Circuit Design for Memristor based In-Memory Computing

by

Abhairaj Singh
Electrical Engineering : Microelectronics
Student Number : 4737482
Email : AbhairajSingh@student.tudelft.nl

Course : Masters Thesis
Advisor : Professor Said Hamdioui
Duration : July 1, 2018 - May 20, 2019

This work was performed in:

Computer Engineering Lab
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Circuit Design for Memristor based In-Memory Computing

by Abhairaj Singh

Abstract

Modern computing systems suffer due to inability of CMOS-device technology and conventional Von-Neumann architectures to support today's ever-increasing demand of high performance, reliability, cost and energy-efficiency. While CMOS device suffers from high static leakage, reduced reliability and manufacturing complexity; conventional computing architecture suffers from high power consumption with memory access and performance bottlenecks. Non-volatile and CMOS-compatible emerging memristive device technology with extremely compact memory structures offers in-memory computing solutions. However, research lacks quantitative benchmarking of memristor-based primitive logic designs. Moreover, the arithmetic and functional circuit design solutions are inefficient and hence incompetent to replace the state-of-the-art.

The thesis first covers device level physics of different memristive devices, elaborating their basic structures, working principles and behavioral analyses using Verilog-A models. Building on single device behavioral analyses, a comprehensive exploration and quantitative benchmarking of all existing primitive gates is provided, thereby concluding that scouting logic design technique is the optimal logic gate to perform in-memory computing. Going forward, using scouting logic as the building block, the work presents efficient arithmetic and functional circuit designs that outperform previously proposed in-memory computing solutions and attempts to make a strong case to challenge the current CMOS-based state-of-the-art computing paradigm.

Different flavours of a novel circuit design are proposed to tackle limitations common to circuits implementing primitive arithmetic operations and complex multiply-accumulate (dot-product) operations supporting data-intensive applications. The proposed circuit deploys in-built sample-and-hold and two bit-wise weighting techniques to enable pipelining and self-timing-path to improve accuracy against variations. As compared to *4-bit* adder utilizing integrate and fire circuit (IFC) that is optimized for area/power, the proposed design improves the speed, area, and energy consumption by 4X, 2.5X, and 11X, respectively. Incorporating additional components such as high-gain differential amplifier and modified IFC provides a highly accurate, linear, power efficient dot product engine with significant improvement in memristor endurance. To perform $64_4 \cdot 64_1$, the proposed dot product engine improves the speed, area and energy consumption by 2X, 3.5X and 54X, respectively, as compared to area-efficient IFC-based engine, while also extending the range of operands operated in parallel by $>$ 3X. Compared to highly accurate SAR-ADC(current sense amplifier) based dot product engine, the proposed design improves the speed, area and energy by a factor of 0.4X(1.2X), 200X(6X) and 260X(108X), respectively, with comparable accuracy. Read endurance is significantly improved as $\leq$ 0.1V is maintained across the memristors during the dot-product operation, as opposed to $\geq$ 1V endured using prior proposed designs. To showcase the scalability and versatility of the proposed circuit designs, design prepositions of multi-operand *4-bit* adder, *4*×*4* multiplier and *4-bit* comparator are also presented. Supporting equations, graphs, figures and tables have been included to justify the choices made as part of this work and to enhance the understanding of novel non-volatile memristor based in-memory computing.

Contents

List of Figures

1 Introduction

Significant advances in semiconductor industry is mainly attributed to the development of CMOS technology, since its introduction in the 70's. Downscaling of CMOS geometric sizes with recurring enhancements in instruction level parallelism (ILP) has contributed towards remarkable improvement in performance and cost efficiency. Whereas, downscaling operating voltage and proficient power management techniques enhanced the power/energy efficiency. However, constantly increasing digitization has started to expose limitations of CMOS device technology in computing systems. High static leakage power, stagnant clock frequency, complex and costly manufacturing standards along with deteriorating device reliability are some of the inefficiencies encountered by CMOS device technology. Moreover, Von-Neumann computing architectures are facing serious issues related to saturated ILP and memory bottlenecks.

Novel memristive device technology is considered as a promising candidate to replace/complement CMOS devices due to its non-volatile properties and high scalability. The ability of memristive device to store and compute data at the same physical location offers in-memory computing architectures; a promising prospect to alleviate issues faced by Von-Neumann computing systems. In this manner, this work aims to provide in-memory computing solutions based on novel memristor device technology.

The organization of this chapter is as follows. Section 1.1 puts forth the key motivation. Section 1.2 covers existing in-memory computing solutions at each abstraction level while citing associated research opportunities. Section 1.3 illustrates the topics of research focused by this work. Therefore, it draws attention to related research work and their limitations. Section 1.4 presents the main contributions of this work. Finally, Section 1.5 lays down the organization of the remainder of this thesis report.

1.1 Motivation

Computing systems are classified into abstraction layers [13], as shown Figure 1.1. Device technology lays down the foundation; logic and circuit designs form the building blocks to built hardware architecture; software compiles the hardware architecture to run computing applications. Modern computing systems essentially rely on CMOS-based Von-Neumann architecture to execute a gamut of analog and digital applications [14, 15, 16]. However, emerging exascale data-intensive applications such as medical imaging, neuromorphic computation involving convolutional/deep neural network (CNN/DNN) have exposed performance and power inefficiencies in well-established device technology as well as computing architecture [17, 18, 19].

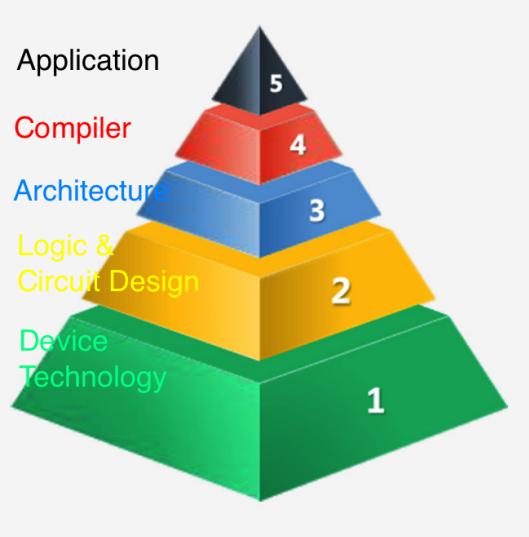


Figure 1.1: Abstraction layers in a computing system.

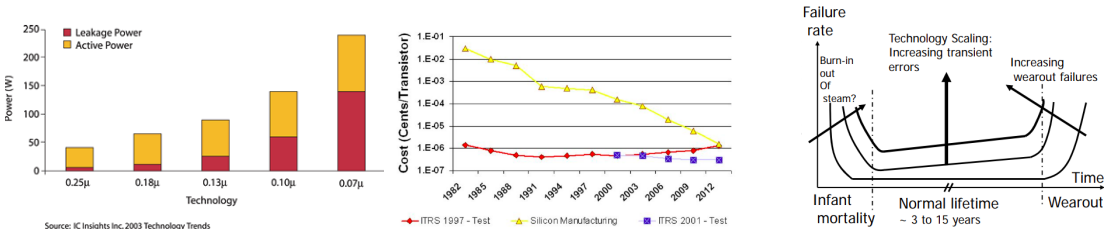
Following are the three well-known walls [20, 21] faced by CMOS device technology.

- **Static Power (Leakage) Wall** - Downscaling operating voltage has been a key technique to keep a check on power density while the density of device components increase $\sim$ 32X per decade [22]. Consequently, similar reduction in CMOS threshold voltage [16] ensures a distinguishable margin between two stable states. However, reducing threshold voltage has drastically increased sub-threshold leakage. Figure 1.2a shows the trend followed by respective static and dynamic power shares with advancing nodes.

- **Cost Wall** - Following Moore's law, recurrent downscaling of CMOS geometric size has offered higher performing and cost-efficient computing systems [22]. Adversely, accompanying complications in the manufacturing process and low yield [19], cost model has taken a hit. Figure 1.2b shows the cost trend with time.

- **Reliability Wall** - With reduction in device sizes, resulting variations and souring manufacturing failure rates have drastically reduced reliability of CMOS devices

[23]. Figure 1.2c displays well-known bathtub diagram to show detrimental failure rates with downscaling.



(a) Leakage wall [4]. (b) Cost wall [4]. (c) Reliability wall [24].

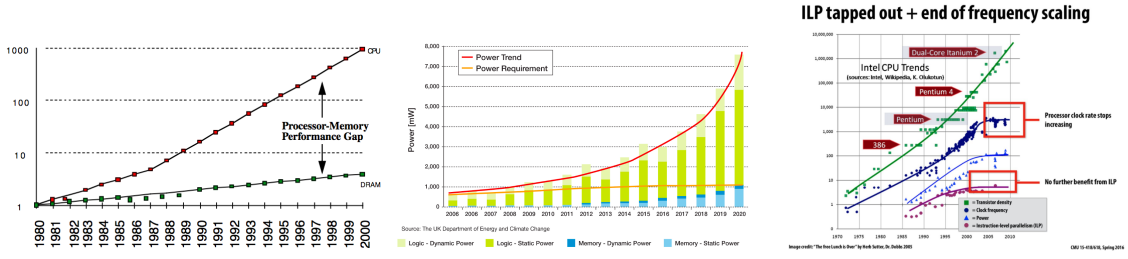
Figure 1.2: Well known CMOS technology walls.

To make matters worse, Von-Neumann architecture also faces another set of three well-known walls [17, 18, 19], which are elaborated as follows.

- **Memory Wall** - Data-intensive applications essentially require large data transfer between storage and processing units. However, the performance gap between processor throughput and memory bandwidth is exponentially increasing as shown in Figure 1.3a, thereby reducing overall performance [21, 25].

- **Power Wall** - Reduced performance is not the only adverse effect of data communication mentioned above. The to-and-fro data transfer consume huge amount of dynamic power [26]. Furthermore, the interconnects have more capacitance due to compact structures, adding to tremendous power density. Figure 1.3b shows the power trend over the years.

