as well as the energy cost per operation of the scalar functions performed by the AAS; the results are aggregated to determine the total computational energy cost

Table 6.1: Breakdown of AMP Circuit Energy Consumption.

Operation	Hardware Units	Energy Cost
$\|\boldsymbol{r}^{i-1}\|$.	AAS	47.6pJ
$\theta = \|\boldsymbol{r}^{i-1}\|/\sqrt{m}$	AAS	1.1pJ
$\boldsymbol{a} = \hat{\boldsymbol{x}}^{i-1} + \boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{r}^{i-1}$	VMMS + AAS	1.654nJ
$\hat{\boldsymbol{x}}^{i} = \text{sign}(\boldsymbol{a})\ \max(\text{abs}(\boldsymbol{a}) - \theta, 0)$	AAS	1.24nJ
$b^{i} = \|\hat{\boldsymbol{x}}^{i}\|_0/m$	AAS	0.58nJ
$\boldsymbol{r}^{i} = \boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{x}}^{i} + b^{i}\boldsymbol{r}^{i-1}$	VMMS + AAS	1.65nJ
Total		5.17nJ

Table 6.2: Comparison of AMP Energy Consumption.

	Herein	Herein	[138]	[118]
Tech. node	14nm	14nm	65nm	65nm
V_{DD}	0.8V	0.8V	N/A	1.2V
Array size	256x64	1024x512	256x64	1024x512
Array precision	8 bits	8 bits	1 bit	26 bits
#Iterations	50	20	50	20
Energy/sample	1.0nJ	2.1nJ	27nJ	61nJ

of running one cycle of AMP. The VMMS consumes a total of 3.15nJ while total energy consumption by the AAS is 2.02nJ, for a total computational energy consumption of 5.17nJ. For 50 iterations, this gives an energy overhead equal to 258nJ for running AMP. Analysis of signal reconstruction error associated with approximations in the AAS units was performed for a signal of size n = 1000, and sparsity k = 100, where n is the total number of elements in each frame of the signal, and k is just the total number of elements per frame that are non-zero. The average

accuracy degradation resulting from computational error was found to be 1.1dB, which is negligible.

Table 6.1 lists the breakdown of energy per computation in execution of a single AMP cycle using the proposed design. A total energy cost of 5.17nJ per cycle yields a total energy consumption of 1.0nJ per sample, assuming 50 iteration cycles and a reconstructed signal consisting of 256 samples. Table 6.2 displays an energy comparison to two recent ASIC implementations for AMP [118, 138]; hardware running the Enhanced AMP algorithm (EAMP) [138] over 50 iterations under the same CS parameters of $(n,m) = (256,64)$ consumes 315mW of power and executes in 8900 clock cycles on a 400MHz system. Thus, the energy consumption is roughly 7μJ, and roughly 27nJ per sample. EAMP is roughly in line with the standard AMP algorithm in terms of mean square error, up to 100 iterations. Thus, the full-analog approach to AMP presented herein provides significant benefits in energy while having a minimal impact on reconstruction accuracy.

6.3 Application to MNIST Digit Recognition

6.3.1 Gradient Decay Problem

Deep neural networks (DNNs) have been gaining popularity in the context of diverse applications including computer vision [139] and speech recognition [140]. At each layer, the DNN takes a vector input, **x**, and outputs a linear transformation of the input, **z,** according to the equation $\mathbf{z} = \mathbf{W}\mathbf{x}$ where **W** is the weight matrix. To facilitate learning non-linear relationships, the output **z** is multiplied by an activation function to yield a final layer output, $\mathbf{h} = f(\mathbf{x})$. The choice of activation function has recently been a subject of research interest due to its significant impact on the training accuracy of a neural network [35]. While hyperbolic tangent has been used frequently, this function suffers drawbacks including the *gradient decay problem,* i.e., the gradient

becomes diminished in multi-layer networks due to repeated multiplication of values having absolute value less than 1 [141].

The gradient decay problem has been addressed by choice of alternative activation functions, e.g., the Rectified Linear Unit (ReLU) which is defined as $f_{\mathrm{ReLU}}(x) = \max(0, x)$ and has a gradient of 1 for all $x > 0$. Another alternative is the square root function, which experiences significantly slower gradient decay compared with hyperbolic tangent. It has been observed that the derivative of the hyperbolic tangent function at $x = 10$ is less than the derivative of the square root function at $x = 10^{16}$. Previous research has demonstrated that replacing hyperbolic tangent with a square root activation function can allow for a 5% improvement in classification accuracy on the CIFAR-10 dataset [34].

Given the robust capabilities of the analog circuit presented herein, we next evaluate its ability to generate improved activation functions for DNN performance. The evaluation is performed in the context of a Deep Belief Network (DBN) used to classify samples from the MNIST dataset.

6.3.2 Impact of Activation Function

Leveraging the capabilities of the analog circuit presented herein, we investigate the impact of three separate activation functions on DBN performance: $f_1(x) = \frac{1}{2}(1 + \tanh(x))$, $f_2(x) = \sqrt{f_1(x)}$, and $f_3(x) = (1 + e^{-x})^{-1}$. Since $f_2'(x) > f_1'(x)$ for $x < -0.55$ and $f_3'(x) > f_1'(x)$ for $|x| > 1.06$, substitution of these functions may potentially alleviate the rate of gradient decay for certain inputs. Moreover, each function may be implemented using the FPAA fabric shown in Figure 5.3; the presence of low-barrier MTJ devices allows for construction of p-bit devices, at which point f_1 is computed via an op-amp integrator at the output. Computation of f_2 requires an

additional 3 op-amps to execute the square root function in analog; finally, f_3 requires a total of 6 op-amps to implement.

A DBN software simulation is performed in MATLAB for each activation function to evaluate the classification accuracy for the MNIST dataset, based on 3000 training samples and 1000 test samples. Figure 6.4 shows the results based on various network topologies. Over the network topologies tested, both f_2 and f_3 demonstrate a consistent improvement in error rate over f_1; the average improvement is 6.4% for f_2 and 8.7% for f_3. Moreover, in certain cases, selection of f_3 versus f_1 as an activation function allows for reduction in error rate while decreasing the size of the array, e.g., from 784×500×10 to 784×200×10, and from 784×200×200×200×10 to 784×100×100×100×10.

A PIN-Sim simulation is conducted, based on the MTJ parameters listed in Table 6.3, for average RBM power consumption in select network topologies using the f_1 and f_2 activation functions. For simulations implementing f_2, the neuron.sp file in the PIN-Sim framework is modified by adding an analog square root unit to the output, using the circuit shown in Figure 5.4.

Simulation results are listed in Table 6.4, including average power consumption and corresponding software error rates; the Power-Error-Product (PEP) is computed as a product of these data points and listed in the table as well. Similar to the previously used Energy-Error-Product [124], PEP is a useful metric for attaining an overall evaluation of each design. Based on the results, the f_2 activation function yields an improvement in PEP for each of the tested topologies; the average improvement is 17.4%.

Table 6.3: MTJ simulation parameters.

Parameter	Value
Saturation Magnetization	1100 emu/cc
Free layer diameter, thickness	22nm, 2nm
Polarization	0.59
TMR	110%
MTJ RA-product	9mW-cm^2
Damping coefficient	0.01
Temperature	300K

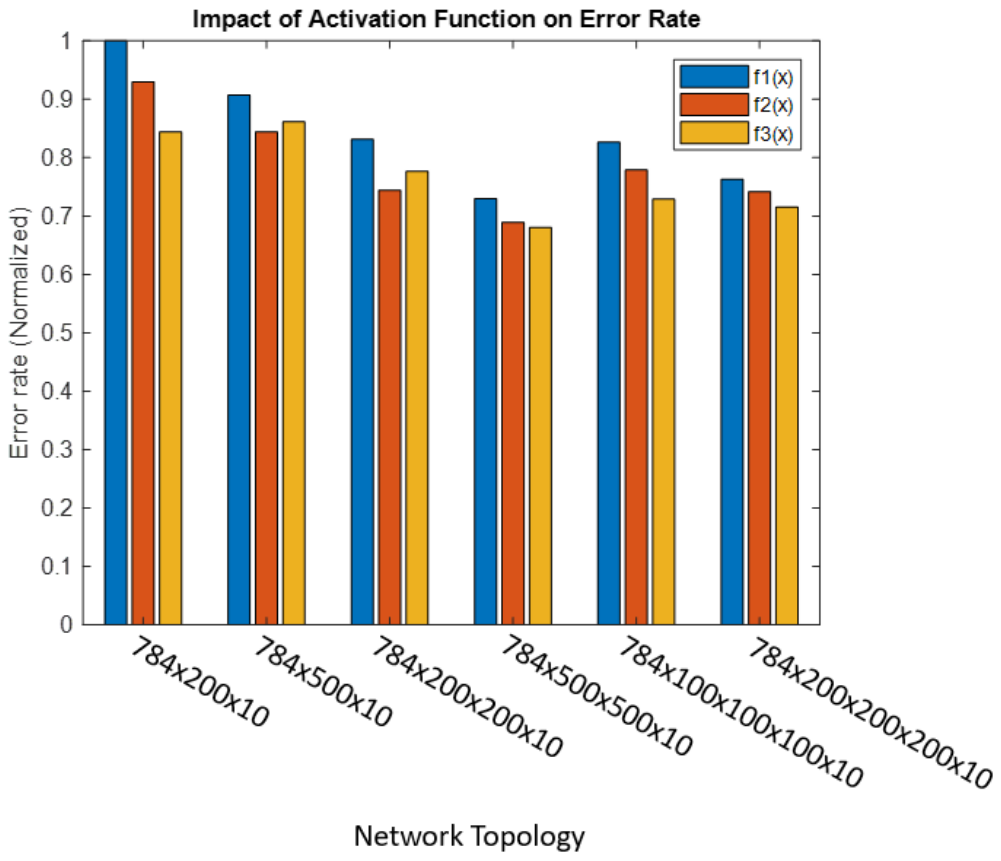


Figure 6.4: Normalized error rate for image classification, based on various DBN topologies and activation functions.

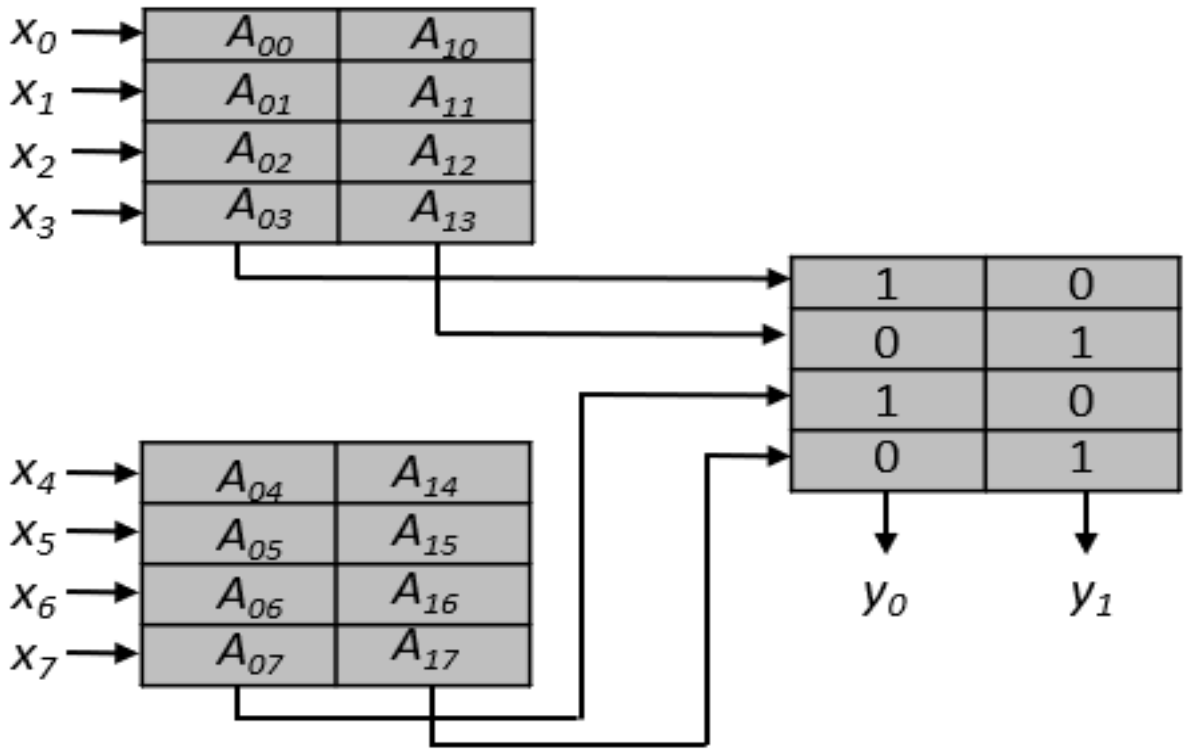


Figure 6.5: Technique for splitting a VMM operation, **y** = **Ax**, between smaller crossbar arrays. In this case, an 8×2 VMM operation is split between 4×2 crossbars.

Table 6.4: Error rate, average DBN power consumption and Power-Error-Product (PEP) for various network topologies and activation functions.