- **Instruction Level Parallelism (ILP) Wall** - Parallel processing offered by multi-thread and multi-core systems have improved overall processing throughput of computing systems [13, 17, 18, 19, 21, 25]. However, as shown in Figure 1.3c, recent saturation of operating frequency and ILP has resulted in performance stagnation.



(a) Memory wall [27]. (b) Power wall [4]. (c) ILP wall [28].

Figure 1.3: Well known computing architecture walls.

Non-conventional architectures attempt to bring memory closer to the processing unit, thereby moderating the limitations suffered due to memory bottlenecks. These architectures include multi-level caches (L1, L2, L3 caches) [29, 30], processor-in-memory i.e processing in SRAMs [31, 32], memory accelerators *i.e.* DaDianNao, RENO accelerators [33, 34]. However, recently proposed CIM and ISAAC architectures, among many others [25, 11] have demonstrated the potential of memristor based computing solutions to surpass these non-conventional architectures. Non-volatile memristor devices support densely compact CMOS-compatible memory structures with the potential to support data storage and processing at the same physical location *i.e.* in-memory computing. Evidently, memristive devices not only resolve the issues faced by CMOS technology but also offers in-memory computing to outperform Von-Neumann architecture. Some of the data-intensive applications targeted by memristor-based in-memory computing architectures are neuromorphic computing, realization of machine learning algorithms (CNN/DNN) [11, 35], face/pattern recognition [36] and physical unclonable function (PUF) [37, 38] for Cyber Security.

1.2 Memristor-based In-Memory Computing : Solutions, Limitations and Research Opportunities

This section provides an overview of contemporary memristor-based in-memory computing solutions, associated limitations and new research opportunities in the topics of research, as shown in Table 1.1. The table also highlights topics of research aimed by this work, albeit this section attempts to provide recent progress corresponding the entire hardware design scope.

In-Memory Computing : Hardware Perspective			
Device Physics	Primitive Logic Design	Circuit Design	Architecture

Table 1.1: Hardware topics of research.

1.2.1 Device Physics

Memristor-based memory structures lay the foundation for in-memory computing paradigm. Well-known memristive memory structures include conductive bridge RAM (CB-RAM) [39], phase change memories (PCM) [40], spin-transfer-torque magnetic (STT-MRAM) [41] and redox-oxide RAM (ReRAM) [42]. Prime features of memristive devices explored by such structures are non-volatility, compatibility with CMOS technology, high scalability and the ability to store, process and compute data at the same physical location. Although, their working principles are different and shall be discussed briefly in Chapter 2; but conceptually, these devices share the same idea *i.e.* quantifiable stable resistive states. Therefore, any primitive logic and circuit design solution is essentially applicable to all memristive device technologies and memory structures. Figure 1.4 compares *32-bit* realization by different memristive devices.

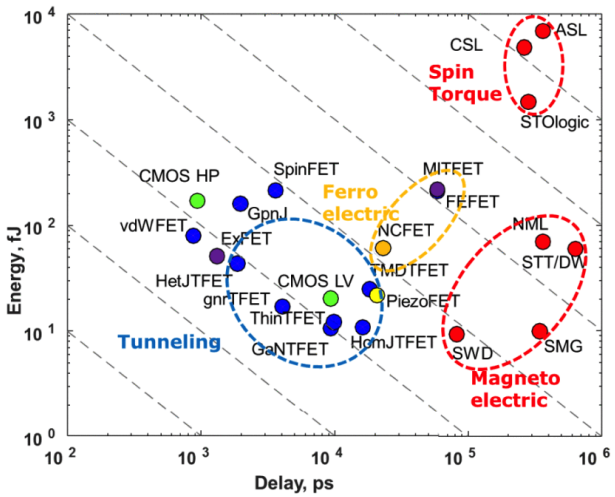


Figure 1.4: *32-bit* realization of ALU : Energy vs delay comparison [1].

However, memristive device technology is still under development phase and is facing severe challenges [41, 43]. Low read and write endurance, high device switching latency and inherent device variability are some of the major drawbacks. Research lacks stable and well-established manufacturing units to extract accurate device models for circuit simulations. Improvement in device features is expected in near future, with promising results by [44, 45, 46].

1.2.2 Primitive Logic Design

Hardware realization of any arithmetic/logic unit primarily consists of primitive logic gates/designs. Several logic design styles based on memristive devices perform commonly utilized logic gates such as NAND, NOR, NOT, OR, IMPLY and majority functions [47, 48, 49, 50, 51]. Since crossbar memory structure provides maximum packing density, only primitive logic designs that support crossbar structures are generally considered.

Correct functionality of these primitive gates is of prime importance but is not the only factor for choosing a certain design type. Efficacy related to speed, peak power, overall energy consumption, area utilization and reliability against device variations for better yield of primitive gates are extremely critical. Unfortunately, research lacks comprehensive quantitative benchmarking of existing primitive gate designs.

1.2.3 Arithmetic/Functional Circuit Design

All processing hardware units are essentially broken down to arithmetic and functional units. Addition, subtraction, multiplication and division are generally known as primitive arithmetic units. Several memristor-based primitive arithmetic units are *1-bit* full adder circuits [47, 49], multi-bit adder circuits [52], multiplers [53] and dividers [54]. Memristor-based fundamental functional units proposed are multiplexers [55], digital comparators [56]. Several dot product engines for neuromorphic applications are pro-

posed, making utmost use of the memristor-based highly dense crossbar memory structure. Various dot-product engines are proposed in [11, 12].

However, these arithmetic and functional units require CMOS technology to build peripheral sensing and control circuitry. Hence, power consumption and area utilization is still significantly dominated by CMOS-based circuits. Following memristor-based analog computing involved in dot-product design engines suffer from the above issues. Peripheral SAR-ADC in In-Situ Analog Arithmetic in Crossbars (ISAAC) architecture [11] accounts for 58% power and 31% area of the chip, integrate and fire circuit (IFC) [12] suffers from reduced scalability. Although, design proposals such as modified current sense amplifier (CSA) circuit [10] solves linearity albeit suffer from large area utilization per row.

1.2.4 Computing Architecture and Applications

As discussed in the Section 1.1, data-intensive applications have exposed the performance and energy consumption bottlenecks of modern day CMOS-based computing systems. In this regard, memristor-based architecture solutions targeted at large scale multi-layered neural networks have been provided. Best known architectural solutions for CNN/DNN algorithms are DaDianNao [34], ISAAC [11], PRIME [57] and integral neuromorphic architectures [58].

However, due to the limitations in arithmetic and functional computational units, computing architectures shall go hand-in-hand with improved versions of arithmetic and functional circuit designs.

1.3 Topics of Research

The topic of research should cover the limitations mentioned in the Section 1.2. This work aims to cover the following topics of research.

1.3.1 Primitive Logic Design

Primarily, the proposed logic designs or gates in research are compared qualitatively and/or theoretically [47]. Consequently, the literature reviews of these gates available mostly cover restrictions and specifications of using a particular gate. Therefore, a quantitative primitive gate benchmarking with accurate performance, area utilization, energy consumption and reliability metrics including control circuitry is indeed need of the hour. These analyses would provide a solid foundation to build simple as well as complex storage and processing units. This work aims to perform such analyses.

1.3.2 Arithmetic Circuit Design

Since the primitive gate logic designs are not quantitatively compared, numerous arithmetic circuit design proposals based on different building blocks *i.e.* primitive logic designs can be found in today's research. Critical benchmarking would provide best/optimal logic design to build processing units which can thereafter be compared for per-

formance, area utilization, energy consumption, reliability and scalability to large computational units.

Since data-intensive applications expose the bottlenecks of utilizing CMOS-based conventional computing paradigm, machine learning algorithm based neural network applications are generally targeted by designers to justify the prospect of memristor-based in-memory computing solutions. However, as mentioned in Section 1.2, these solutions are highly inefficient in terms of speed, area utilization, energy consumption and accuracy of the arithmetic functions performed. This work aims to provide efficient circuit design solutions to build data-intensive processing units.

1.4 Thesis Contributions

Following are the detailed contributions corresponding to the research topics mentioned in Section 1.3.

1.4.1 Quantitative Benchmarking of Primitive Logic Designs

All well-known primitive logic designs are explored to find the optimal gate design for building arithmetic/functional units. Primitive gates are implemented using analytical Verilog-A models and 90nm BSIM CMOS device technology. The main contributions are as follows.

- This work is first-of-its-kind which presents **quantitative benchmarking all primitive logic designs** that support memristor based crossbar structure.
- Comprehensive benchmarking compares latency, area and energy consumption per operation performed for each logic design, which yields the most efficient gate design. Respective shmoo plots include memristive device and voltage variations are also presented.

1.4.2 Proposal of Novel Arithmetic Circuit Designs

This work presents efficient arithmetic/functional circuit design solutions to support memristor-based in-memory computing. In this regard, the main contributions are as follows.

- A novel circuit design is presented to perform *4-bit* multi-operand addition. Key features of this design are
  1. **Built-in sample-and-hold *(S+H)* circuit** - Enables pipelining which improves processing throughput. Additionally, *S+H* circuit allows isolation of peripheral sensing circuit and memristor crossbar. Therefore, read supply voltage is disconnected once sensing circuit begins the evaluation, resulting in significant improvement in energy-efficiency.
  2. **Build-in bit-wise weighting technique** - Allows several rows to share an analog-to-digital converter (ADC). This technique also eliminates the need of

post-processing shift-and-add *(S+A)* digital circuits, required to accumulate digital outputs from different crossbar columns (representing bit-positions). Not to mention CMOS-based registers to store intermediate digital sum and carry outputs, along with extra control circuitry to sequentially perform shifting and addition. This technique thereby provides a high performing, area-efficient solution.

3. **Self-Timing-Path circuit** - Enables adaptive total evaluation time, essential to provide accurate analog-based computing units. Hence, digital output of arithmetic/functional operations are virtually effectively immune to process, voltage, temperature (PVT) and memristor device variations.

Therefore, the proposed adder design offers high performing, variation-aware, area and power efficient solution.

- A modified adder circuit design is presented to perform data-intensive dot-products *i.e.* multiply-and-accumulate operations for vector-matrix multiplications. Besides the features introduced in the proposed adder circuit, *additional* key feature of this design is as follows.