Network topology	Act. function	Error rate	Power (mW)	PEP
784×200×10	f_1	0.1239	72.4	8.97
784×200×10	f_2	0.1152	76.1	8.77
784×200×200×10	f_1	0.1030	106.3	10.95
784×200×200×10	f_2	0.0922	88.5	8.16
784×200×200×200×10	f_1	0.0945	153.7	14.52
784×200×200×200×10	f_2	0.0919	119.2	10.95

6.3.3 Mapping Larger Networks

This section provides a brief analysis of the scalability of the presented architecture to larger-sized networks. While the scope of this research has been limited to MNIST benchmarks, larger data sets can be processed by splitting matrices present in software between multiple crossbar arrays. This is similar to the technique given in [142]. Figure 6.5 shows a representative example of 8×2 VMM, performed via matrix splitting. Mathematically, this operation is given as $\mathbf{y} = \mathbf{Ax}$, where $\boldsymbol{x} \epsilon \mathbb{R}^8$, $\boldsymbol{A} \epsilon \mathbb{R}^{2\times 8}$ and $\boldsymbol{y} \epsilon \mathbb{R}^2$. By leveraging the linear properties of dot product, the input vector, $\boldsymbol{x}$, can be split between two 4×2 crossbar arrays to yield intermediate outputs. The intermediate outputs are then added to produce the final output, $\boldsymbol{y}$, using a third crossbar consisting of 1 and 0 elements. Transimpedance amplifiers, which convert current to analog voltage at the output of each array, act to ensure that the output from each array is within the operational input range of the next array. Generally, the accumulated current does not scale with input size. If the crossbar size is limited to 128×64, then the maximum output current is 9.4mA, under a line resistance of 2.5Ω per cell.

6.4 DBN Accuracy Enhancement via Triple Modular Redundancy

6.4.1 Redundant Computing

Redundancy is a useful strategy for improving recognition accuracy of a DBN classifier without incurring the overheads of a more complex network. One implementation is Spatial Triple Modular Redundancy (STMR), as Figure 6.6 shows. In the STMR approach, the image classification is performed three times in parallel and each input is converted to a digital representation using SS-PIR. The majority of these outputs then determines the final result of the circuit. It is advantageous to use a distinct activation function in each cycle to avoid common-mode misrecognition resulting from model similarity.

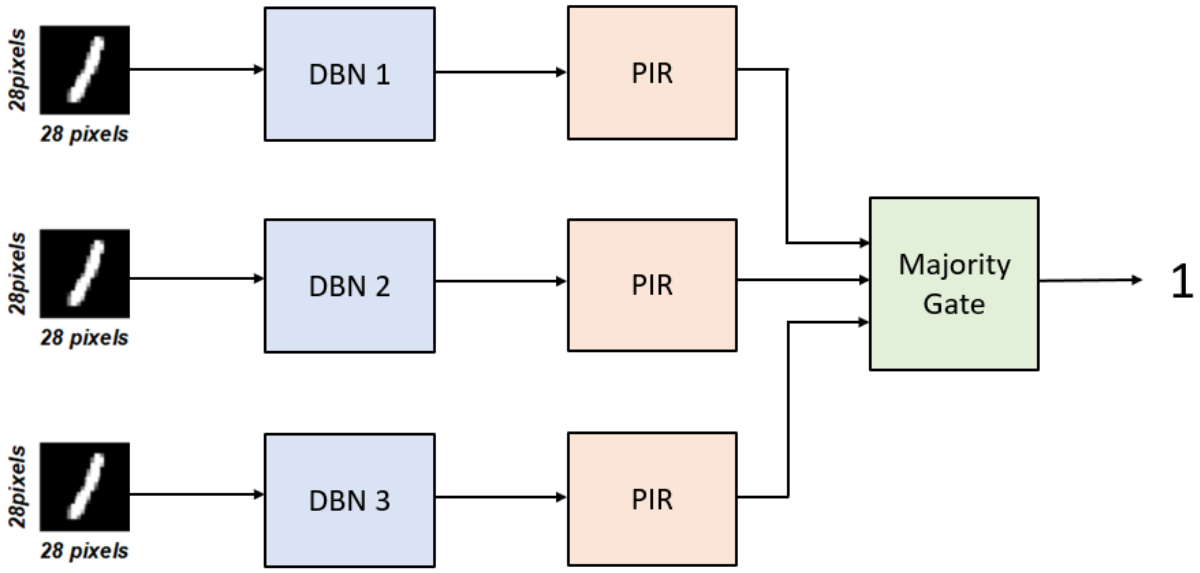


Figure 6.6 Spatial Triple Modular Redundancy (STMR) architecture.

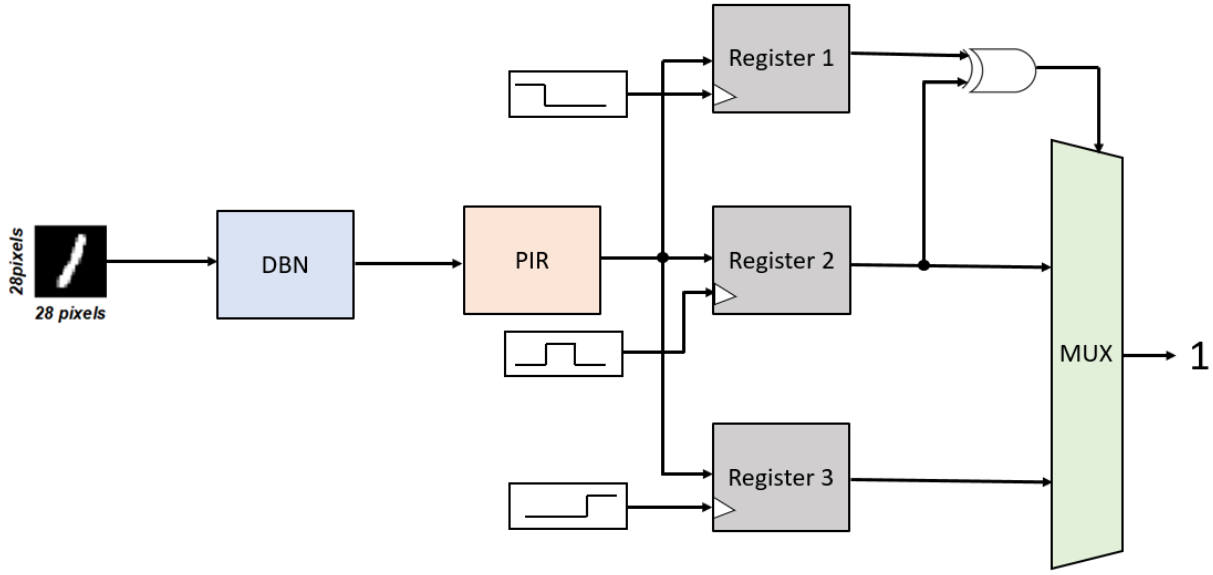


Figure 6.7: Progressive Triple Modular Redundancy (PTMR) architecture.

An alternative approach is provided by Progressive Temporal Modular Redundancy (PTMR), as Figure 6.7 shows. In contrast to STMR, PTMR does not physically duplicate the network. Instead, computations are done sequentially and results are stored in registers, each having a clock input shifted by one cycle. As in STMR, the majority of outputs is used to determine the final

result of the circuit. However, PTMR determines the majority using a multiplexer with the XOR result of the first two PIR outputs as the selector. If the first two outputs match, then the XOR result is a 0, in which case the second PIR output is selected as the final result. Otherwise, if there is a mismatch between the first two outputs, the XOR result is a 1 and the third PIR output is used as the final result. The advantage to the multiplexer approach is that the computation can be halted after two cycles when the first two outputs match, which occurs in the majority of cases. STMR and PTMR deliver separate tradeoffs: while STMR is faster, PTMR is more area efficient, and also more energy efficient due to the intermediate halting capability.

6.4.2 Performance of STMR and PTMR

We evaluate the application-level performance of a DBN running on the SCAPE platform and employing the STMR and PTMR redundancy techniques for 784×100×10 DBNs, against a baseline 784×500×500×10 DBN trained on sigmoid activation without redundancy. Hybrid spin-based majority and XOR gates are utilized in STMR and PTMR, respectively. Activation functions are computed in analog using the AAS of the SCAPE platform. Simulations are performed using HSPICE, with SHE-MTJ model and device parameters similar to those used in [135] and listed in Table 6.5. Registers, multiplexers and other control peripherals are designed using the CMOS PTM 14nm HP library [127], at V_{DD} = 0.8V.

Majority and XOR gates implemented based on the designs in [143] and [144] consume 0.0273mW and 0.0375mW, respectively. The power consumption of the overall peripherals consisting of majority gates in STMR, as well as XOR gates, registers and multiplexers in PTMR, are evaluated as 0.819mW and 1.13mW, respectively. The 3 DBN stages in STMR and PTMR are trained with 3000 images from the MNIST data set and tested using 100 images.

Table 6.5: SHE-MTJ and CMOS Device Simulation Parameters.

Symbol	Parameter	Value
R_P	Parallel MTJ Resistance	2.8kΩ
R_{AP}	Anti-Parallel MTJ Resistance	5.6kΩ
TMR	Tunnel Magnetic Ratio	100%
α	Damping Coefficient	0.007
T	Temperature	300K
P	Polarization	0.52
V_{th_pmos}	Threshold Voltage (PMOS)	460mV
W_{pmos}	Width (PMOS)	44nm
V_{th_nmos}	Threshold Voltage (NMOS)	500mV
W_{nmos}	Width (NMOS)	22nm
MTJ Area	MTJ Length × MTJ Width × $\pi/4$	60nm×30nm×$\pi/4$
HM Volume	L × W × T	100 nm×60 nm×3 nm

Table 6.6: Evaluation of STMR and PTMR based on error, power, delay and area.

Network	Avg. Error	Avg. Power	Delay	Norm. X-bar Area	PEP
A	30%	316.7mW	17ns	647,000x	95.01
B	45%	43mW	13ns	79,400x	19.35
B (PTMR)	27%	44.02mW	45ns	79,400x	11.88
B (STMR)	27%	167.2mW	13ns	238,200x	45.14

For the PTMR approach, the sampling time is 45 ns, i.e., 3 times the delay of the STMR architecture, plus additional delay to rewrite weights before each trial. There are no overheads associated with training the networks since trained weights and biases are pre-loaded into the input buffers. Results are listed in Table 6.6 for a large 784×500×500×10 network architecture (labeled as A) and a significantly smaller 784×100×10 network (labeled as B). Both of these baseline networks are trained using the sigmoidal activation function. Also considered is B (PTMR), which consists of executing PTMR over 3 cycles using Network B trained with sigmoidal, sigmoidal square root and sigmoidal square activation functions, respectively. In this context, the sigmoidal activation function is defined as $f(x) = \frac{1}{2}(1 + \tanh(x))$ and sigmoidal square root and square

are defined as $\sqrt{f(x)}$ and $[f(x)]^2$, respectively. B (STMR) uses the STMR approach with the same network and activation functions as B (PTMR). Error is based on the top-3 recognition accuracy following analog to digital conversion using PIR, average power consumption and delay include the entire circuit including peripherals, area is calculated only using the crossbar, and PEP is the Power-Error-Product which gives a quantitative measure of digit-recognition efficiency of spin-based DBNs.

The results show that the accuracy of Network B is significantly lower, compared to Network A. However, use of triple modular redundancy with different activation functions allows comparable accuracy using Network B. Both B (PTMR) and B (STMR) show improvements in terms of power, area and PEP, compared to the baseline, with the STMR approach trading off area and power for speed. The PTMR allows for reduced power consumption as well as area, achieving 86.1% reduction in power, 87.7% reduction in area overhead and 87.5% reduction in PEP, at the cost of a 2.6× increase throughput latency. B (PTMR) saves power by selectively using the third computational cycle, which occurs only 35% of the time. However, the PTMR approach consumes additional power due to peripherals, including use of the AAS for computation of enhanced activation functions. Thus, the time-averaged power consumption of B (PTMR) is similar to that of the Network B baseline.

Further evaluation of PTMR is conducted using a more comprehensive set of network architectures. Power (not including peripherals), error and PEP are provided for each network topology in Figure 6.8, Figure 6.9 and Figure 6.10 respectively. The Mixed topology consists of 784×200×10 networks in the first two PTMR stages, followed by a 784×100×10 network in the third stage. With the exception of the Mixed topology, the sigmoidal, sigmoidal square root and sigmoidal square activation functions are used in the first, second and third stages, respectively.