  1. **High-gain voltage differential amplifier (VDA) circuit** - Provides a highly linear, accurate and scalable dot product engine. This circuit aims to stabilize bitline nodal voltage (V_{BL}) which is essential to improve the accuracy of the dot-product engine. It also allows working with a small voltage difference across the memristive devices during the operation, thus providing a highly energy-efficient solution with improved read endurance and reliability. Since the stabilization of V_{BL} is not in the critical path, size of the DA is kept relatively small (inexpensive additional area overhead).
  2. **PMOS-based DA circuit** - Replaces the original NMOS-based DA to provide a highly linear and accurate dot product engine.
  3. **Additional build-in bit-wise weighting technique** - Provides a highly power-efficient dot product design to perform in-built shift-and-add operation. This circuit, along with the high-gain DA ensures working with small voltage applied across memristors.

1.5 Organization

The remainder of this thesis report is organized as follows. The report is divided into chapters which covers the following.

Chapter 2 gives a brief background of novel memristor device technology. It illustrates basic theory, structure and working principle of three well-known memristor devices. It also provides a brief tabulation of potential applications targeted by memristive technology. Thereafter, this chapter presents a detailed exploration of two broad classes of memristor device models to choose a model for further analyses. Finally, it demonstrates behavior of a memristor device under device and voltage variations using the chosen model.

Chapter 3 begins with the classification of existing primitive logic designs supporting memristor-based crossbar memory structures. Thereafter, it illustrates the structure and working principle of these primitive logic designs in detail. Subsequently, it demonstrates quantitative benchmarking to provide the optimal logic design for building arithmetic/-functional units.

Chapter 4 first illustrates existing primitive adder circuits, while citing their design inefficiencies. To improve these inefficiencies, the chapter presents a novel adder circuit design, illustrating its key design features in detail and associated improvements qualitatively. Thereafter, it provides results and comparison of the proposed *4-bit* adder circuit design. To showcase scalability of the proposed circuit design, it also proposes multi-operand *4-bit* adder.

Chapter 5 begins by introducing data-intensive neural networks from a hardware perspective, emphasizing on the implementation of dot product engines which forms the building block of such applications. Following, it illustrates existing dot product engines, while pointing out their design inefficiencies. Thereafter, it presents a modified version of the previously proposed adder circuit design to perform dot product operation, illustrating its key design features in detail and associated improvements qualitatively. Subsequently, it provides results and comparison of the proposed dot product engine. To showcase versatility of the proposed circuit design, it also proposes circuit implementation of a *4*×*4* multiplier and a *4-bit* comparator.

Chapter 6 concludes the research work with future research directions.

Novel Non-Volatile Memristive Devices

2

Memristor, the forth fundamental two-terminal element other than resistor, capacitor and inductor, was first described by Leon Chua in 1971. Regardless of the discovery, memristive device technology underwent a long period of silence due to strong dominance of CMOS technology in computing systems. However, since the successful demonstration of TiO_2-based memristor device by HP labs in 2008, amid recent stagnation faced by CMOS technology, memristive device technology is considered as a potential to replace/-complement CMOS technology. Some of the attractive attributes offered by memristive device technology are non-volatility, high scalability and good compatibility with CMOS technology.

The shared characteristics of the family of memristive devices include the property to switch between stable resistance states and retain the state even without any voltage supply (non-volatility). Some of the well-known memristor-based memories are phase change memories (PCM), spin-torque transfer magnetic RAM (STT-MRAM), conductive bridge RAM (CB-RAM), redox oxide based RAM (ReRAM).

The organization of this chapter is as follows. Section 2.1 delineates memristive device theory, discusses the structure and working principle of three most promising memristive materials/devices, while highlighting their prospect in computing systems. Section 2.2 classifies different memristor device models available in research. Two device models are illustrated extensively, covering the premise of the two broad classes. Detailed discussion substantiates the selection of a device model to perform further analyses. Section 2.3 demonstrates the effects of device and voltage variation on device characteristics using the selected model.

2.1 Basic Theory and Working Principle of Memristive Devices

Chua *et al.* first described memristive device [59] as the fourth *missing* element when combining the fundamental physical quantities in electromagnetism *i.e.* current (I), charge (Q), flux (ϕ), voltage (V) and time (t). Figure 2.1 shows the relation between each pair formed from these quantities. Out of the possible six combinations, three elements were already known prior to his work; *i.e.* capacitor, inductor and resistor and two are based on well-established Ampere and Lens law. The relation between flux (ϕ) and charge (Q) was identified as the missing element as pointed out in Figure 2.1. This implies that this missing element keeps track of not only the amount and direction but also the history of charge flown through that element, as described by Equation 2.1 and 2.2. A memristive device is interchangeably denoted by memristors, memory resistors, memresistors.

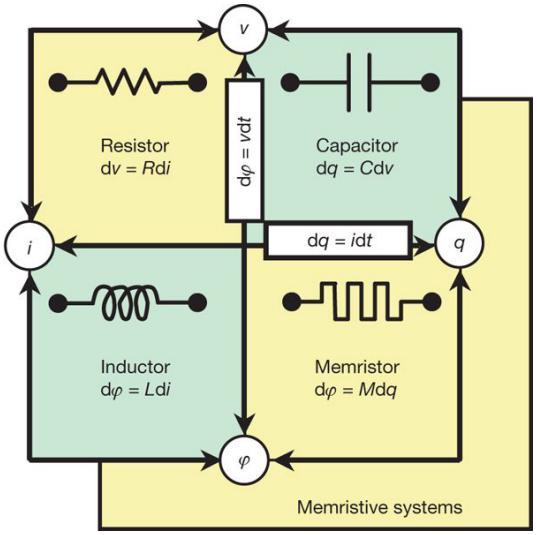


Figure 2.1: Memristor : The missing element [2].

$$\frac{dS}{dt} = f(S, i) \tag{2.1}$$

$$v(t) = M(S, i) \times i(t) \tag{2.2}$$

where S is the internal state variable, $i(t)$ is the memristive device current, $v(t)$ is the memristive device voltage, $M(S, i)$ is the memristance, and t is time. Well-known PCM, STT-MRAM and ReRAM memristor-based memory structures are built using three different classes of memristor devices. Structure and working principle of these memristive devices are discussed briefly, followed by the applications targeted by memristor device technology.

2.1.1 Structure, Property and Working Principle

Memristor-based memory structure is a crossbar network of metal electrodes, with a memristor device placed at all such (perpendicular) cross-sections. Figure 2.2 depicts a general representation of memristor crossbar structure. This arrangement ensures minimum possible feature size of $4F^2$, thus provides maximum packing density. Following memristive devices essentially share the same crossbar structure and are illustrated as follows.

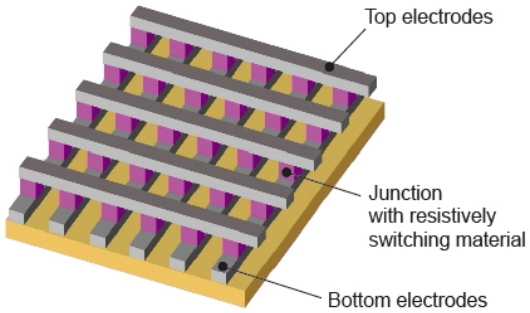


Figure 2.2: Memristor-based crossbar memory array. Top and bottom metal electrodes (grey) sandwich the memristive element (purple) [3].

- **STT-MRAM** - Spin-transfer-torque magnetic device is a memristive device which switches between stable resistive states based on whether the polarized pair formed by magnetic elements is parallel or anti-parallel [23]. The device is based on magnetic torque switching due to electron spin under the influence of electric field. Recently, MgO-based spin-transfer (spintronic) device, also known as magnetic tunnel junction (MJT) device, is proposed in [60] due to its very high tunneling magneto-resistance (TMR) ratio *i.e.* easily distinguishable resistance states. The working principle of MgO-based MJT device is demonstrated in Figure 2.3. MJT consists of 2 dielectric materials, one with fixed polarity and the other with free polarity of magnetic moment. Subsequent paragraph illustrates the working principle of MgO-based MJT device.

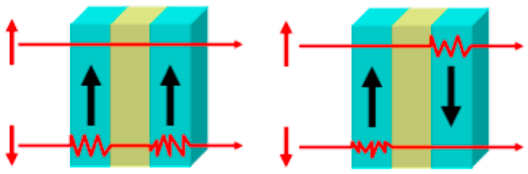


Figure 2.3: Spin-transfer torque magnetic device [4]. Ron state (left) and $Roff$ states.

The amplitude, duration and direction of current applied to write a MJT-based memristor instigates a change in angular momentum of the electrons, which tend to change the magnetization of the free layer. A change of magnetization in the same direction as fixed layer allows high current *i.e.* low (Ron) resistive state whereas

change in the opposite direction allows low current through it *i.e.* high ($Roff$) resistive state. Therefore, as shown in Figure 2.4, the current flowing through MJT at any voltage is not only dependent on the voltage applied, but also on the past history of the applied current *i.e.* the cumulative charge flown through the device in the past.

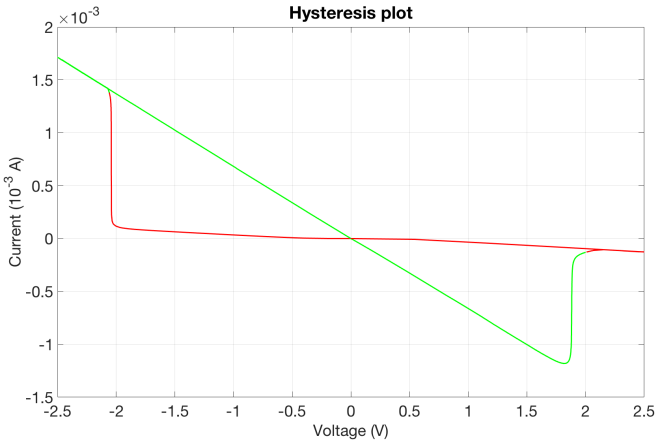


Figure 2.4: Hysteresis plot of memristive devices.