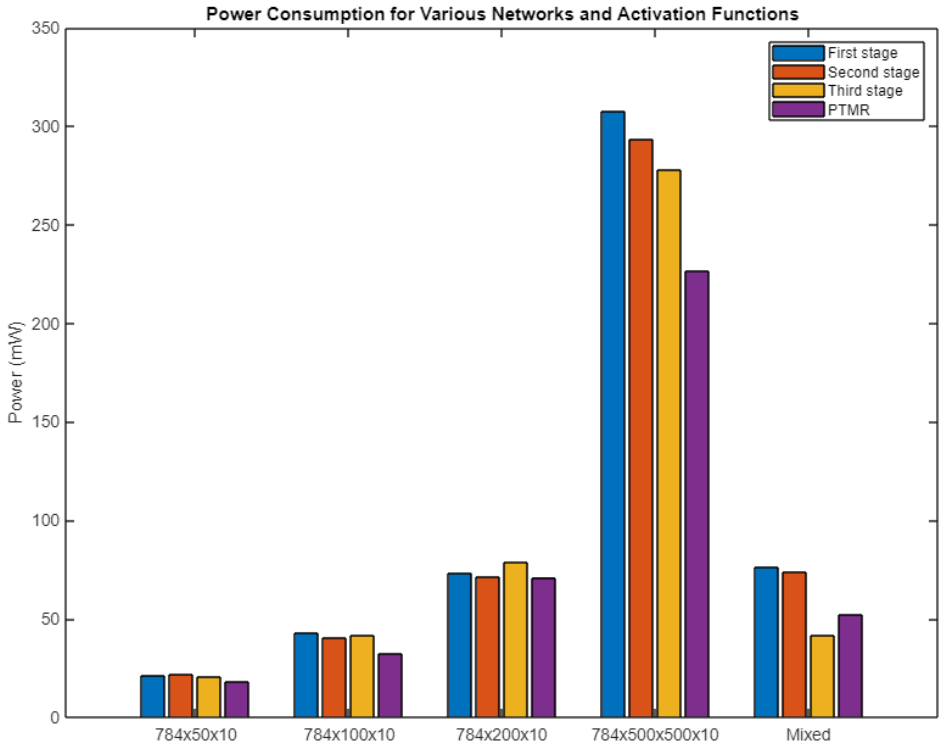


Figure 6.8: PTMR power consumption for various DBN network topologies.

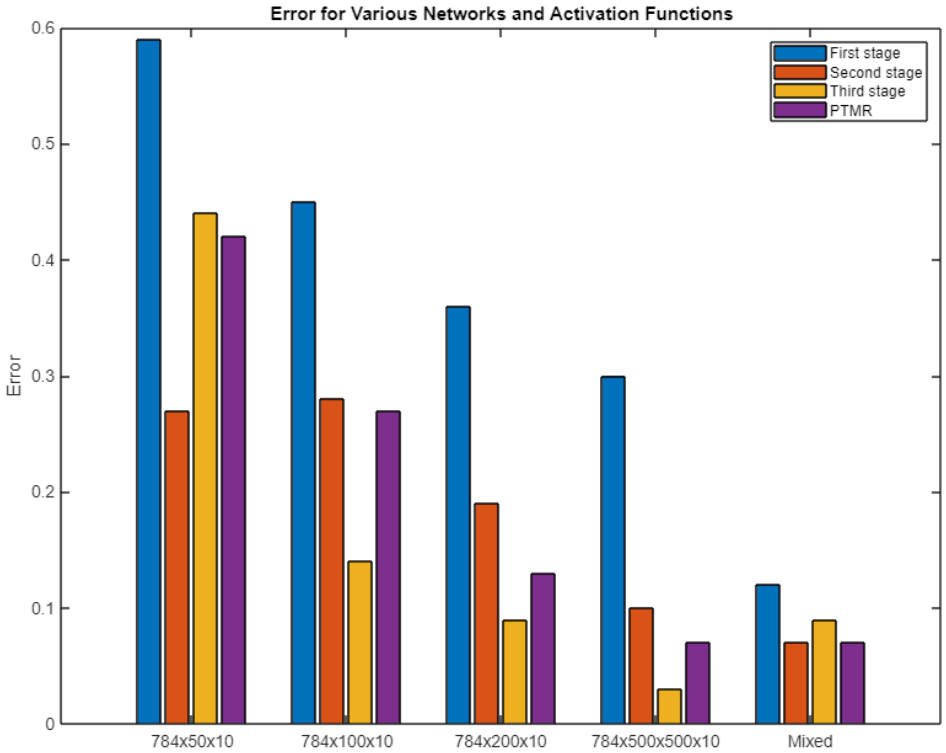


Figure 6.9: PTMR image classification error rate for various DBN network topologies.

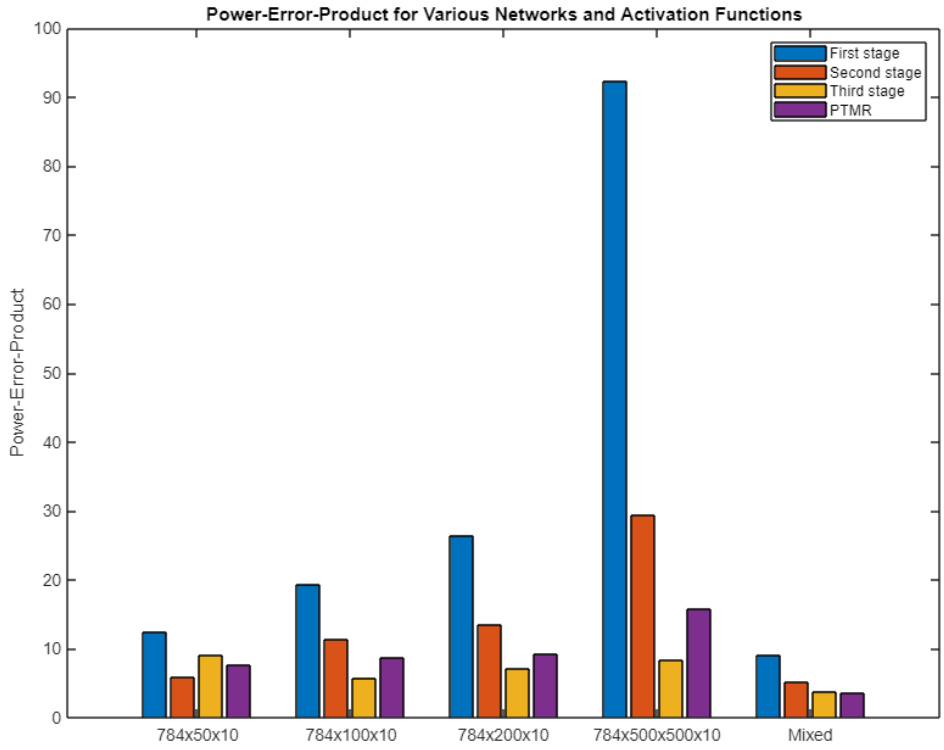


Figure 6.10: PTMR Power-Error-Product for various DBN network topologies.

The Mixed topology makes use of the advantages offered by enhanced activation functions by using a sigmoidal square root activation function in the first stage, followed by a sigmoidal square function in the second and third stages.

Based on the data, the following observations may be made: 1) PTMR offers a lower error rate than the 784×500×500×10 network, for every topology shown besides 784×50×10 and 2) PTMR using the Mixed topology yields a 7% error rate, which is only 4% greater than the best performance offered by the 784×500×500×10 network, achieved using sigmoidal square activation. However, PTMR using the Mixed topology offers an 81.2% reduction in power and 56.2% reduction in PEP compared to a single run of the larger network. Furthermore, Figure 6.9 reinforces the observation that choice of activation function significantly impacts network performance, since there is a 64.8% average reduction in error between the first stage using

sigmoidal activation and the third stage using sigmoidal square activation, among the four network topologies tested. Finally, PTMR consistently yields an error rate between that achieved using sigmoidal square and sigmoidal square root activation. Thus, PTMR is beneficial since it is commonly not known which activation function will yield the best error rate.

6.5 Summary

The analog computational circuit presented in Chapter 5 is combined with a spin-based crossbar architecture to yield the Spintronically-Configurable Adaptive in-memory Processing Environment (SCAPE) for signal processing and machine learning computations. The AMP signal reconstruction algorithm implemented using SCAPE yields a 96% energy reduction compared to a recent design, with negligible loss in accuracy. Furthermore, SCAPE allows efficient computation of activation functions in analog for machine learning applications. Simulation results demonstrate that varying the activation function of a neural network can allow for significant improvements in accuracy without increasing network size.

The ability to efficiently compute diverse activation functions enables construction of architectures using triple modular redundancy, which incorporate multiple activation functions to eliminate common-mode misrecognition of input data. Simulation results indicate a 64.8% average reduction in error attained by replacing the sigmoidal activation function with a squared sigmoidal function. Furthermore, it is seen that triple modular redundancy using a collection of smaller networks operating under distinct activation functions yields error rate within 4% of that of a significantly larger network, with 81.2% reduction in power.

CHAPTER 7: LAYER-WISE QUANTIZATION OF DEEP BELIEF NETWORKS

7.1 DNN Precision Analysis

Due to the evolving complexity of DNNs, management of resource utilization has become key to their hardware implementation. State-of-the-art networks may require approximately 1 billion multiply-and-accumulate operations and storage of 100 million parameters [145]. Compression techniques such as pruning [146, 147] and quantization [145, 148] have been suggested as approaches to mitigating these overheads.

Recent research [145] has shown that the layers within a DNN have varying sensitivities to quantization. Specifically,

$$p_m \leq \sum_{l=1}^{L}\left(\Delta_{A,l}^2 E_{A,l} + \Delta_{W,l}^2 E_{W,l}\right) \tag{7.1}$$

where p_m is the probability that the predicted label of a fixed-point network is different from the predicted label of a high-precision floating-point network. Furthermore, L is the number of layers, $\Delta_{A,l}$ is the quantization step-size of the activation function at layer l and $\Delta_{W,l}$ is the quantization step-size of the weights at layer l. The quantization step-sizes can be expressed as $\Delta_{A,l} = 2^{-(B_{A,l}\ -\ 1)}$ and $\Delta_{W,l} = 2^{-(B_{W,l}\ -\ 1)}$ where $B_{A,l}$ and $B_{W,l}$ refer to layer-wise bit-widths used to quantize the activation function and weights, respectively, at layer l. Finally, $E_{A,l}$ and $E_{W,l}$ are defined as:

$$E_{A,l} = \mathrm{E}\left(\sum_{i\neq Y_{fl}} \frac{\sum_{h\in A_l}\left|\frac{\partial(Z_i - Z_{Y_{fl}})}{\partial A_h}\right|^2}{24\left|Z_i - Z_{Y_{fl}}\right|^2}\right), \tag{7.2}$$

$$E_{W,l} = \mathrm{E}\left(\sum_{i\neq Y_{fl}} \frac{\sum_{h\in w_l}\left|\frac{\partial(Z_i - Z_{Y_{fl}})}{\partial w_h}\right|^2}{24\left|Z_i - Z_{Y_{fl}}\right|^2}\right), \tag{7.3}$$

where Y_{fl} is the predicted label of a floating-point network, Z_i refers to the soft outputs, and A_l and w_l are layer-wise indexing sets of activations and weights, respectively.

Eq. 7.2 – 7.3 express the non-uniformity in sensitivity of DNN layers. For example, a layer where the soft outputs vary more strongly with weight values produces a greater value of $E_{W,l}$ as a result of Eq. 7.3. Thus, the weights in this layer are more sensitive to noise and require greater precision as a result of Eq. 7.1.

An alternative layer-wise precision analysis is given by [148], which shows that, given a maximum accuracy degradation Δ_{acc},

$$\left\|\boldsymbol{r}_Z^i\right\|^2 \leq t_i(\Delta_{acc})\frac{(z_1-z_2)^2}{2} \tag{7.4}$$

where $\boldsymbol{r}_Z^i$ is the noise on the last feature map, *Z*, resulting from quantization of weights in layer *i*, and is given by $\mathrm{E}\left(\left\|\boldsymbol{r}_Z^i\right\|^2\right) = C_i e^{-\alpha b_i}$ with b_i representing bit-width in layer *i*. Moreover, t_i is a robustness parameter and z_1 and z_2 are the top two elements of *Z*. According to Eq. 7.4, a greater separation between z_1 and z_2 allows for greater quantization noise in the network without affecting accuracy; moreover, layers with a higher robustness parameter, $t_i(\Delta_{acc})$, are more tolerant to noise and may be assigned a coarser quantization without affecting accuracy.