- **Redox Oxide (ReRAM)** - Redox Oxide is a memristive device which switches between stable resistive states based on the deficiency/excess of oxygen ions. TiO_2, HfO_2, MoS_2/MoO_2 [61] are most widely used materials to develop oxide-based memristive devices. The structure of a TiO_2-based memristor is shown in Figure 2.5. The structure basically comprises of top and bottom metal electrodes with TiO_2 as a memristive material. The presence and absence the valence O_{2-x} ions (which acts as a conductive filament), as shown in Figure 2.5, defines the low (Ron) and high ($Roff$) resistance state, respectively.

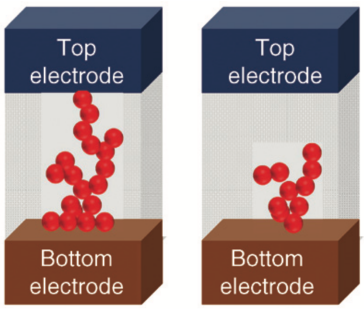


Figure 2.5: Redox-oxide based memristor device [5]. ON (left) and OFF state.

As shown in Figure 2.4, amplitude, duration and direction of voltage applied to write a memristor defines the resistive state. Therefore, similar to MgO-based MJT device, the current flowing through a redox oxide memristor at any voltage is not only dependent on the voltage applied, but also on the past history of the applied

voltage. Section 2.2 presents a detailed demonstration of TiO_2-based memristor device using analytical and experimental Verilog-A device models.

- **PCM** - Phase change device is a memristive device which on application of joule heating, switches its physical form (or stable resistive state) between crystalline or amorphous [6]. Joule heating required for switching is applied in form of electric field or voltage. Some of the recently proposed phase change devices are chalcogenide glass-based GeSbTe, Sb_2Ti_3 AgInSbTe. A SbTe-based phase change device is shown in Figure 2.6. Subsequent paragraph illustrates the working principle of a SbTe-based phase change device.

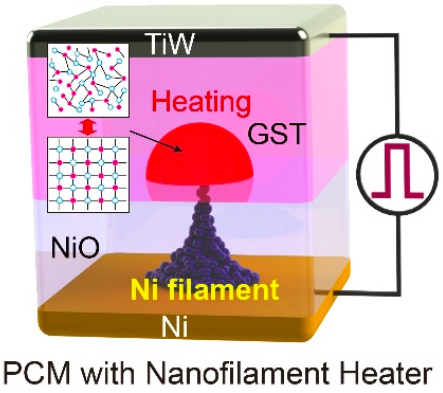


Figure 2.6: Phase change memristive device [6].

SbTe switches to amorphous form by heating the filament above its melting point and then rapidly cooling to room temperature. It switches to crystallize form by heating the filament at a specific temperature *i.e.* between its critical temperature and melting point for a fixed period. Crystalline and amorphous form corresponds to low resistance (Ron) and high resistance ($Roff$) state, respectively. It is due to the fact that crystalline form exhibits a definite path for the flow of electric charge (electrons) whereas amorphous form disrupts this path.

Read operation for memristive devices - Generally, read operation is performed by applying a small voltage across the memristor device and subsequently sensing the integrated current flown through it for a fixed interval of time. The sensing circuit differentiates between the current allowed by ON (I_{ON}) and OFF (I_{OFF}) state, where $I_{ON} \sim \frac{Roff}{Ron} I_{OFF}$.

2.1.2 Prospects of Memristive Device Technology

Memristive devices offer various potential applications due to its non-volatility, compatibility with well-established CMOS technology and high packing density and scalability. Prospective applications of memristor technology are classified, as shown in Table 2.1.

Several in-memory computing architectures based on highly dense memristor crossbar structures target a multitude of data-intensive applications. Physically unclonable

Memristor Applications			
Crossbar Array		Discrete	
Analog	Digital	Analog	Digital
PUF	TCAM	Chaos Circuit	Logic Circuits
Neuromorphic Applications	PCM	Schmitt Trigger	Multiplexers
STDP	ReRAM	Oscillator	Arithmetic Circuits
	STT-MRAM	V/I Amplifier	

Table 2.1: Potential applications of memristive device technology.

function (PUF) [37, 38], spike timing-dependent learning (STDP) [62] are some of the potential applications proposed recently. Various memristor-based dot product engines for neuromorphic applications are proposed in [11, 12, 10]. Memristor-based memory structures such as ternary content addressable memory (TCAM), ReRAM, STT-MRAM and PCM possess highly dense and energy-efficient data storing capabilities to replace conventional non-volatile flash memory and DRAM [63].

Discrete analog circuits such as chaos circuit [64], schmitt trigger [65], difference comparator [66], variable gain amplifier and oscillator [67] show promising results. Several well-known primitive digital logic designs are presented in [54, 48]. Memristor-based primitive arithmetic units include *1-bit* full adders [47, 49], multi-bit adder [52], multiplers [53] and dividers [54]. Memristor-based fundamental functional units proposed are multiplexers [55] and digital comparators [56].

2.2 Existing Device Models

Device modeling is an integral task to validate the promise of novel memristor device technology. Verilog-A or SPICE equivalent memristor device models are used to simulate circuits to evaluate their functionality and design efficiency. Although memristor technology is still under development phase, various research groups have successfully manufactured memristors to provide experimentally accurate device characterization and modelling [2, 68, 69, 70]. Alternatively, proposed analytical models [71, 72, 73] provide a more scientific approach *i.e.* physics driven models illustrating the behaviour of a memristor device with supporting scientific equations governed by physical phenomena. Classification of well-known device models used by circuit designers is shown in Table 2.2. Detailed illustration of two class of models is as follows.

2.2.1 Physics-Driven Models

Physics-driven or analytical or scientific models are driven by scientific equations involving physical device parameters, universal constants and laws of physics illustrating the phenomena. Aachen verilog-A device model [71], along with Stanford and CalTech spice

Memristor Device Models	
Physics-Driven	**Behavioural**
Aachen model	Linear model
Stanford model	Non-linear model
CalTech model	STB model
	VTEAM/TEAM

Table 2.2: Classification of memristive device models.

models [72, 73] are identified as the most popular scientific models used for simulating the behaviour of a memristor device, validating logic gate designs and evaluating circuit designs based on memristor devices. Most of the scientific models share the same principle and are supported by similar equations, therefore Aachen model is chosen to illustrate this classification of models.

- **Aachen Verilog-A Model.**

The physical structure of a TiO_2-based memristor device described by Aachen model [71] is shown in Figure 2.7. At every crossbar intersection of a memristive memory array, these cylindrical structures are deployed with TiO_2 as a memristor device. TiO_2 material is selected due to its high ionic conductivity, high schottky barrier height, multiple stable oxidation states and CMOS-compatibility [61]. The top electrode (*te*) and bottom electrode (*be*) are metallic platinum (Pt) or silver (Ag) electrodes which exhibit high oxidation resistance, high work-function and CMOS-compatibility. The physical parameters, variables, universal constants are defined in Table 2.3 and related equations are subsequently stated to illustrate the details of the model.

(a) Cylindrical front view. (b) Cross-sectional view.

Figure 2.7: Structural representation of TiO_2-based memristor described by Aachen model. This figure illustrates that TiO_{2-x} ions occupy $l_{act} \times r_{act}$ (green) volume given a full capacity of $l \times r_d$ (red) available to the memristor material.

The cylindrical memristor device has a total radius of r_d and total length of l_d which includes *te* and *be*. The maximum length of available for TiO_2 material (excluding *te*

Parameters	Description
te	Top electrode (positive terminal)
be	Bottom electrode (negative terminal)
V_{tb}	Voltage applied across the device
l_d	Total length of the device
l_{max}	Maximum length of TiO_2 material
l_{act}	Actual length of TiO_{2-x}
r_d	Total radius of the device
r_{act}	Actual radius of TiO_{2-x} material
N_{max}	Maximum concentration of O_{2-x} ions
N_{min}	Minimum concentration of O_{2-x} ions
N_{real}	Change in concentration of O_{2-x} ions
f	Frequency of operation
d_{Wa}	Activation energy required for $O \rightarrow O_{2-x}$ ions (1.3 eV)
R_{cont}	Resistance of metal electrode or contact (1500 Ω)
k_T	Conductivity of by-default channel (0.04 Scm^{-1})
A	Area of TiO_{2-x} disc
T_{real}	Operating temperature
K	Boltzman constant ($8.617e^{-5}eV$)
K_{Si}	Boltzman constant ($1.38e^{-23}Kgm^2s^{-2}K^{-1}$)
e	Electron charge ($1.6e^{-19}C$)
T_o	Ambient temperature ($293K$)
M_e	Mass of electron ($9.12e^{-31}Kg$)
H	Plank's constant ($6.626e^{-34}Kgm^2s^{-1}$))
h_d	Hopping distance ($0.4e^{-9}m$)
ϕ_{n0}	Work function default
ϕ	Work function
ϵ_s	Permittivity of O_{2-x}($1.51e^{-10}Fm^{-1}$)
μ_o	Mobility of electron
ϵ_{ϕ_b}	Permittivity of Schottky tunneling
n	Number of oxygen valence electrons (2)

Table 2.3: Parameters, variables and universal constants used for Aachen model.

and be) is given by l_{max}. The formation/recombination of conducting O_{2-x} ions and two e- during $TiO_2 \leftrightarrow TiO_{2-x}$ + 2e- changes the share of insulating and conducting region in a 3-D fashion *i.e.* both length and radius of the conducting TiO_{2-x} changes. These changes however differ according to the sign of voltage difference applied across te and be (V_{tb}). Region occupied by conducting TiO_{2-x} is given by

If $V_{tb} \geq 2e^{-5}V$, *i.e.* RESET operation with positive voltage difference,

$$r_{act} = r_{ini} + (r_d - r_{ini}) \times \frac{N_{ini} - N_{real}}{N_{ini} - N_{min}} \tag{2.3}$$

$$l_{act} = l_{ini} + (l_d - l_{ini}) \times \frac{N_{ini} - N_{real}}{N_{ini} - N_{min}} \tag{2.4}$$

Alternatively, if $V_{tb} \leq 2e^{-5V}$, *i.e.* SET operation with negative voltage difference,

$$r_{act} = r_{ini} + (r_d - r_{ini}) \times \frac{N_{real} - N_{ini}}{N_{max} - N_{ini}} \tag{2.5}$$

$$l_{act} = l_{ini} + (l_d - l_{ini}) \times \frac{N_{real} - N_{ini}}{N_{max} - N_{ini}} \tag{2.6}$$

where, initial (present) values of radius, length occupied and concentration of oxide ions are r_{ini} (r_{act}), l_{ini} (l_{act}) and N_{ini} (N_{real}), respectively. It is worth noting here that $Ron(Roff)$ is primarily depends on $N_{max}(N_{min})$.