7.2 Architecture for Layer-wise Quantization

Herein an MRAM-based crossbar architecture is proposed for layer-wise quantization of DBNs. The scope of the presented research is limited to quantization of weights since activation functions are computed in analog. As Figure 7.1 shows, the input stage of the proposed architecture consists of MRAM-based NVM crossbar arrays (labeled as NVM X-BAR in the figure). In PIN-Sim, the weighted connections of the network are represented by devices with resistance values in the range

from 1kΩ to 5kΩ. Thus, in the proposed design, two classes of crossbars are included: Class A crossbars, shown in orange, have MTJs with resistances of 1kΩ and 5kΩ in the P- and AP-states, respectively. Moreover, Class B crossbars, shown in blue, have MTJs with resistances of 2kΩ and 4kΩ. In the scope of the presented research, each layer is assigned either 1-bit or 2-bit quantization. A single Class A crossbar is allocated to layers that are assigned a 1-bit quantization. Layers that require 2-bit quantization are assigned a Class A crossbar together with a Class B crossbar. Thus,

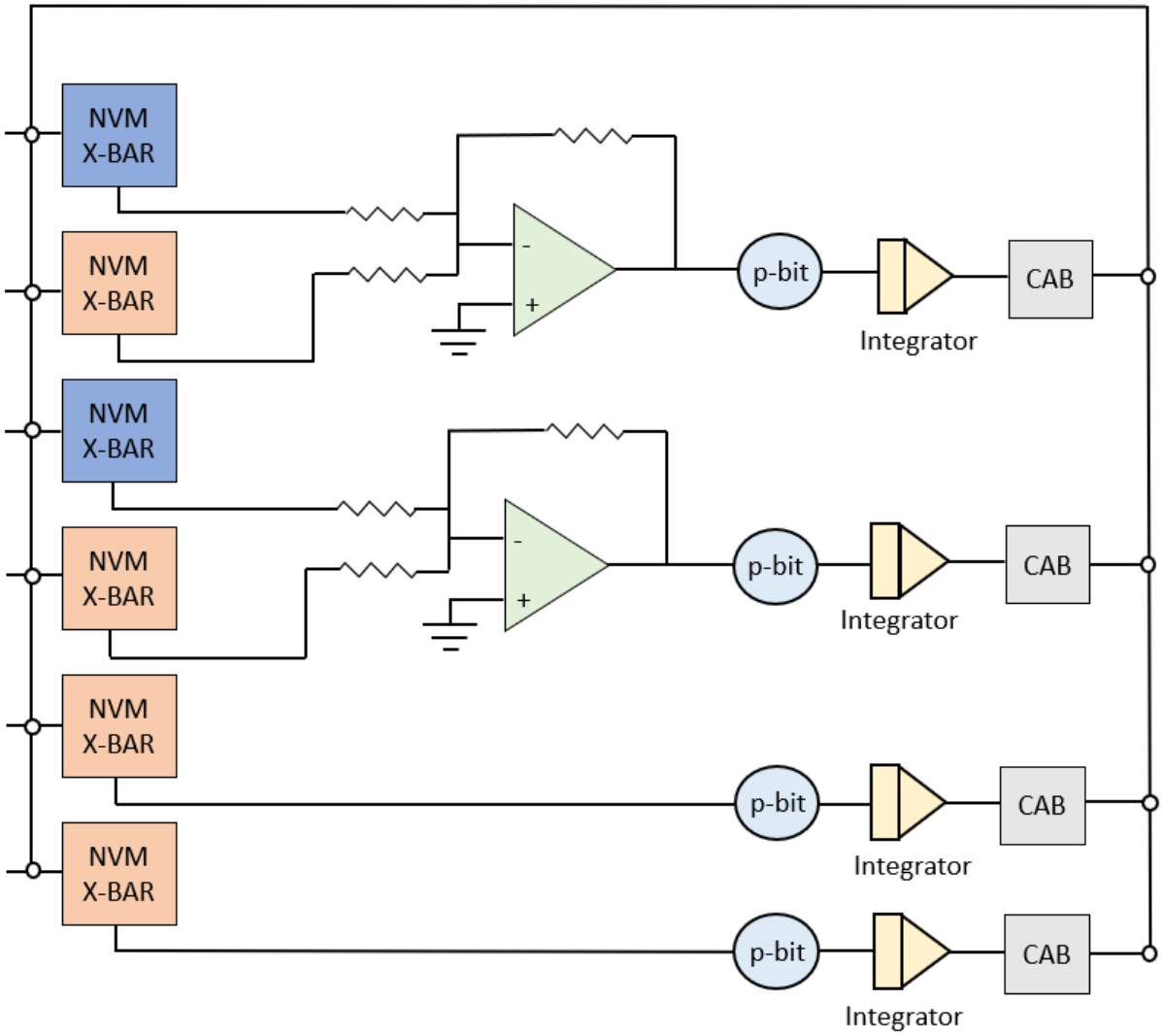


Figure 7.1: Architecture for layer-wise quantization of DBNs.

weights in these layers may be mapped to 1-, 2-, 4- or 5-kΩ resistances. When mapping is performed, the crossbar that contains the device capable of reaching the target resistance value is written; for 2-bit layers, the corresponding device in the second assigned crossbar is deactivated by means of access transistors. For layers assigned a 2-bit quantization, the final outputs from the two allocated crossbars are combined using an analog adder, as shown in Figure 7.1.

Once dot product operations for a given layer are complete, the activation function is computed using a three-stage pipeline consisting of a p-bit device and integrator combined with a Computational Analog Block (CAB) for adaptive selection of enhanced activation functions as described in Chapter 6.

7.3 Optimization using Genetic Algorithm

A Genetic Algorithm (GA) approach is proposed for optimal allocation of crossbars to network layers. GAs have previously been used for resource allocation problems; for example, to determine optimal allocation of FPGA processing elements to DCT coefficients [29]. GAs are particularly useful for larger networks, which may have over 1000 layers [145]. In the GA methodology, hardware configurations are represented using chromosomes. Figure 7.2 gives an example of the chromosome mapping methodology given a 4-layer DBN. The figure shows four crossbar groups, such that assignment to Group 1 or 2 corresponds to a 2-bit quantization while assignment to Group 3 or Group 4 corresponds to 1-bit quantization. The chromosome consists of four elements, i.e., genes, such that the position of each element indicates the DBN layer, while the value of the element indicates the number of bits allocated to that layer. In the figure, 2 bits are allocated to weights in Layers 2 and 3 while 1 bit is allocated to weights in Layers 1 and 4. This configuration is represented by the chromosome 1221.

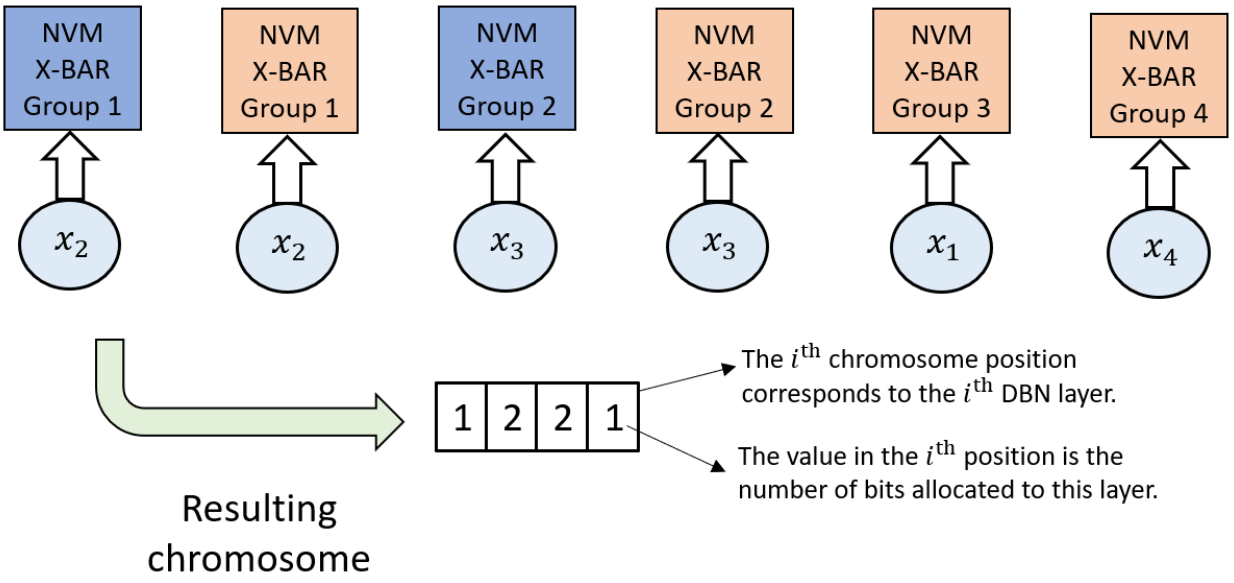


Figure 7.2: Illustration of GA methodology.

The GA, provided by Algorithm 2, commences by initializing the index, g, to zero and calling the initialize() function to generate a random population of N chromosomes (Lines 1 – 2). Each chromosome has a fitness value defined as $f_i = 1/AEP_i$, where AEP_i is the DBN area-error-product. In Line 4, the GA computes the fitness of each individual in the population by calling the evaluate() function. The algorithm then applies the standard evolutionary operators: selection, crossover, mutation and elitism, as described in [102, 149]. In Line 5, select() chooses *n* individuals

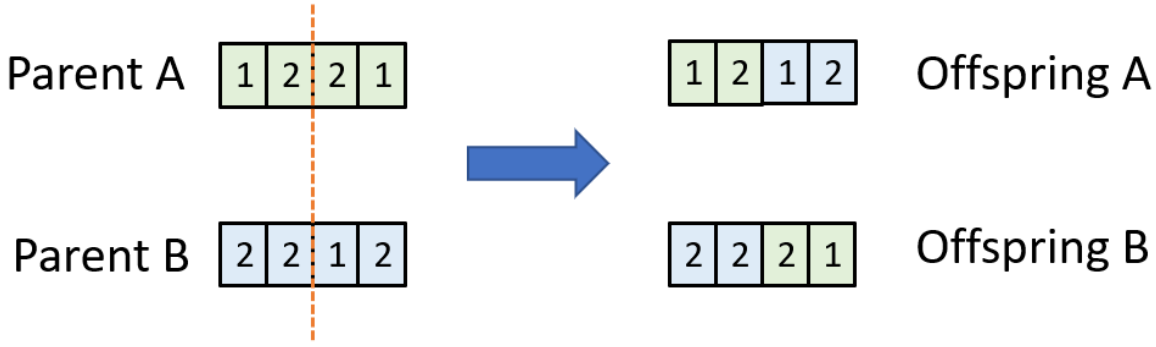


Figure 7.3: Illustration of crossover.

Algorithm 2 Genetic Algorithm for Layer-wise Quantization

Inputs: The population size, N
The selection size, n
The number of generations, $gmax$
Output: The optimal configuration, C

Procedure:

1: $g \leftarrow 0$
2: Population $\leftarrow$ initialize(N)

3: **while** $g < gmax$ **do**
4: Fitness $\leftarrow$ evaluate(Population)
5: Parents $\leftarrow$ select(Population, Fitness, n)
6: Offspring $\leftarrow$ crossover(Parents)
7: Offspring $\leftarrow$ mutation(Offspring)
8: Elite $\leftarrow$ elitism(Parents, Fitness, 2)
9: Population $\leftarrow$ merge(Offspring, Elite)
10: $g \leftarrow g + 1$
11: **end while**

12: Fitness $\leftarrow$ evaluate(Population)
13: $C \leftarrow$ elitism(Population, Fitness, 1)

from the initial population using the roulette wheel approach [149], where each individual's share of the wheel is defined as $p_i = f_i / \sum_i f_i$ to give an advantage to more fit individuals. In Line 6, crossover() selects pairs of individuals as the parents and crosses over their genes to generate a set of N – 2 offspring. As illustrated in Figure 7.3, the genes of Offspring A are identical to those of Parent A, up to a randomly chosen gene position. The remaining genes are then determined by Parent B. Offspring B is determined in a similar way.

Next, Line 7 applies mutation() to each offspring. The mutation function iterates through each gene in the offspring chromosomes; in each iteration, the target gene is replaced with a randomly-selected gene, with a probability of 1%. Line 8 then calls the elitism() function to select the two

most fit members of the current generation and Line 9 defines the next-generation population by combining the $N - 2$ offspring chromosomes determined by crossover() and mutation() with the two chromosomes determined by elitism(). The counter is updated in Line 10 and the loop repeats. After *gmax* iterations, the GA performs a fitness evaluation of the final generation and returns the single most fit chromosome, *C*.

7.4 Simulation Results

Hardware DBN simulations are performed using PIN-Sim to evaluate various layer-wise quantization configurations. A separate Python file, quantizer.py, is used to quantize DBN resistance values generated by mapRBM. By default, mapRBM sets resistance values between 1 and 5kΩ, in intervals of 0.5kΩ. Quantizer modifies these values based on a user-defined input configuration, specifying the number of bits allocated to each layer. A 1-bit layer changes all resistances below 3kΩ to 1kΩ and all other resistances to 5kΩ. A 2-bit layer changes all resistances in the range [1 kΩ, 1.5 kΩ] to 1kΩ, [2kΩ, 3kΩ] to 2kΩ, [3.5kΩ, 4kΩ] to 4kΩ and [4.5kΩ, 5kΩ] to 5kΩ. Hardware simulations are then performed on the modified DBN using PIN-Sim via the testDBN module.

Two network topologies are used: 784×200×200×10 (three-layer) and 784×200×200×200×10 (four-layer), both using a sigmoidal activation function. In each case, all possible configurations using 1-bit and 2-bit layers are evaluated for area, MNIST image classification error and area-error-product. Area is computed using two crossbars for 2-bit layers and a single crossbar for 1-bit layers, in accordance with Figure 7.1. The Default configuration assumes 4 crossbars per layer to implement the PIN-Sim default of 9 resistance levels [123]. Area, error and area-error-product for the three-layer topology are shown by Figure 7.4, Figure 7.5 and Figure 7.6, respectively, and for the four-layer topology by Figure 7.7, Figure 7.8 and Figure 7.9, respectively.