A denotes the area of the conducting TiO_{2-x} region or volume occupied per unit length and R_{th} denotes the contact-to-contact ($te - be$) device resistance.

$$A = \pi \times \mathrm{r}_{act}{}^2 \tag{2.7}$$

$$R_{th} = \frac{1}{\mathrm{k}_T} \times \frac{l_d}{A} \tag{2.8}$$

Change in the concentration of TiO_{2-x} ions (ΔN) for any current I and length l_{act} is

$$\Delta N = -\int \frac{I dt}{n \times e \times A \times l_{act}} \tag{2.9}$$

$$N_{real} = N_{ini} + \Delta N \tag{2.10}$$

Total voltage across the memristor from te to be is broken down to smaller components denoted by V_{te-skt}, V_{skt-r0}, V_{r0-be1} and V_{be1-be}. These components are shown in Figure 2.8.

$$V_{te-be} = V_{tb} = V_{te-skt} + V_{skt-r0} + V_{r0-be1} + V_{be1-be} \tag{2.11}$$

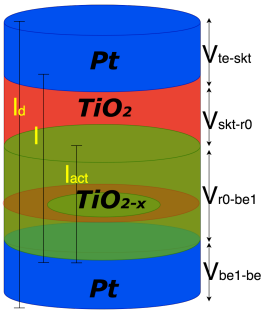


Figure 2.8: Voltage components across the memristor device.

Operating temperature is given by nominal room temperature plus change in temperature required to cause a given change in resistance (final - initial resistance value). Therefore, T_{real} is

$$T_{real} = T_o + \frac{\left(\frac{V_{te-be}}{I_{te-be}}\right) - R_{th}}{R_{th} \times \alpha} \tag{2.12}$$

where, α is the temperature coefficient of the memristor, $V_{te-be} = V_{tb}$ and $I_{te-be} = I_{tb}$ are total voltage across and current flowing through the memristor.

The working function of Si *i.e.* ϕ_{si} due to Schottky tunneling is related to the voltage V_{te-skt} described as follows.

If $V_{te-skt} < \phi_{n0} - \phi$,

$$\phi_{si} = \phi_{n0} - \phi - V_{te-skt} \tag{2.13}$$

$$\phi_b = \phi_{n0} - \left(\frac{e^3 \times n \times N_{real} \times \phi_{si}}{8 \times \pi^2 \times \epsilon^3}\right)^{\frac{1}{4}} \tag{2.14}$$

Else if $V_{te-skt} > \phi_{n0} - \phi$,

$$\phi_{si} = 0 \tag{2.15}$$

$$\phi_b = \phi_{n0} \tag{2.16}$$

The below equations provide intermediate values ($E0$, $E00$, $epsstrich$), which are further used to find current through the memristor *i.e.* I_{te-skt}.

$$E00 = e \times \frac{h}{2} \times \sqrt{\frac{n \times N_{real} \times 10^{26}}{M_e \times \epsilon_s}} \tag{2.17}$$

$$E0 = \frac{E00}{\tanh\left(\frac{E00}{K_{Si}\times 1000\times T_{real}}\right)} \tag{2.18}$$

$$epsstrich = \frac{E00}{\left(\frac{E00}{K_{Si}\times 1000\times T_{real}}\right) - \tanh\left(\frac{E00}{K_{Si}\times 1000\times T_{real}}\right)} \tag{2.19}$$

If $V_{te-skt} < 0$, *i.e.* for SET case, I_{te-skt} is

$$I_{te-skt} = -\frac{A \times Arich \times T_{real} \times 1000}{K_{Si}} \times \sqrt{\pi \times E00 \times e \times |V_{te-skt}|}$$

$$+\frac{\phi_b}{\cosh\left(\frac{E00}{K_{Si} \times 1000 \times T_{real}}\right)} \times exp\left(-e \times \frac{\phi_b}{E0}\right) \times \left[exp\left(e \times \frac{|V_{te-skt}|}{epsstrich}\right) - 1\right] \tag{2.20}$$

Else if $V_{te-skt} > 0$, *i.e.* for RESET case, I_{te-skt} is

$$I_{te-skt} = -A \times Arich \times (T_{real} \times 1000)^2 \times exp\left(-\frac{\phi_b}{K_b \times 1000 \times T_{real}}\right)$$

$$\times \left[exp\left(\frac{1}{K_b \times 1000 \times T_{real}} \times V_{te-skt}\right) - 1\right] \tag{2.21}$$

A memristor device experiences different electric field $\overrightarrow{E}$ across it for a small positive and negative voltage *i.e.* say $I(V_{tb} = 0.1V) \neq I(V_{tb} = -0.1V)$

$$\overrightarrow{E} = \frac{V_{te-be}}{L_d}, V_{tb} \cong 0.1V \tag{2.22}$$

$$\overrightarrow{E} = \frac{V_{skt-r0}}{L_{act}}, V_{tb} \cong -0.1V \tag{2.23}$$

The parameter γ below is represented by the ratio of electric force generated to cover one hopping distance by two e- and activation energy required by $TiO_2 \rightarrow TiO_{2-x}$.

$$\gamma = \frac{n \times e \times \overrightarrow{E}}{\pi \times d_{Wa}} \times h_d \tag{2.24}$$

The difference between the minimum and maximum energy band gap for TiO_2 is given by

$$d_{Wamin} = d_{Wa} \times e \times (\sqrt{1-\gamma^2} - \gamma \times \frac{\pi}{2} + \gamma \times \arcsin\gamma) \tag{2.25}$$

$$d_{Wamax} = d_{Wa} \times e \times (\sqrt{1-\gamma^2} + \gamma \times \frac{\pi}{2} + \gamma \times \arcsin\gamma) \tag{2.26}$$

Total current flown containing two valence e- per TiO_{2-x} ion through the memristor device is given by total charge (Q) x frequency of operation (f) over a volume (Area x hopping distance of two free e-) given by

$$I_{ions} = \left[(n \times e \times N_{real}) \times (\mathrm{A} \times h_d) \times \left\{\exp(\frac{-d_{Wamin}}{K_{Si} \times T_{real}}) - \exp(\frac{-d_{Wamax}}{K_{Si} \times T_{real}})\right\}\right] \times f \tag{2.27}$$

At any N_{real}, the individual sub-voltages across the memristor are expressed as

$$V_{skt-r0} = \frac{1}{n \times e \times N_{real} \times \epsilon_n} \times \frac{L_{act}}{\mathrm{A}} \times I_{skt-r0} \tag{2.28}$$

$$V_{r0-be} = \frac{1}{n \times e \times N_{real} \times \epsilon_n} \times \frac{(L_d - L_{act})}{\mathrm{A}} \times I_{r0-be} \tag{2.29}$$

$$V_{be1-be} = R_{cont} \times I_{be1-be} \tag{2.30}$$

Therefore, all these voltages along with V_{te-skt} (Equations 2.20, 2.21) adds up to the total voltage given by Equation 2.11.

The total resistance offered by the cylindrical TiO_2-based memristor disc is given by the ratio of total voltage V_{te-be} (Equation 2.11) seen across the memristor while total current of I_{ions} (Equation 2.27) is allowed through it.

2.2.2 Behavioral Models

Behavioral or mathematical or experimental models are compiled based on manufactured device characterization. However, these models deviates from analytically extracted models mainly due to the following reasons [74]. Firstly, memristor device technology is still in research phase *i.e.* under-developed and inaccurate device manufacturing units. Secondly, lack of in-depth understanding of the novel device, hence analytical models may neglect unknown (second or third order) factors affecting experimental results. Nevertheless, experimental models with acceptable accuracy are proposed for simulation purposes, contributing to an important aspect of research development.

Behavioral memristor models are proposed in research that can be utilized to simulate circuits for numerous set of applications. However, different applications require unique characteristics that only a single type of memristor can not provide. Digital applications such as ability to store data with distinguishable stable states, fast read/write operations and non-destructive read requires a highly non-linear (abrupt switching) memristive device. Whereas, analog circuits such as amplifiers, schmitt trigger and oscillators require linear device with deterministic integral current over time. Both current and voltage based models for linear, non-linear, schottky-tunnel non-linear devices, with special window functions, are some of the well-established variety of models proposed, each focusing

on a unique characteristic. However, due to a gamut of complex models, Shahar *et al.* proposed current(voltage) based threshold (voltage threshold) adaptive memristor model or TEAM(VTEAM) model [75]([76]) which combines these well established behavioral models into a single unified model. The paper claims that both TEAM and VTEAM models are simple, intuitive, closed form, computationally efficient and can be tuned to different types of memristors. TEAM and VTEAM model share the same premise (being current or voltage based is the only appreciable difference). Therefore, unified TEAM model is illustrated to explain this class of models, while integrating the main aspects of a behavioral memristor device model.

- **TEAM model**

TEAM model describes TiO_2-based memristor device, which is structurally represented as shown in Figure 2.9. Unlike analytical model, the device structure is two-dimensional (2-D) but with similar separable conducting, insulating and metallic contact (Pt) regions. The voltage (V_{tb}) is applied across contacts (top and bottom electrode) for read and write operations. As shown in the figure, each of the conducting (TiO_{2-x}) region (x) and insulating (TiO_2) region (D-x) have a 2-D range of [0,D], where x=0 and x=D represents $Roff$ and Ron, respectively. The illustration and supporting equations are described next.

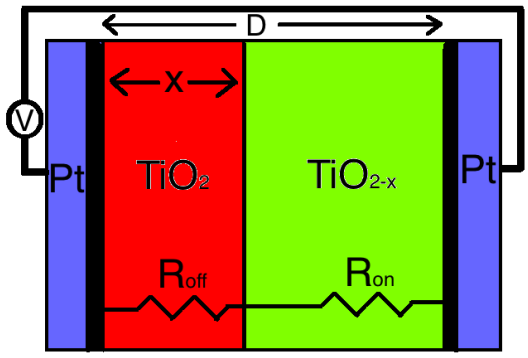


Figure 2.9: Behavioural (Mathematical) model.

The fully exhaustive expression presenting the main idea of the unified model is described by the following equation.

$$\frac{dx}{dt} = \begin{cases} K_{off} \times \left(\frac{i_t}{i_{off}} - 1\right)^{\alpha_{off}} \times f_{off}, & 0 \leq i_{off} \leq i_t \\ 0, & i_{on} \leq i_t \leq i_{off} \\ K_{off} \times \left(\frac{i_t}{i_{on}} - 1\right)^{\alpha_{on}} \times f_{off}, & i \leq i_{on} \leq 0 \end{cases} \tag{2.31}$$

where x is the state variable with range [0,D]; K_{off} and K_{on} are positive and negative constants; α_{off} and α_{on} are parametric constants; i_{off} and i_{on} are current thresholds;

f_{off} and f_{on} are window functions (used to restrict the state variable within its allowed range) associated with OFF and ON states, respectively. These window functions are equal(unequal) for symmetrical(asymmetrical) memristive devices around i_t=0 and are explained later.

The final state of x is the step-by-step summation of the above expression integrated over time and the present state x_{last}, which is given by

$$x_{new} = x_{last} + \left(\frac{dx}{dt}dt\right) f(x) + S \times N_p \tag{2.32}$$

where final state x value is given by value of x_{new} at the end of the operation, $f(x)$ is either f_{off} or f_{on} associated with OFF or ON operation, respectively, N_p is a constant representing noise window and S is the associated sign multiplier.

The expression governing different sets of models are described next. **Linear ion drift model** [2] represents $v(t)$ by the following expression.

$$v(t) = \underbrace{\left[R_{on} + \frac{R_{off} - R_{on}}{x_{off} - x_{on}}\right](x - x_{on})}_{R_{mem}} \times i(t) \tag{2.33}$$

Here, R_{mem} linearly increases with R_{on} and R_{off} as minimum and maximum values of resistance at boundary conditions x as x_{on} and x_{off}, respectively.

Non-linear ion drift model [68] represents $v(t)$ by the following expression.

$$v(t) = \underbrace{R_{on} \times exp\left(\frac{\lambda}{x_{off} - x_{on}}(x - x_{on})\right)}_{R_{mem}} \times i(t) \tag{2.34}$$

where λ is expressed as

$$\lambda = ln\left(\frac{R_{off}}{R_{on}}\right) \tag{2.35}$$

Simmons tunnel barrier model [70] is a non-linear and asymmetric behavioral model with movement of ionized dopants (switching of state variable x) following an exponential function. The exponential behaviour is due to electron tunnelling taken into account. This model represents $x(t)$ as the oxygen vacancy drift (here, $\frac{dx(t)}{dt}$ is the drift velocity) by the following expression.

$$\frac{dx(t)}{dt} = \begin{cases} c_{off} \sinh\left(\frac{i_t}{i_{off}}\right) \exp\left[-\exp\left(\frac{x - a_{off}}{w_c} - \frac{|i|}{b}\right) - \frac{x}{w_c}\right], & \text{i}\geq 0 \\ c_{on} \sinh\left(\frac{i_t}{i_{on}}\right) \exp\left[-\exp\left(\frac{x - a_{on}}{w_c} - \frac{|i|}{b}\right) - \frac{x}{w_c}\right], & \text{i}\leq 0 \end{cases} \tag{2.36}$$

where c_{off}, c_{on}, a_{off}, a_{on}, w_c and b are fitting parameters and $i_{off}(i_{on})$ are current thresholds for OFF(ON) state.

Previously mentioned **window functions** are functions that restrict the state variable x to be within the stipulated range (here, say [0,D]) and account for asymmetrical behaviour of a memristor device. Following unifies well known Jogelkar [77], Biolek [78] and Prodromakis [79] window functions.

$$f_{off}(x) = \exp\left[-\exp\left(\frac{x - a_{off}}{w_c}\right)\right],$$

$$f_{on}(x) = \exp\left[-\exp\left(\frac{x - a_{on}}{w_c}\right)\right] \tag{2.37}$$

These windows are incorporated using the Equation 2.31. Below is a self-explanatory example that describes fitting parameters chosen for a linear memristor device.

$$k_{off} = k_{on} = \mu_v \left(\frac{R_{ON}}{\mathrm{D}}\right) i_{on},$$

$$\alpha_{off} = \alpha_{on} = 1,$$

$$i_{off} = i_{on} = 0, \tag{2.38}$$

$$x_{on} = \mathrm{D},$$

$$x_{off} = 0,$$

$$x = \mathrm{D} - \mathrm{w}$$

It is worth mentioning here that using current-based TEAM model as voltage-based VTEAM model require some minor changes. The same Equations 2.33 and 2.4 can be defined for $i(t)$ instead of $v(t)$, replacing (in Equation 2.36) i_{off} and i_{on} with v_{off} and v_{on}, respectively, while keeping the fitting parameters unchanged.

2.2.3 Model Selection

1. **Supporting device variations** - Aachen model supports variation analyses for memristive devices. Physical device parameters namely N_{min}, N_{max}, r_d and l can be varied intuitively, thereby taking device variations into account to simulate circuit designs for functionality and reliability. Although, in the behavioral TEAM or VTEAM model, fitting parameters and window functions can be varied accordingly as well, but it is less-intuitive to quantify the extent of variation with respect to actual device dimensions.

2. **Ease of usability** - Unified TEAM model has an appreciable feature of simplicity and generality for various set of memristor devices. However, since this work only focuses on digital circuits (non-linear ion drift based device). Therefore, working with model supporting additional device types with their corresponding (complex) window functions may be a bit tedious and unnecessary.

3. **Collaboration** - This thesis work is performed as part of the project "computation-in-memory" (CiM) and RWTH Aachen university are one of the major collaborators with Delft university of technology. In order to have a unified approach and methodology to perform cohesive research, Aachen model is considered to be a favourable choice.

2.3 Device Simulation using Verilog-A Models

This section covers the analyses of a memristor device using Aachen Verilog-A model. Since control circuitry is still based on CMOS device technology, 90nm-TSMC CMOS device models are utilized to initialize the memristor and regulate voltage of operation. Variation analyses with emphasis on the device parameters causing the variations and their corresponding (extent of) change in some of the key device metrics is presented. Key learning and pointers are subsequently discussed. Critical device metrics extracted for this analyses are as follows.

- $Roff/Ron$ - Ratio of OFF and ON stable resistance states.
- $PVth(NVth)$ - Positive(negative) voltage thresholds for OFF(ON) state.
- $E_r(E_w)$ - Energy consumption during a read(write) operation.
- $T_{on}(T_{off})$ - Write time required to switch OFF(ON) $\rightarrow$ ON(OFF) state.

2.3.1 Read and Write Operations

The fundamental read and write operations are performed to validate the model, provide hands-on experience in using these models, offer better understanding of the memristor device behaviour and establish setup and measuring methodology that can be utilized later for evaluating logic and circuit designs. Setup and measuring methodology is provided below, with supporting figures and tables to present the results and trends followed by key device metrics.

- Setup and measurement methodology are enumerated as follows.

    1. Alternate write followed by read operation is performed with relaxed cycle period of 1μs each. Piece-wise-linear (PWL) description of voltage application (V_{tb}) is shown in Figure 2.10a.
    2. Read operation is performed by providing $V_{tb} = V_{read}$ = -0.1 V and mentoring the current through the memristor. An ON state has a current of nearly $\frac{Roff}{Ron}$ times OFF state current.

3. $Roff$ is normalized with respect to Ron. Four nominal resistance ratios of 10, 100, 1000, 1k are explored for this analyses.
4. Write operation is performed by providing V_{tb} with incremental(decremental) step of 0.1V(-0.1V) to investigate $PVth$ and $NVth$ *i.e.* switching voltage for ON(OFF) $\rightarrow$ OFF(ON) state.
5. Write operation is considered to be complete if the memristor reaches within 2-3% of its destined state and thereafter write time (T_{off} or T_{on}) is calculated.
6. E_r, E_w is calculated as the product of voltage applied (V_{tb}) and time-integral of the current (I_t) flowing through the device *i.e.* $V_{tb} \times \int_0^T I_t dt$ during the time-constrained read and write operation.

- Discussion regarding waveforms to illustrate read and write operation :-

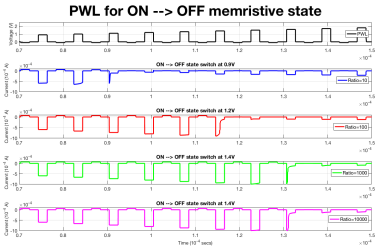
1. Figure 2.10a shows PWL of applied V_{tb} with alternative write and read cycles to monitor the state of the memristor device for the four $Roff/Ron$ configurations.
2. Figure 2.10b displays the hysteresis curves for the four configurations. Plot beyond positive(negative) threshold voltage remains in OFF(ON) state. It is inferred from the graph that for larger $Roff/Ron$, $NVth$ has a well defined as compared to smaller $Roff/Ron$.
3. Figure 2.10c provides an indication of the change in write time (here, shown T_{off}) with change in the write voltage (V_{tb}). As expected, T_{off} decreases with increase in V_{tb} applied for writing. Here, voltage is swept from 1V to 1.8V with step size of 0.1V to show the above trend.
4. Figure 2.10d illustrates the manner in which T_{off} is calculated. Here, the memristor switches from ON $\rightarrow$ OFF state at $V_{tb} = PVth = 1.1$V with $Roff/Ron$=10. Another point worth mentioning here is that at $V_{tb}(< PVth)$, it starts to switch but 1μs write time is not enough.