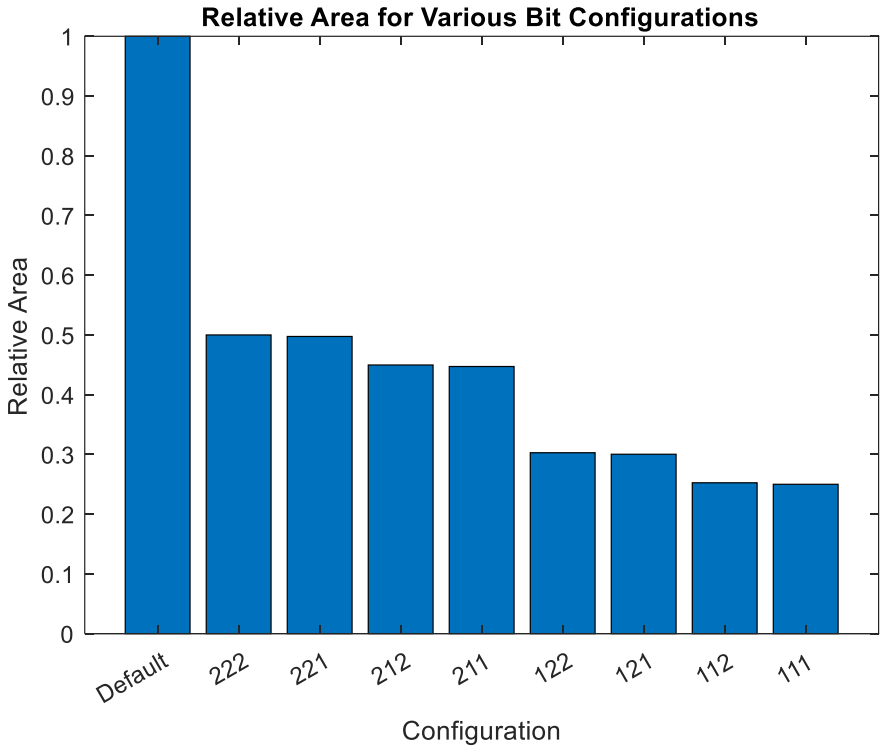


Figure 7.4: Relative area for various layer-wise bit configurations for three-layer topology.

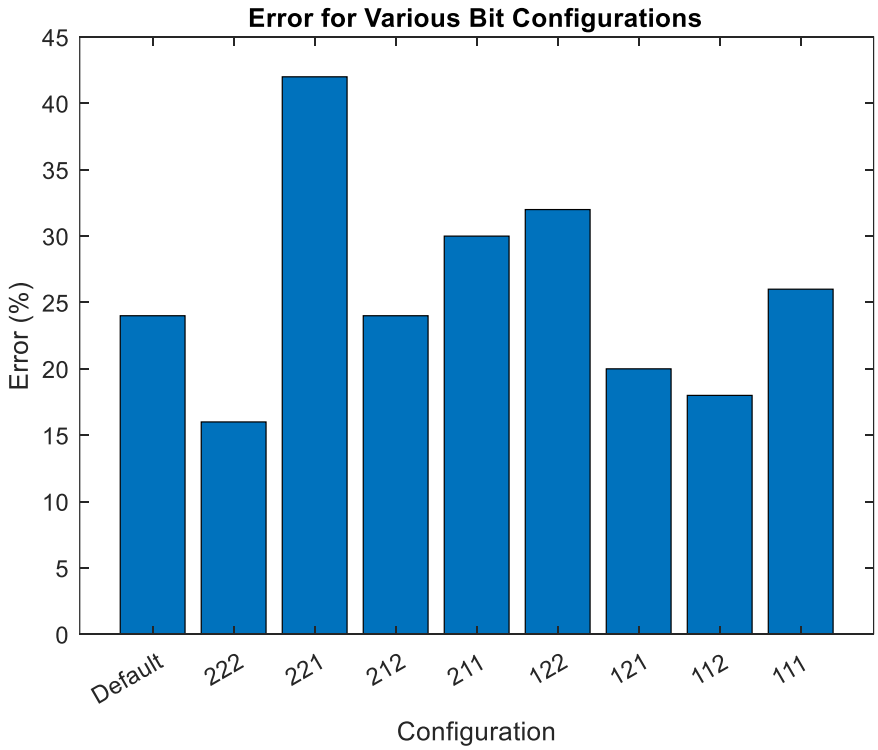


Figure 7.5: Error for various layer-wise bit configurations for three-layer topology.

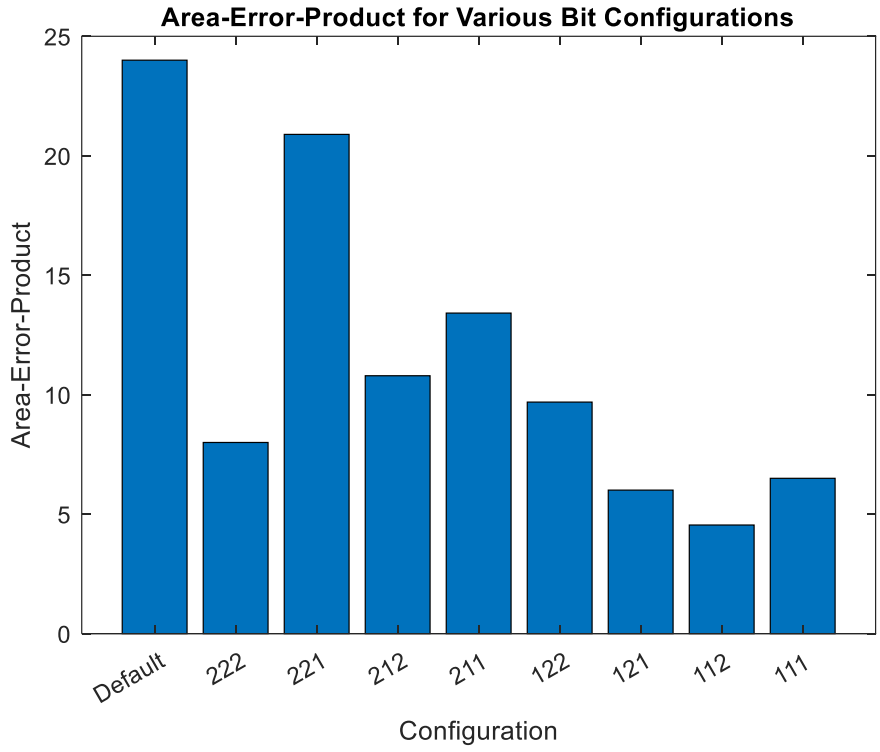


Figure 7.6: Area-error-product for various layer-wise bit configurations for three-layer topology.

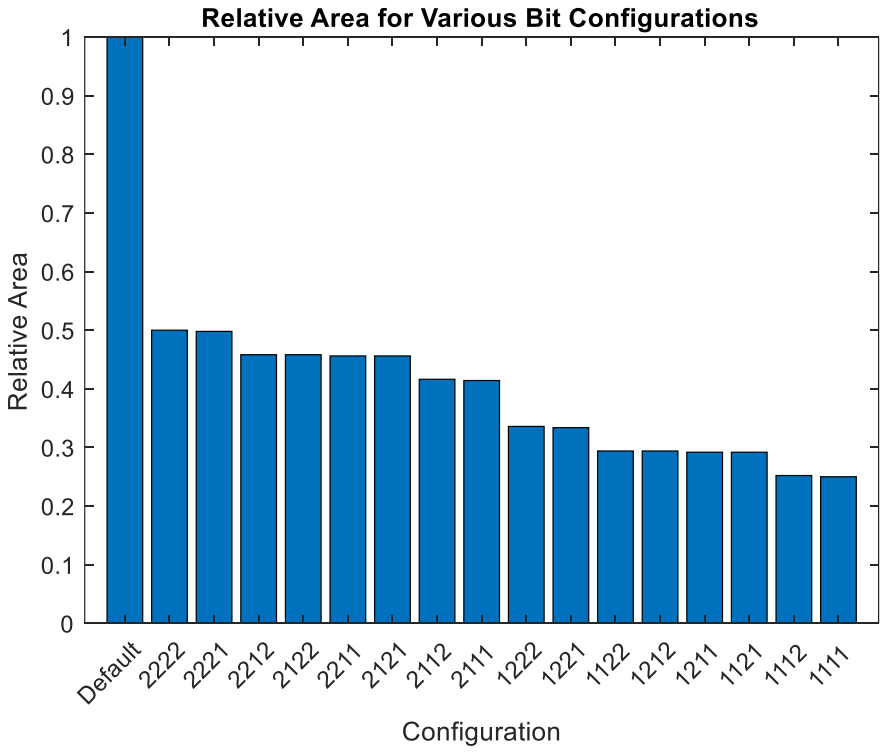


Figure 7.7: Relative area for various layer-wise bit configurations for four-layer topology.

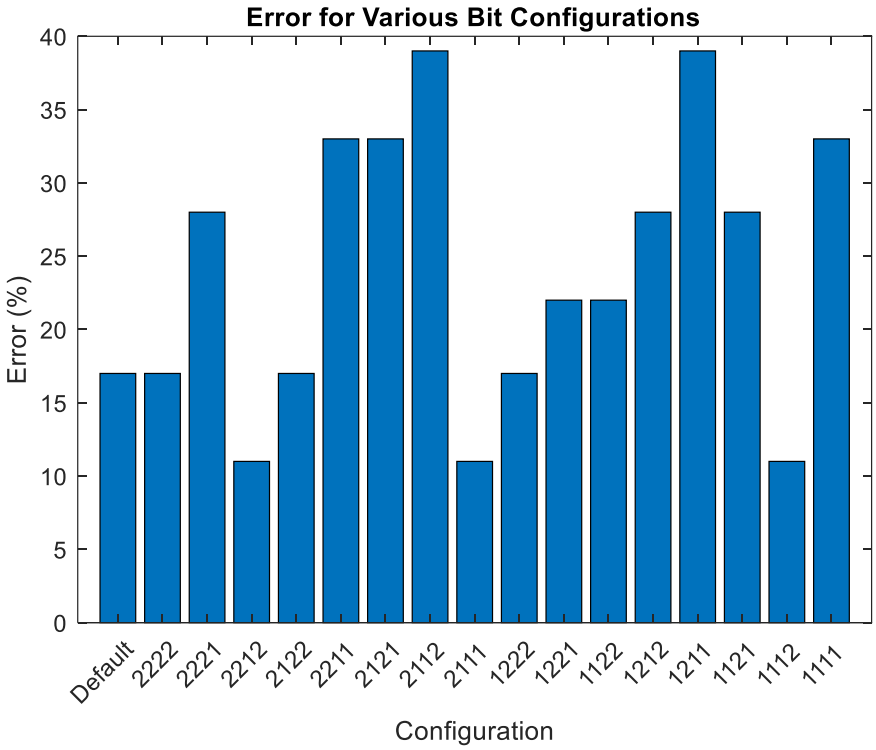


Figure 7.8: Error for various layer-wise bit configurations for four-layer topology.

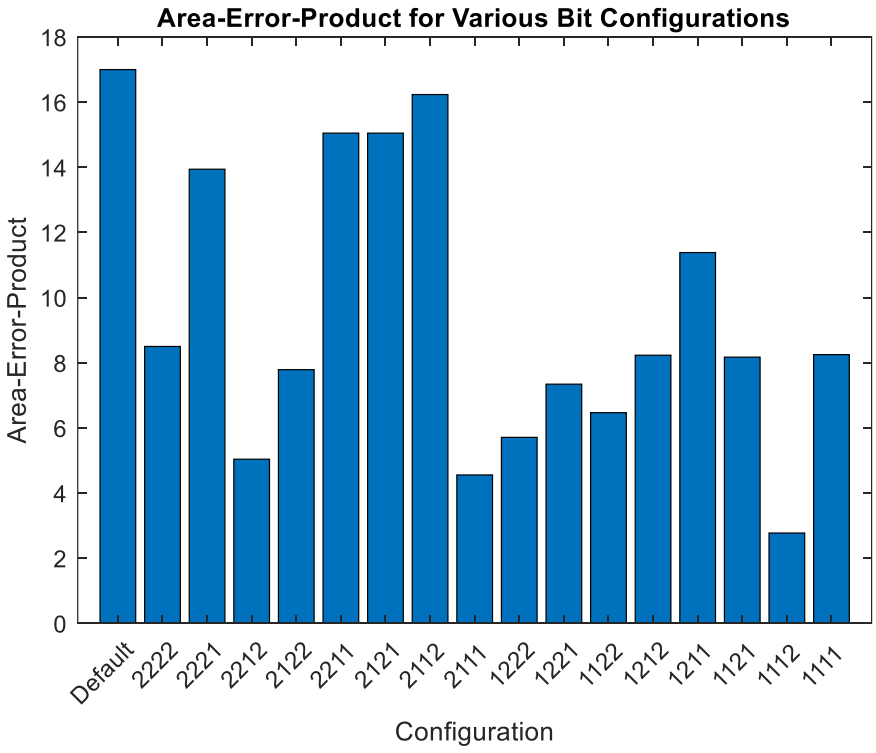


Figure 7.9: Area-error-product for various layer-wise bit configurations for four-layer topology.

In Figure 7.4 – Figure 7.9, configurations are represented in the same way as in the GA approach, i.e., as four-digit strings where the digit in position *i* of the string represents the number of bits allocated to Layer *i*. For example, a configuration of 2121 for the 784×200×200×200×10 (four-layer) topology indicates that 2 bits are allocated to weights in the 784×200 layer as well as the second 200×200 layer, while 1 bit is allocated to each weight in the remaining two layers. Two interesting and unexpected observations may be made based on the data. First, both topologies reveal lower error, in certain cases, for smaller architectures. For both topologies, there are cases where decreasing the precision level of a layer results in improved accuracy. Examples include the 112 vs. 212 configuration in the three-layer topology and the 1112 vs. 2222 configuration in the four-layer topology. Moreover, assigning higher precision levels to deeper levels in the network is generally more favorable than assigning the same precision level to an earlier layer. For the three-layer network, the 122 and 112 configurations perform better than the 221 and 211 configurations, respectively. Moreover, for the four-layer network, the 1222, 1212 and 1122 configurations perform better than the 2221, 2121 and 2211 configurations, respectively. The 1112 and 2111 configurations show equal performance.

Despite the unexpected nature of these observations, they are consistent with earlier DBN results attained using PIN-Sim [123]. While the results in [123] only consider uniform quantization, they confirm that in certain cases, the DBN accuracy can improve after a reduction in weight precision levels. An explanation offered in the literature [150] is that quantization may reduce the level of overfitting within the network parameters. These results demonstrate that the default PIN-Sim precision level is not necessary to maintain accuracy. In the three-layer topology, the 112 configuration offers a 74.7% reduction in area, compared to the Default configuration. Similarly, for the four-layer topology, the 1112 configuration offers a 74.8% area reduction

compared to Default. In both cases, the smaller network delivers a 6% reduction in error. Furthermore, the results demonstrate the need for layer-wise quantization since in the case of both topologies, assigning a 2-bit quantization to only the last layer results in a significant reduction in error at incremental area cost, compared to using a 1-bit uniform quantization.

Results in Figure 7.6 and Figure 7.9 demonstrate significant variability in area-error-product between quantization configurations for both network topologies. The GA approach introduced in the previous section is employed to find the optimal configuration for the four-layer topology, using the inverse of area-error-product as the fitness function. The GA is run using two sets of inputs: $N = 6$, $n = 4$ and $N = 4$, $n = 2$. In both cases, $gmax = 1$ and 100 trials are conducted, with results listed in Table 7.1.

The objective of the GA is to select the optimal configuration, i.e., the 1112 configuration having a fitness value of 0.3606. Results indicate that, using $N = 6$ and $n = 4$, the average fitness of the configuration returned by the GA is 0.3008 and the optimal configuration is selected 68% of the time. Using $N = 4$ and $n = 2$, the performance is lower due to the narrower scope of the algorithm: the average fitness is reduced to 0.2264 and the optimal configuration is selected 45% of the time. These data are in comparison with a total-population average fitness of 0.1381, with only 6.25% of individuals having the optimal fitness value.

Table 7.1: GA Performance in selecting optimal bit configuration for four-layer DBN.

N	***n***	***gmax***	**Average fitness**	**%Optimal**
6	4	1	0.3008	68
4	2	1	0.2264	45
Total Population			0.1381	6.25

7.5 Summary

Herein, we have presented a spin-based analog architecture for adaptive layer-wise quantization of Deep Belief Networks (DBNs). The presented architecture implements 2-bit weights by using an analog adder to combine the outputs of two MRAM-based crossbar arrays. Moreover, weights within layers that do not require the enhanced precision are represented with 1 bit using a single crossbar. This architecture also embeds Computational Analog Blocks (CABs) for efficient computation of enhanced activation functions, which may result in significant performance improvement as demonstrated in Chapter 6. Finally, we have proposed a Genetic Algorithm (GA) approach for optimizing the bit configuration.

Simulation results demonstrate that quantization may yield significant area benefits without diminishing accuracy and may even result in improved accuracy by eliminating errors due to overfitting. Compared to the default PIN-Sim configuration, layer-wise quantization enables a 74.8% reduction in area while simultaneously achieving 6% reduction in error, using the optimal configuration. The optimal configuration is identified by the GA in 68% of trials.

CHAPTER 8: CONCLUSION[6]

8.1 Technical Summary

In this dissertation, a reconfigurable architecture is developed which harnesses intrinsic properties of MRAM-based crossbar arrays together with analog computation for area and energy-efficient implementation of Compressive Sensing (CS) and Deep Belief Networks (DBNs). The proposed architecture is particularly beneficial in IoT sensing applications, where the challenge is processing and transmitting vast quantities of data despite constraints in energy, processing area, memory and bandwidth.

First, we develop an approach to non-uniform CS sampling based on dynamically configured sampling rates using naturally occurring voltage degradation in a crossbar array. This technique embeds some required computations to be conducted intrinsically by the cross-points of the array, thus bypassing overheads of conventional instruction execution and eliminating the need for costly hardware components such as lookup tables and data converters. This architecture is shown to be robust for various array sizes and parasitics and achieves a 583-fold reduction in energy and 23-fold reduction in transistor count compared with the baseline design.

We next demonstrate the Area Conserving Crossbar Leveraging Adaptive Information Mapping (ACCLAIM) architecture as a further means of optimizing signal compression via Discrete Cosine Transform (DCT) and non-uniform CS. ACCLAIM leverages the spectral sparsity of real-world signals to implement an adaptive quantization approach by assigning a variable number of word lines, and hence a variable weight precision, to each input coefficient. Simulation

[6] ©IEEE. Part of this chapter is reprinted, with permission, from [132, 136, 137].

results indicate that at fixed area, ACCLAIM attains an 18dB improvement in reconstruction accuracy for DCT and 9dB for CS, compared to a traditional crossbar approach using uniform quantization. Moreover, at a fixed standard of error, ACCLAIM allows for 70.5% reduction in area and 30.2% reduction in power in CS sampling.

Next, we develop a reconfigurable analog circuit for performing generalized exponentiation within a mixed-signal field programmable array architecture. The resulting analog module consisting of MRAM devices along with FET-based sensing and amplification circuits are circuit-switched-configurable with terminal-level programmable control. By leveraging intrinsic properties of embedded devices, the design is configured to rapidly evaluate various arithmetic operations within acceptable error tolerances for selected applications. When compared to a state-of-the-art approximate digital multiplier, the presented design achieves roughly 95% reduction in area while generating a stable output within a period comparable to single-cycle execution.

This analog computational approach is used as a fundamental building block of the Spintronically-Configurable Adaptive in-memory Processing Environment (SCAPE). SCAPE combines a Vector Matrix Multiplication Stage (VMMS) consisting of MRAM-based crossbar arrays with an Analog Activation Stage (AAS) based on the presented analog computational circuit for applicability to generalized use cases involving dot products in addition to scalar operations. The Approximate Message Passing (AMP) CS reconstruction algorithm is evaluated on SCAPE, demonstrating a 96% reduction in energy with negligible accuracy loss, compared with a recent state-of-the-art design. Moreover, SCAPE allows for efficient and versatile computation of activation functions in DBNs: simulation results demonstrate the possibility of reducing network size while retaining accuracy through such an approach. The benefits of enhanced activation functions are amplified by a Progressive Temporal Modular Redundancy (PTMR) architecture,

which executes multiple DBN trials with varying activation functions and outputs the majority result. PTMR allows for an 81.2% reduction in power at only 4% accuracy loss, compared to a larger network.

Finally, an architecture is presented allowing for layer-wise adaptive quantization in deep neural networks. Similarly to SCAPE, the architecture leverages intrinsic computation by combining MRAM-based crossbars with analog arithmetic. Together with the hardware design, a Genetic Algorithm (GA) is given for determining the optimal layer-wise bit configuration. Simulation results indicate that layer-wise quantization may be applied to significantly reduce the size of a network while simultaneously increasing accuracy by reducing the level of overfitting. Using the optimal bit configuration on a four-layer DBN, a 74.8% reduction in area is attained with a simultaneous 6% accuracy improvement. Moreover, the GA is able to identify the optimal configuration in two out of three trials.

8.2 Future Directions

The key limitation of the PTMR approach is increased power overhead due to repeated computations. PTMR reduces power consumption by stalling during the third computational cycle if identical results are attained in the first two cycles. A further optimization could be realized by predicting the outputs of the second and third cycles and interrupting the computation if the output is predicted to match the previous cycle's result. The problem is defined as follows: given two DBNs having different model parameters but identical inputs, with the outputs after Layer *i* given as $\boldsymbol{y_{1i}}$ and $\boldsymbol{y_{2i}}$ in the two networks, and with the final outputs of the two networks given as $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$, what is the correlation between $\|\boldsymbol{y_{1i}} - \boldsymbol{y_{2i}}\|$ and $\|\boldsymbol{z_1} - \boldsymbol{z_2}\|$? If the correlation is strong enough, then knowing the intermediate outputs of two networks and also the final output of one network allows one to predict the final output of the second network. In that case, one can establish a

confidence level k and thresholds ε and δ such that $\|\boldsymbol{z_1} - \boldsymbol{z_2}\| < \delta$ with a probability of k whenever $\|\boldsymbol{y_{1i}} - \boldsymbol{y_{2i}}\| < \varepsilon$. Thus, redundant computations may be terminated at Layer i whenever the threshold is met, which can save significant power and also reduce delay for larger networks. Determining the parameters δ and k then represents a tradeoff between power consumption and accuracy and is application dependent.

Finally, security is an aspect of IoT sensor design that has been outside the scope of this dissertation but may be addressed in future work. An attacker may significantly reduce the accuracy of a neural network classifier by inserting adversarial noise into the input data [151, 152]. Such an attack may be carried out either during the inference phase or during the training phase; in the latter case, model parameters within the network are modified such that the network performs poorly whenever a backdoor trigger is present within the input. It has been shown that quantization [151] as well as pruning [152] can be used as defenses against adversarial noise. Moreover, it is seen [151] that the level of adversarial noise may significantly affect the network accuracy for a given quantization level. Thus, it is interesting to extend the layer-wise quantization approach introduced in Chapter 7 to a) consider the effect of adversarial noise and b) also incorporate pruning for power reduction and adversarial defense.

APPENDIX: COPYRIGHT PERMISSIONS

A Reconfigurable and Compact Spin-Based Analog Block for Generalizable nth Power and Root Computation

Conference Proceedings: 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)
Author: Adrian Tatulian
Publisher: IEEE
Date: July 2021

LIST OF REFERENCES

[1] Y.-K. Chen, "Challenges and opportunities of internet of things," in *17th Asia and South Pacific design automation conference*, 2012: IEEE, pp. 383-388.

[2] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497-1516, 2012.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[4] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996-2018, 2014.

[5] H. Tan and R. F. DeMara, "A multilayer framework supporting autonomous run-time partial reconfiguration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 5, pp. 504-516, 2008.

[6] J. Huang, M. Parris, J. Lee, and R. F. Demara, "Scalable FPGA-based architecture for DCT computation using dynamic partial reconfiguration," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 9, no. 1, pp. 1-18, 2009.

[7] J. Lohn, G. Larchev, and R. DeMara, "A genetic representation for evolutionary fault recovery in Virtex FPGAs," in *Evolvable Systems: From Biology to Hardware: 5th International Conference, ICES 2003 Trondheim, Norway, March 17–20, 2003 Proceedings 5*, 2003: Springer, pp. 47-56.

[8] R. F. DeMara and K. Zhang, "Autonomous FPGA fault handling through competitive runtime reconfiguration," in *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*, 2005: IEEE, pp. 109-116.

[9] M. G. Parris, C. A. Sharma, and R. F. Demara, "Progress in autonomous fault recovery of field programmable gate arrays," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, pp. 1-30, 2011.

[10] M. Lin, S. Chen, R. F. DeMara, and J. Wawrzynek, "ASTRO: Synthesizing application-specific reconfigurable hardware traces to exploit memory-level parallelism," *Microprocessors and Microsystems*, vol. 39, no. 7, pp. 553-564, 2015.

[11] R. Al-Haddad, R. Oreifej, R. A. Ashraf, and R. F. DeMara, "Sustainable modular adaptive redundancy technique emphasizing partial reconfiguration for reduced power consumption," *International Journal of Reconfigurable Computing*, vol. 2011, 2011.

[12] N. Imran, R. F. DeMara, J. Lee, and J. Huang, "Self-adapting resource escalation for resilient signal processing architectures," *Journal of Signal Processing Systems*, vol. 77, pp. 257-280, 2014.

[13] R. S. Oreifej, R. N. Al-Haddad, H. Tan, and R. F. DeMara, "Layered approach to intrinsic evolvable hardware using direct bitstream manipulation of Virtex II Pro devices," in *2007 International Conference on Field Programmable Logic and Applications*, 2007: IEEE, pp. 299-304.

[14] K. Zhang, G. Bedette, and R. F. DeMara, "Triple modular redundancy with standby (TMRSB) supporting dynamic resource reconfiguration," in *2006 IEEE Autotestcon*, 2006: IEEE, pp. 690-696.

[15] R. A. Ashraf and R. F. DeMara, "Scalable FPGA refurbishment using netlist-driven evolutionary algorithms," *IEEE Transactions on Computers,* vol. 62, no. 8, pp. 1526-1541, 2013.

[16] R. F. DeMara, K. Zhang, and C. A. Sharma, "Autonomic fault-handling and refurbishment using throughput-driven assessment," *Applied Soft Computing,* vol. 11, no. 2, pp. 1588-1599, 2011.