2.3.2 Effects of Variations

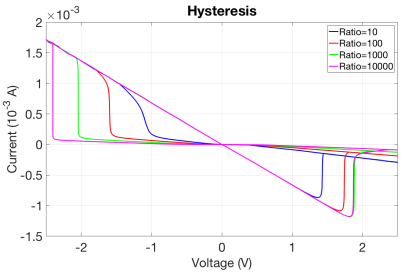
This subsection provides an overview of the causes and effects of memristor device variations on key device metrics using Verilog-A Aachen model.

- Parameters responsible for memristor device variations, in order of their impact on device metrics, are enumerated as follows.

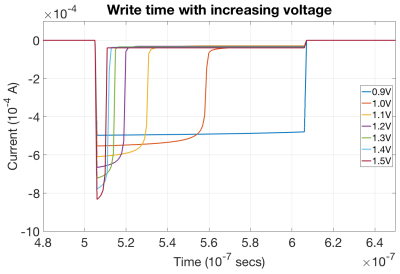
1. N_{min} - N_{min} is the minimum possible concentration of conducting (TiO_{2-x}) ions *i.e.* concentration at OFF state ($Roff$). Understandably, variations in this parameter has maximal impact on $Roff/Ron$ ratio, which in turn impacts $PVth$, E_r (only for the OFF state), E_w, both T_{off} and T_{on}. Figure 2.11 showcases the variation effect with 30% variation in N_{min} (nominal N_{min}=0.0267$\times 10^{26}$ per cm^3 with $Roff/Ron$=100).



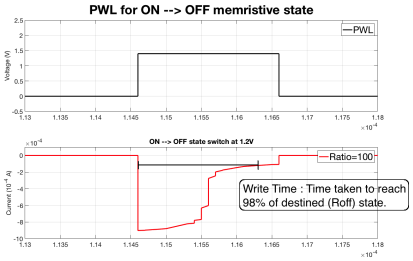
(a) Plot to illustrate PWL description of V_{tb} of the four configurations. Here, switching voltage $\leq PVth$.



(b) Hysteresis plot using relaxed sinusoidal V_{tb} supply using 10, 100, 1000, 1k resistance ratio.



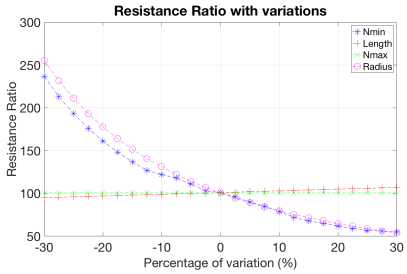
(c) Different write times (T_{off}) with increasing write voltage (V_{tb}).



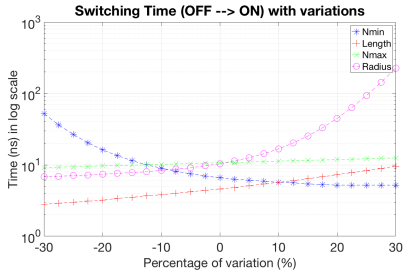
(d) Write time (T_{off}) illustrated as time taken to reach 98% $Roff$.

Figure 2.10: Read and write operation using Verilog-A Aachen model.

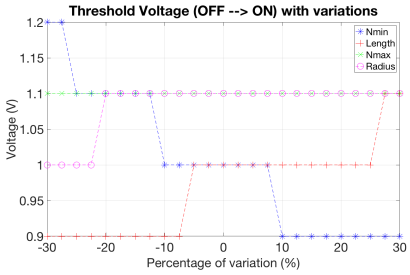
2. r_d - r_d is the radius of the cylindrical (TiO_2)-based memristor device with maximum radius occupied by (TiO_2) or (TiO_{2-x}) ions. Understandably, variations in this parameter also has a significant impact on $Roff/Ron$ ratio, which in turn impacts $PVth$, E_r (only for the OFF state), E_w, both T_{off} and T_{on}. Figure 2.11 showcases the variation effect with 30% variation in r_d (nominal r_d=32nm with $Roff/Ron$=100).
3. l - l_d is the total length of the cylindrical memristor device with maximum length of l available occupied by (TiO_2) or (TiO_{2-x}) ions, while the rest ($l_d - l$) is occupied by metallic Pt contacts. For a 30% variation in l (nominal l=1nm with $Roff/Ron$=100, Figure 2.11 supports that there is a very small impact on $Roff/Ron$ ratio and in turn on the other device metrics.
4. N_{max} - N_{max} is the maximum possible concentration of conducting (TiO_{2-x}) ions *i.e.* concentration at ON state (Ron). Intuitively, since its nominal value is considered to be a comparatively larger value than N_{min} ($>> 100 \times N_{min}$), variations in this parameter has negligible impact on $Roff/Ron$ ratio and all the other device metrics. Figure 2.11 showcases the variation effect with 30% variation in N_{max} (nominal $N_{max} = 10 \times 10.0^{26}$ per cm^3 with $Roff/Ron$=100 and Ron=2 $K\Omega$).



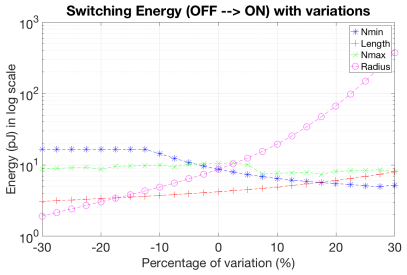
(a) $Roff/Ron$ (nominally 100) variation impact due to 30% parametric variations.



(b) T_{off} $(ON \rightarrow OFFstate)$ variation impact due to 30% parametric variations.



(c) $NVth$ $(ON \rightarrow OFFstate)$ variation impact due to 30% parametric variations.



(d) E_w $(ON \rightarrow OFFstate)$ variation impact due to 30% parametric variations.

Figure 2.11: Device metrics variation with 30% parametric (physical dimensions) variations.

- Key take-aways and pointers regarding memristor device behavioral analyses :-

  1. Sinusoidal PWL voltage (V_{tb}) is only used for hysteresis plotting only. The plot predicts slightly less V_{th} voltage as generally required to switch between the states. This is due to the fact that for any switching voltage, preceding applied voltages corresponding to the sinusoidal curve has somewhat already pushed the memristor towards the destined state. Hence, the transient analyses using sinusoidal PWL does not show exact $PVth$, $NVth$ and understandably, accurate write time.
  2. Same can be stated to explain that step incremental increase of V_{tb} (during the write cycle) as PWL (with alternate write and read cycles) is also not the correct method to find V_{th}. Therefore, for accurate measurement of $PVth(NVth)$, the memristor is SET(RESET) after every attempted write cycle aimed to switch ON(OFF) $\rightarrow$ OFF(ON) state.
  3. For read operation, V_{tb} as V_r = -0.1V (or a small negative voltage) is suggested mainly for two following reasons. Firstly, a negative voltage essentially pushes (negligibly though) the OFF state towards ON side which allows a flow of more stable and deterministic OFF current. Secondly, a small voltage because it inflicts least read disturbance and consumes least read power. Although, a sensitive sense amplifier is required to be able to differentiate between such low

ON and OFF currents. *(The proposed arithmetic circuit design in Chapter 5 has an additional feature to ensure a low (access) voltage of operation).*

4. There is an obvious trade-off between power consumed and write time. Based on the application, a sound choice can be made.
5. N_{min} and r_d are main source of variations in a TiO_2-based memristor, with 30% variation in each of these parameters cause more than 2.5X, 100X, > 100, 2-3X variation in $Roff/Ron$, T_w, E_w and E_r, respectively.

Overview and Benchmarking of Memristor-based Primitive Logic Designs

3

Primitive logic designs or primitive gates are the building blocks of all arithmetic and functional circuit designs utilized to perform computing for countless digital applications. Correct functionality of a digital circuit design using primitive logic designs is of prime importance but is not the only factor for choosing a certain logic design type. Efficacy related to speed, peak power, overall energy consumption, area utilization and reliability against device variations for better yield of primitive logic design building them are extremely critical as well. In this regard, due to lack of such analyses at the gate level, this chapter provides a comprehensive quantitative benchmarking of existing memristor-based primitive logic designs.

The organization of this chapter is as follows. Section 3.1 classifies different memristor-based primitive logic designs available in research. Section 3.2 briefly illustrates the structure and working principle for each of these primitive logic designs. Section 3.3 begins by describing benchmarking methodology and defining key efficiency metrics essential for comparison. The approach used to account for device and voltage variations is also illustrated. Thereafter, extensive quantitative benchmarking is presented which substantiates that scouting logic is the best design to build arithmetic and functional circuits.

3.1 Classification

A multitude of memristor-based primitive logic designs have been proposed in literature with or without the involvement of CMOS device(s) to ensure the functionality of the design [80]. But the designs involving CMOS device(s) can not support highly compact crossbar structure and hence are not included for this analyses. Although CMOS device (an NMOS) is still used as a pass gate in a 1T-1R (1 transistor + 1 memristor bit-cell) configuration, but its involvement is only limited to selecting a bit-cell for read and write operations. (Not to mention, control circuitry, drivers and peripheral circuits are CMOS-based but again, not involved in functionality of primitive logic designs.) Classification of existing primitive logic designs (supporting crossbar structure) is presented in Figure 3.1. The basis of classification is described next.

Computation-in-Memory (CIM)			
CIM - A		CIM - P	
Output in ARRAY		Output in PERIPHERY	
Output representation - RESISTIVE		Output representation - VOLTAGE	
CIM - Ar	CIM - Ah	CIM - Pr	CIM - Ph
Input - RESISTIVE	Input - HYBRID	Input - RESISTIVE	Input - HYBRID
• Snider - NOT, COPY, NAND • IMPLY - IMP, NAND • MAGIC - NOT, NOR • FBL - AND, S/MFO COPY, S/MFO NAND	PLiM - MAJ	Scouting - OR, AND, XOR	

Figure 3.1: Primitive logic designs classification.