[17] K. Zhang, R. F. DeMara, and C. A. Sharma, "Consensus-based evaluation for fault isolation and on-line evolutionary regeneration," in *Evolvable Systems: From Biology to Hardware: 6th International Conference, ICES 2005, Sitges, Spain, September 12-14, 2005. Proceedings 6*, 2005: Springer, pp. 12-24.

[18] M. Shaban and A. Abdelgawad, "A study of distributed compressive sensing for the Internet of Things (IoT)," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018: IEEE, pp. 173-178.

[19] A. Nisar, S. Dhull, S. Mittal, and B. K. Kaushik, "SOT and STT-based 4-bit MRAM cell for high-density memory applications," *IEEE Transactions on Electron Devices,* vol. 68, no. 9, pp. 4384-4390, 2021.

[20] S. Salehi, A. Zaeemzadeh, A. Tatulian, N. Rahnavard, and R. F. DeMara, "MRAM-based stochastic oscillators for adaptive non-uniform sampling of sparse signals in IoT applications," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019: IEEE, pp. 403-408.

[21] Z. Gao, L. Dai, S. Han, I. Chih-Lin, Z. Wang, and L. Hanzo, "Compressive sensing techniques for next-generation wireless communications," *IEEE Wireless Communications,* vol. 25, no. 3, pp. 144-153, 2018.

[22] Y. Zhang, Y. Xiang, L. Y. Zhang, Y. Rong, and S. Guo, "Secure wireless communications based on compressive sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1093-1111, 2018.

[23] N. Rahnavard, A. Talari, and B. Shahrasbi, "Non-uniform compressive sensing," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011: IEEE, pp. 212-219.

[24] Y. Massoud, F. Xiong, and S. Smaili, "A memristor-based random modulator for compressive sensing systems," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012: IEEE, pp. 2445-2448.

[25] F. Qian, Y. Gong, G. Huang, M. Anwar, and L. Wang, "Exploiting memristors for compressive sampling of sensory signals," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2737-2748, 2018.

[26] N. Khoshavi, R. A. Ashraf, and R. F. DeMara, "Applicability of power-gating strategies for aging mitigation of CMOS logic paths," in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2014: IEEE, pp. 929-932.

[27] A. Tatulian, S. Salehi, and R. F. DeMara, "Mixed-signal spin/charge reconfigurable array for energy-aware compressive signal processing," in *2019 International conference on ReConFigurable computing and FPGAs (ReConFig)*, 2019: IEEE, pp. 1-8.

[28] B. Zhang, N. Uysal, and R. Ewetz, "Computational Restructuring: Rethinking Image Compression Using Resistive Crossbar Arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 5, pp. 836-849, 2020.

[29] N. Imran, R. A. Ashraf, and R. F. DeMara, "Power and quality-aware image processing soft-resilience using online multi-objective GAs," *International Journal of Computational Vision and Robotics,* vol. 5, no. 1, pp. 72-98, 2015.

[30] X. Yan, Y. Fan, K. Chen, X. Yu, and X. Zeng, "Qnet: an adaptive quantization table generator based on convolutional neural network," *IEEE Transactions on Image Processing,* vol. 29, pp. 9654-9664, 2020.

[31] F. Zhai, S. Xiao, and L. Quan, "A new non-uniform quantization method based on distribution of compressive sensing measurements and coefficients discarding," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2013: IEEE, pp. 1-4.

[32] R. N. Strickland, T. Draelos, and Z. Mao, "Edge detection in machine vision using a simple L1 norm template matching algorithm," *Pattern recognition,* vol. 23, no. 5, pp. 411-421, 1990.

[33] Y. Shi, S. Xia, Y. Zhou, and Y. Shi, "Sparse signal processing for massive device connectivity via deep learning," in *2020 IEEE international conference on communications workshops (ICC Workshops)*, 2020: IEEE, pp. 1-6.

[34] X. Yang, Y. Chen, and H. Liang, "Square root based activation function in neural networks," in *2018 International conference on audio, language and image processing (ICALIP)*, 2018: IEEE, pp. 84-89.

[35] M. Sipper, "Neural networks with à la carte selection of activation functions," *SN Computer Science,* vol. 2, no. 6, p. 470, 2021.

[36] A. Hasnat, T. Bhattacharyya, A. Dey, S. Halder, and D. Bhattacharjee, "A fast FPGA based architecture for computation of square root and Inverse Square Root," in *2017 Devices for Integrated Circuit (DevIC)*, 2017: IEEE, pp. 383-387.

[37] H. Jiang, C. Liu, F. Lombardi, and J. Han, "Low-power approximate unsigned multipliers with configurable error recovery," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 189-202, 2018.

[38] N. Arya, T. Soni, M. Pattanaik, and G. Sharma, "Area and energy efficient approximate square rooters for error resilient applications," in *2020 33rd international conference on VLSI design and 2020 19th international conference on embedded systems (VLSID)*, 2020: IEEE, pp. 90-95.

[39] A. J. S. de Sousa *et al.*, "A very compact CMOS analog multiplier for application in CNN synapses," in *2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS)*, 2019: IEEE, pp. 241-244.

[40] R. B. Wunderlich, F. Adil, and P. Hasler, "Floating gate-based field programmable mixed-signal array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1496-1505, 2012.

[41] C. Schlottmann and P. Hasler, "FPAA empowering cooperative analog-digital signal processing," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2012: IEEE, pp. 5301-5304.

[42] Y. Huang, *Hybrid analog-digital co-processing for scientific computation*. Columbia University, 2018.

[43] B. Rumberg and D. W. Graham, "A low-power field-programmable analog array for wireless sensing," in *Sixteenth international symposium on quality electronic design*, 2015: IEEE, pp. 542-546.

[44] D. Moldovan, S. Cha, I. Urn, R. DeMara, and J. Kim, "pp" Direct Memory Access Translation on SNAP,"" Technical Report PKPLab-90-9, Department of Electrical Engineering-Systems …, 1990.

[45] R. F. DeMara and D. I. Moldovan, "The SNAP-1 parallel AI prototype," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 8, pp. 841-854, 1993.

[46] Y. Xu, B. Wu, Z. Wang, Y. Wang, Y. Zhang, and W. Zhao, "Write-efficient STT/SOT hybrid triple-level cell for high-density MRAM," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1460-1465, 2020.

[47] Y. Zhang, L. Zhang, W. Wen, G. Sun, and Y. Chen, "Multi-level cell STT-RAM: Is it realistic or just a dream?," in *Proceedings of the International Conference on Computer-Aided Design*, 2012, pp. 526-532.

[48] S. Miura *et al.*, "Scalability of quad interface p-MTJ for 1X nm STT-MRAM With 10-ns low power write operation, 10 years retention and endurance> 10^{11}," *IEEE Transactions on Electron Devices*, vol. 67, no. 12, pp. 5368-5373, 2020.

[49] S. Verma and B. K. Kaushik, "Low-power high-density STT MRAMs on a 3-D vertical silicon nanowire platform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1371-1376, 2015.

[50] S. Yuasa, A. Fukushima, T. Nagahama, K. Ando, and Y. Suzuki, "High tunnel magnetoresistance at room temperature in fully epitaxial Fe/MgO/Fe tunnel junctions due

to coherent spin-polarized tunneling," *Japanese Journal of Applied Physics*, vol. 43, no. 4B, p. L588, 2004.

[51] S. Matsunaga *et al.*, "Fabrication of a nonvolatile full adder based on logic-in-memory architecture using magnetic tunnel junctions," *Applied Physics Express*, vol. 1, no. 9, p. 091301, 2008.

[52] V. K. Joshi, P. Barla, S. Bhat, and B. K. Kaushik, "From MTJ device to hybrid CMOS/MTJ circuits: A review," *IEEE Access*, vol. 8, pp. 194105-194146, 2020.

[53] L. Zhu *et al.*, "Heterogeneous 3D integration for a RISC-V system with STT-MRAM," *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 51-54, 2020.

[54] K. C. Chun, H. Zhao, J. D. Harms, T.-H. Kim, J.-P. Wang, and C. H. Kim, "A scaling roadmap and performance evaluation of in-plane and perpendicular MTJ based STT-MRAMs for high-density cache memory," *IEEE journal of solid-state circuits*, vol. 48, no. 2, pp. 598-610, 2012.

[55] S. Salehi and R. F. DeMara, "SLIM-ADC: Spin-based logic-in-memory analog to digital converter leveraging she-enabled domain wall motion devices," *Microelectronics Journal*, vol. 81, pp. 137-143, 2018.

[56] R. Zand, A. Roohi, and R. F. DeMara, "Fundamentals, modeling, and application of magnetic tunnel junctions," in *Nanoscale Devices*: CRC Press, 2018, pp. 337-368.

[57] S. Salehi, M. B. Mashhadi, A. Zaeemzadeh, N. Rahnavard, and R. F. DeMara, "Energy-aware adaptive rate and resolution sampling of spectrally sparse signals leveraging VCMA-MTJ devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 679-692, 2018.

[58] W. Kang, Y. Ran, Y. Zhang, W. Lv, and W. Zhao, "Modeling and exploration of the voltage-controlled magnetic anisotropy effect for the next-generation low-power and high-speed MRAM applications," *IEEE Transactions on Nanotechnology,* vol. 16, no. 3, pp. 387-395, 2017.

[59] S. Salehi, D. Fan, and R. F. Demara, "Survey of STT-MRAM cell design strategies: Taxonomy and sense amplifier tradeoffs for resiliency," *ACM Journal on Emerging Technologies in Computing Systems (JETC),* vol. 13, no. 3, pp. 1-16, 2017.

[60] J. G. Simmons, "Electric tunnel effect between dissimilar electrodes separated by a thin insulating film," *Journal of applied physics,* vol. 34, no. 9, pp. 2581-2590, 1963.

[61] W. Brinkman, R. Dynes, and J. Rowell, "Tunneling conductance of asymmetrical barriers," *Journal of applied physics,* vol. 41, no. 5, pp. 1915-1921, 1970.

[62] M. Julliere, "Tunneling between ferromagnetic films," *Physics letters A,* vol. 54, no. 3, pp. 225-226, 1975.

[63] J. C. Slonczewski, "Conductance and exchange coupling of two ferromagnets separated by a tunneling barrier," *Physical Review B,* vol. 39, no. 10, p. 6995, 1989.

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

[65] Y. Zhang *et al.*, "Compact modeling of perpendicular-anisotropy CoFeB/MgO magnetic tunnel junctions," *IEEE transactions on Electron devices,* vol. 59, no. 3, pp. 819-826, 2012.

[66] L. Yuan, S.-H. Liou, and D. Wang, "Temperature dependence of magnetoresistance in magnetic tunnel junctions with different free layer structures," *Physical Review B,* vol. 73, no. 13, p. 134403, 2006.

[67] C. H. Shang, J. Nowak, R. Jansen, and J. S. Moodera, "Temperature dependence of magnetoresistance and surface magnetization in ferromagnetic tunnel junctions," *Physical Review B,* vol. 58, no. 6, p. R2917, 1998.

[68] T. Hagler, R. Kinder, and G. Bayreuther, "Temperature dependence of tunnel magnetoresistance," *Journal of Applied Physics,* vol. 89, no. 11, pp. 7570-7572, 2001.

[69] R. Zand, A. Roohi, and R. F. DeMara, "Energy-efficient and process-variation-resilient write circuit schemes for spin hall effect MRAM device," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2394-2401, 2017.

[70] 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.

[71] S. Datta, "p-Bits for probabilistic computing," in *2019 Device Research Conference (DRC)*, 2019: IEEE, pp. 35-36.

[72] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, pp. 4123-4128, 2019.

[73] T. Cao, C. Liu, Y. Gao, and W. L. Goh, "Parasitic-aware modelling for neural networks implemented with memristor crossbar array," in *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2021: IEEE, pp. 122-126.

[74] C. Xu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *2015 IEEE 21st international symposium on high performance computer architecture (HPCA)*, 2015: IEEE, pp. 476-488.

[75] Y. Li, W. Chen, W. Lu, and R. Jha, "Read challenges in crossbar memories with nanoscale bidirectional diodes and ReRAM devices," *IEEE Transactions on Nanotechnology,* vol. 14, no. 3, pp. 444-451, 2015.

[76] G. Papandroulidakis, I. Vourkas, A. Abusleme, G. C. Sirakoulis, and A. Rubio, "Crossbar-based memristive logic-in-memory architecture," *IEEE transactions on nanotechnology,* vol. 16, no. 3, pp. 491-501, 2017.

[77] P.-Y. Chen and S. Yu, "Compact modeling of RRAM devices and its applications in 1T1R and 1S1R array design," *IEEE Transactions on Electron Devices,* vol. 62, no. 12, pp. 4022-4028, 2015.

[78] M. Shevgoor, N. Muralimanohar, R. Balasubramonian, and Y. Jeon, "Improving memristor memory with sneak current sharing," in *2015 33rd IEEE International conference on computer design (ICCD)*, 2015: IEEE, pp. 549-556.

[79] Y. Zhang *et al.*, "CACF: A novel circuit architecture co-optimization framework for improving performance, reliability and energy of ReRAM-based main memory system," *ACM Transactions on Architecture and Code Optimization (TACO),* vol. 15, no. 2, pp. 1-26, 2018.

[80] Y. Zhang, D. Feng, W. Tong, J. Liu, C. Wang, and J. Xu, "Tiered-ReRAM: A low latency and energy efficient TLC crossbar ReRAM architecture," in *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, 2019: IEEE, pp. 92-102.

[81] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, and J. Henkel, "Approximation-aware multi-level cells STT-RAM cache architecture," in *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2015: IEEE, pp. 79-88.

[82] C. Xu, D. Niu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Understanding the trade-offs in multi-level cell ReRAM memory design," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1-6.

[83] S. Salehi and R. F. DeMara, "Adaptive non-uniform compressive sensing using SOT-MRAM multi-bit precision crossbar arrays," *IEEE Transactions on Nanotechnology,* vol. 20, pp. 224-228, 2021.

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

[85] 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.

[86] M. Zou, Z. Zhu, Y. Cai, J. Zhou, C. Wang, and Y. Wang, "Security enhancement for rram computing system through obfuscating crossbar row connections," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020: IEEE, pp. 466-471.

[87] Y. Cai, T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Low bit-width convolutional neural network on RRAM," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 39, no. 7, pp. 1414-1427, 2019.

[88] Z. Zhu *et al.*, "A configurable multi-precision CNN computing framework based on single bit RRAM," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1-6.

[89] Y.-C. Chen, H. Li, W. Zhang, and R. E. Pino, "The 3-D stacking bipolar RRAM for high density," *IEEE transactions on nanotechnology,* vol. 11, no. 5, pp. 948-956, 2012.

[90] M. N. I. Khan and S. Ghosh, "Multi-bit read and write methodologies for diode-MTJ crossbar array," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, 2020: IEEE, pp. 93-98.

[91] M. F. F. Khan, N. A. Jao, C. Shuai, K. Qiu, M. Mahdavi, and V. Narayanan, "Mixed precision Quantization scheme for re-configurable ReRAM crossbars targeting different energy harvesting scenarios," in *Internet of Things. A Confluence of Many Disciplines: Second IFIP International Cross-Domain Conference, IFIPIoT 2019, Tampa, FL, USA, October 31–November 1, 2019, Revised Selected Papers 2*, 2020: Springer, pp. 197-216.

[92] Y. Shi, Z. Huang, S. Oh, N. Kaslan, J. Song, and D. Kuzum, "Adaptive quantization as a device-algorithm co-design approach to improve the performance of in-memory unsupervised learning with SNNs," *IEEE Transactions on Electron Devices*, vol. 66, no. 4, pp. 1722-1728, 2019.

[93] D. Kwon *et al.*, "Adaptive weight quantization method for nonlinear synaptic devices," *IEEE Transactions on Electron Devices*, vol. 66, no. 1, pp. 395-401, 2018.

[94] S. Abden and E. Azab, "Multilayer perceptron analog hardware implementation using low power operational transconductance amplifier," in *2020 32nd International Conference on Microelectronics (ICM)*, 2020: IEEE, pp. 1-4.

[95] M. T. Abuelma'Atti and A. M. Abuelmaatti, "A new current-mode CMOS analog programmable arbitrary nonlinear function synthesizer," *Microelectronics Journal*, vol. 43, no. 11, pp. 802-808, 2012.

[96] A. Buscarino, C. Corradino, L. Fortuna, M. Frasca, and J. C. Sprott, "Nonideal behavior of analog multipliers for chaos generation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 4, pp. 396-400, 2015.

[97] N. Guo *et al.*, "Energy-efficient hybrid analog/digital approximate computation in continuous time," *IEEE Journal of Solid-State Circuits,* vol. 51, no. 7, pp. 1514-1524, 2016.

[98] R. J. D'Angelo and S. R. Sonkusale, "A time-mode translinear principle for nonlinear analog computation," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 62, no. 9, pp. 2187-2195, 2015.

[99] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on evolutionary computation,* vol. 1, no. 2, pp. 109-128, 1997.

[100] Y. A. Sapargaliyev and T. G. Kalganova, "Open-ended evolution to discover analogue circuits for beyond conventional applications," *Genetic Programming and Evolvable Machines,* vol. 13, pp. 411-443, 2012.

[101] M. J. Streeter, M. A. Keane, and J. R. Koza, "Iterative refinement of computational circuits using genetic programming," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, 2002, pp. 877-884.

[102] S. D. Pyle, V. Thangavel, S. M. Williams, and R. F. DeMara, "Self-Scaling Evolution of analog computation circuits with digital accuracy refinement," in *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2015: IEEE, pp. 1-8.

[103] V. Thangavel, Z.-X. Song, and R. F. DeMara, "Intrinsic evolution of truncated Puiseux series on a mixed-signal field-programmable soc," *IEEE Access,* vol. 4, pp. 2863-2872, 2016.

[104] S. George *et al.*, "A programmable and configurable mixed-mode FPAA SoC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2253-2261, 2016.

[105] Y. Choi, Y. Lee, S.-H. Baek, S.-J. Lee, and J. Kim, "CHIMERA: A field-programmable mixed-signal IC with time-domain configurable analog blocks," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 431-444, 2017.

[106] H. Rabah, A. Amira, B. K. Mohanty, S. Almaadeed, and P. K. Meher, "FPGA implementation of orthogonal matching pursuit for compressive sensing reconstruction," *IEEE Transactions on very large scale integration (VLSI) Systems*, vol. 23, no. 10, pp. 2209-2220, 2014.

[107] S. Salehi, R. Zand, and R. F. DeMara, "Clockless spin-based look-up tables with wide read margin," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019, pp. 363-366.

[108] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21-30, 2008.

[109] H. A. Almurib, T. N. Kumar, and F. Lombardi, "Approximate DCT image compression using inexact computing," *IEEE Transactions on computers*, vol. 67, no. 2, pp. 149-159, 2017.

[110] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Proceedings of 2010 IEEE international symposium on circuits and systems*, 2010: IEEE, pp. 3316-3319.

[111] H. Kung and S. J. Tarsa, "Partitioned compressive sensing with neighbor-weighted decoding," in *2011-MILCOM 2011 Military Communications Conference*, 2011: IEEE, pp. 149-156.

[112] L. Gan, "Block compressed sensing of natural images," in *2007 15th International conference on digital signal processing*, 2007: IEEE, pp. 403-406.

[113] Y. Yu, B. Wang, and L. Zhang, "Saliency-based compressive sampling for image signals," *IEEE signal processing letters*, vol. 17, no. 11, pp. 973-976, 2010.

[114] Y. Shen, W. Hu, R. Rana, and C. T. Chou, "Nonuniform compressive sensing for heterogeneous wireless sensor networks," *IEEE Sensors journal*, vol. 13, no. 6, pp. 2120-2128, 2013.

[115] A. Zaeemzadeh, M. Joneidi, and N. Rahnavard, "Adaptive non-uniform compressive sampling for time-varying signals," in *2017 51st Annual conference on information sciences and systems (CISS)*, 2017: IEEE, pp. 1-6.

[116] N. Karim, A. Zaeemzadeh, and N. Rahnavard, "RL-Ncs: Reinforcement learning based data-driven approach for nonuniform compressed sensing," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019: IEEE, pp. 1-6.

[117] E. C. Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, "A review of sparse recovery algorithms," *IEEE access*, vol. 7, pp. 1300-1322, 2018.

[118] P. Maechler *et al.*, "VLSI design of approximate message passing for signal restoration and compressive sensing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 3, pp. 579-590, 2012.

[119] A. Maleki, "Approximate message passing algorithms for compressed sensing," Stanford University, 2010.

[120] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-speed compressed sensing reconstruction on FPGA using OMP and AMP," in *2012 19th IEEE international conference on electronics, circuits, and systems (ICECS 2012)*, 2012: IEEE, pp. 53-56.

[121] S. Liu, A. Ren, Y. Wang, and P. K. Varshney, "Ultra-fast robust compressive sensing based on memristor crossbars," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017: IEEE, pp. 1133-1137.

[122] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing with approximate message passing using in-memory computing," *IEEE Transactions on Electron Devices*, vol. 65, no. 10, pp. 4304-4312, 2018.

[123] R. Zand, K. Y. Camsari, S. Datta, and R. F. DeMara, "Composable probabilistic inference networks using MRAM-based stochastic neurons," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 2, pp. 1-22, 2019.

[124] 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*, vol. 9, no. 4, pp. 2146-2157, 2020.

[125] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, "Representable matrices: Enabling high accuracy analog computation for inference of DNNs using memristors," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020: IEEE, pp. 538-543.

[126] B. Liu *et al.*, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014: IEEE, pp. 63-70.

[127] Arizona State University, "Predictive Technology Model." Available at: http://ptm.asu.edu

[128] S. Shin *et al.*, "Dynamic reference scheme with improved read voltage margin for compensating cell-position and background-pattern dependencies in pure memristor array," *JSTS: Journal of Semiconductor Technology and Science*, vol. 15, no. 6, pp. 685-694, 2015.

[129] S. Kim, J. Zhou, and W. D. Lu, "Crossbar RRAM arrays: Selector device requirements during write operation," *IEEE Transactions on Electron Devices*, vol. 61, no. 8, pp. 2820-2826, 2014.

[130] F. Juanda, W. Shu, and J. S. Chang, "A 10-GS/s 4-bit single-core digital-to-analog converter for cognitive ultrawidebands," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 1, pp. 16-20, 2016.

[131] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature electronics*, vol. 1, no. 1, pp. 52-59, 2018.

[132] A. Tatulian and R. F. DeMara, "Nonuniform Compressive Sensing via Ohmic Voltage Attenuation: A Memristive Crossbar Design Approach Leveraging Intrinsic Computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 9, pp. 3157-3161, 2021.

[133] H. Kubota *et al.*, "Quantitative measurement of voltage dependence of spin-transfer torque in MgO-based magnetic tunnel junctions," *Nature Physics*, vol. 4, no. 1, pp. 37-41, 2008.

[134] S. Wang, H. Lee, C. Grezes, P. Khalili, K. L. Wang, and P. Gupta, "MTJ variation monitor-assisted adaptive MRAM write," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1-6.

[135] M. Hossain, A. Tatulian, S. Sheikhfaal, H. Thummala, and R. DeMara, "Scalable Reasoning and Sensing Using Processing-In-Memory With Hybrid Spin/CMOS-Based Analog/Digital Blocks," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[136] A. Tatulian and R. F. DeMara, "A Reconfigurable and Compact Spin-Based Analog Block for Generalizable n th Power and Root Computation," in *2021 IEEE computer society annual symposium on VLSI (ISVLSI)*, 2021: IEEE, pp. 302-307.

[137] A. Tatulian and R. F. DeMara, "Generalized Exponentiation Using STT Magnetic Tunnel Junctions: Circuit Design, Performance, and Application to Neural Network Gradient Decay," *SN Computer Science*, vol. 3, no. 2, p. 148, 2022.

[138] K. N. S. Batta and I. Chakrabarti, "VLSI Architecture for Enhanced Approximate Message Passing Algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 3253-3267, 2019.

[139] E. Protas, J. D. Bratti, J. F. Gaya, P. Drews, and S. S. Botelho, "Visualization methods for image transformation convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2231-2243, 2018.

[140] C.-F. Juang, C.-T. Chiou, and C.-L. Lai, "Hierarchical singleton-type recurrent neural fuzzy networks for noisy speech recognition," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 833-843, 2007.

[141] S. Basodi, C. Ji, H. Zhang, and Y. Pan, "Gradient amplification: An efficient way to train deep neural networks," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 196-207, 2020.

[142] B. R. Fernando, Y. Qi, C. Yakopcic, and T. M. Taha, "3D memristor crossbar architecture for a multicore neuromorphic system," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020: IEEE, pp. 1-8.

[143] 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 and Systems,* vol. 36, no. 12, pp. 2134-2138, 2017.

[144] P. Barla, V. K. Joshi, and S. Bhat, "Design and analysis of SHE-assisted STT MTJ/CMOS logic gates," *Journal of Computational Electronics,* vol. 20, no. 5, pp. 1964-1976, 2021.

[145] C. Sakr and N. Shanbhag, "An analytical method to determine minimum per-layer precision of deep neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018: IEEE, pp. 1090-1094.

[146] S.-K. Yeom *et al.*, "Pruning by explaining: A novel criterion for deep neural network pruning," *Pattern Recognition,* vol. 115, p. 107899, 2021.

[147] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "DeepSZ: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proceedings of the 28th international symposium on high-performance parallel and distributed computing*, 2019, pp. 159-170.

[148] Y. Zhou, S.-M. Moosavi-Dezfooli, N.-M. Cheung, and P. Frossard, "Adaptive quantization for deep neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32, no. 1.

[149] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications,* pp. 43-55, 2019.

[150] W. Chen *et al.*, "Quantization of deep neural networks for accurate edge computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 4, pp. 1-11, 2021.

[151] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in *International Conference on Learning Representations*, 2019: International Conference on Learning Representations, ICLR.

[152] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*, 2018: Springer, pp. 273-294.