- **Output location** - Output of a logic design is present either in the periphery or within the memristor crossbar array itself. The associated design types are respectively classified as computation-in-memory-periphery (CiM-P) and CiM-A.
- **Output state representation** - Output state is represented either as resistance or voltage value. Logic designs with their output obtained in(within) the periphery(crossbar array) can only be represented as voltage(resistance) value *i.e.* CiM-P(CiM-A).
- **Input state representation** - Input state(s) is represented as resistance and/or voltage value. However, one of the input is bound to have resistance representation *i.e.* atleast one input stored within the crossbar array, to ensure in-memory computing. Logic designs with atleast one voltage input are classified as hybrid (denoted by suffix h) while logic designs with all mem-resistive inputs are classified as resistive (denoted by suffix r).

Advantages and disadvantages of having a certain input/output representation and location of the output are accessed along with description of all logic designs listed in Figure 3.1.

3.2 Existing Primitive Logic Designs

This section begins with the demonstration of write and read operations performed in a memristor based crossbar array. This is followed by detailed illustration of all the primitive logic designs, while following the order in which they are classified in the previous section. First, CiM-Ar primitive logic designs are covered citing a common issue inherent to this class, especially concerning their reliability against device and voltage variations. CiM-Ah based majority logic design is discussed next, while drawing attention to reduced device endurance as a major disadvantage. Finally, CiM-P based scouting logic design is described, discussing its key advantages over others.

- **Write and read operation in memristor-based crossbar array**

While working with memristor-based primitive logic design, write followed by read operation ensures correct initialization (*i.e.* input mem-resistance values) of the memristors. The approach followed to write a memristor, say $R_{2,2}$, in a crossbar array is shown in Figure 3.2 while voltages applied to all rows and columns are enlisted in Table 3.1. The choice of voltages applied ensures that voltage across the target memristor ($R_{2,2}$) has a value V_w, such that $V_w > |NVth|(PVth)$ to switch from $Roff(Ron) \rightarrow Ron(Roff)$. To ensure unwanted switching of half-select memristor bit-cells, voltages across them are either V_{wh} or $(V_w - V_{wh})$, such that both V_{wh} and $(V_w - V_{wh}) < min(|NVth|, PVth)$.

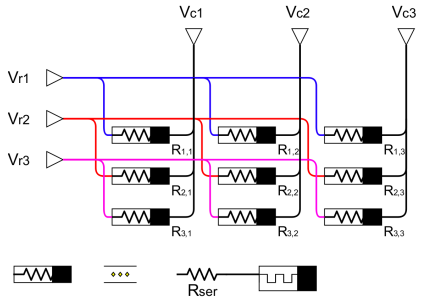


Figure 3.2: Write operation.

Vr/Vc	Vc1	Vc2	Vc3
Vr1	Vwh, Vwh	Vwh, 0	Vwh, Vwh
Vr2	Vw, Vwh	Vw, 0	Vw, Vwh
Vr3	Vwh, Vwh	Vwh, 0	Vwh, Vwh

Table 3.1: Voltage values to write $R_{2,2}$.

Read operation is performed by applying a row (read) voltage $V_r << V_w$ at one terminal (preferably bottom terminal *be*, as explained in the Section 2.3) of the target memristor, say $R_{2,2}$, and a sensing circuit connected to the other (floating terminal), as shown in Figure 3.3. The sensing circuit is shared by an entire column and hence half-select cells (all cells other than target cell in the column) are grounded by their respective row drivers. This enables the sensing circuit to evaluate(or quantify) the current exclusively flowing through the target memristor. As mentioned in the Section 2.3.1, sensing circuit should be able to differentiate ON and OFF currents, where ON state current is $\sim \frac{Roff}{Ron} \times$ OFF state current

Subsequently, primitive logic operation is performed in the third cycle (first two cycles involve initialization and ensuring its correctness) that may take one or more cycles depending on the design's functionality. On completion of this evaluation phase, target memristor (storing the output resistance) is read out to quantify the final outcome.

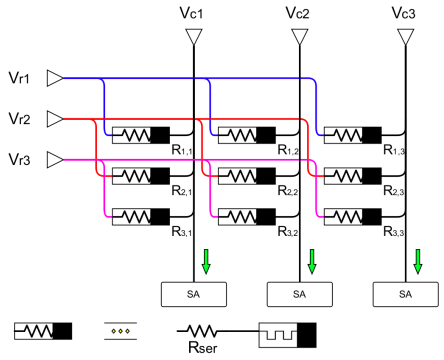


Figure 3.3: Read operation.

Vr/Vc	Vc1	Vc2	Vc3
Vr1	0, 0	0, Z	0, 0
Vr2	Vr, 0	Vr, Z	Vr, 0
Vr3	0, 0	0, Z	0, 0

Table 3.2: Read Voltage V_{ri} applied.

The remainder of this section illustrates only the third cycle *i.e.* the evaluation phase, while considering the execution of pre-write (initialization), pre-read (initialization correctness) and post-read (output read-out) to be implicit.

CiM-Ar

CiM-Ar primitive logic designs, also known as memristive stateful logic gates, have all input(s) and output(s) within the crossbar array and are represented as resistance values. Primitive logic designs discussed under this category are material implication logic (IMPLY), Snider, fast boolean logic (FBL) and memristive aided logic (MAGIC). Advantages common to this category of logic designs are -

- Intermediate output in a cascaded set of operations does not require additional read-out and write operation.
- Sensing circuitry is required only once *i.e.* for the final output after the last evaluation.

However, these designs have major disadvantages, namely -

- Constant writing during evaluation phase reduces device endurance.
- State machines to regulate control voltages utilize large area and consume high power.

3.2.1 Material Implication Logic (IMPLY)

- **IMPLY**

Borghetti *et al.* proposed material implication logic (IMPLY) [49], which is denoted by p IMPLIES q or $p \rightarrow q$ and is defined as "if not p then q". The structure as shown in Figure 3.4, consists of two memristors, namely P and Q, with applied supply voltages(stored resistance states) as $V_p(p)$ and $V_q(q)$, respectively. A resistor R_g connects the common node (joining their top electrodes) of the two memristors to ground. The truth table of IMPLY is shown in Table 3.3, which describes the logic function *(NOT p) OR q*.

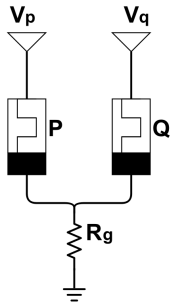


Figure 3.4: IMPLY.

p	q	q_{final} $(p \rightarrow q)$
0	0	1
0	1	1
1	0	0
1	1	1

Table 3.3: Truth table for IMPLY.

Gate Design Specification - IMPLY is evaluated by applying suitable voltages V_p=V_{cond} and V_q=V_{set} while connecting floating node x through a resistor R_g ($Ron << R_g << Roff$) to ground. Correct functionality requires $V_{cond} < |NVth|$ and $V_{set} = 2 \times V_{cond} > |NVth|$. The evaluated result is stored in Q and therefore this gate configuration is destructive.

Working Principle - The four possible cases are discussed as follows. Case I *i.e.* pq=00 implies that initial voltage across Q (V_{qx}) is $\sim V_{set}$, enough for Q to switch from state 0 $\rightarrow$ 1. Case II *i.e.* pq=01 implies that initial $V_{qx} \sim 0$, therefore Q remains at state 1. Case III *i.e.*, pq=10, $V_{qx} \sim V_{set}$ - V_{cond}, which is not enough for Q to switch implying Q remains at state 0. Case IV *i.e.*, pq=11 implies initial $V_{qx} \sim 0$, therefore Q remains at state 1. Case I is known as "write case" because it is only case with actual switching and hence determines the minimum period of operation. Case III is known as "endurance case" because output memristor Q needs to endure $V_{qx} = V_{set}$ - V_{cond} without being disturbed (or written) during the operating cycle and hence determines maximum period of operation.

- **IMPLY NAND/NOR**

IMPLY primitive gate can be cascaded sequentially to perform traditional logic functions such as NAND/NOR. The structure as shown in Figure 3.5, consists of three memristors, namely P, Q and S, with applied supply voltages(stored resistance states) as $V_p(p)$, $V_q(q)$ and $V_s(s)$, respectively. Similar to IMPLY, resistor R_g connects the common node (joining their top electrodes) of the three memristors to ground. The truth table of IMPLY NAND is shown in Table 3.5, which describes the logic function *NOT (p AND q)*.

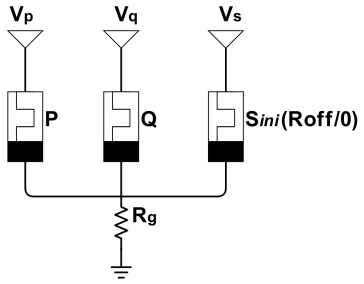


Figure 3.5: IMPLY NAND/NOR.

Cycles	Vp	Vq	Vs
Ini			-Vreset
Eva1		Vcond	Vset
Eva2	Vcond		Vset

Table 3.4: Evaluation steps.

p	q	$s = \bar{q}$ $(q \rightarrow 0)$	$s_{final} = \overline{pq}$ $(p \rightarrow \bar{q})$
0	0	1	1
0	1	0	1
1	0	1	1
1	1	0	0

Table 3.5: Truth table for IMPLY NAND.

Gate design specification - NAND function requires two evaluation cycles while performing two consecutive IMPLY functions. Table 3.4 shows the cycle-wise application of V_p, V_q and V_s supply voltages while connecting the floating node x through a resistor R_g ($Ron << R_g << Roff$) to ground for correct functionality. The required voltages are similar to IMPLY *i.e.* $V_{cond} < |NVth|$ and $V_{set} = 2 \times V_{cond} > |NVth|$ while keeping $V_{reset} > PVth$. The final result is stored in S (not an input) and therefore this gate configuration is not destructive. Evaluation phase uses two cycles with its principle described next.

Working principle - Evaluation steps are enlisted in Table 3.4 and truth table (including intermediate and final outputs) is shown in Table 3.5. Output memristor S is initialized to $Roff(0)$ regardless of the inputs. Case I, II, III *i.e.* pq=00, 01, 10 involves switching of S from $0 \rightarrow 1$ atleast once during the two evaluation cycles (write cases) and hence determine the minimum period. Case IV *i.e.* pq=11 implies S has to endure two cycles without switching (endurance case) and hence determines the maximum period.

In the first evaluation step (Eva1), input configuration of $qs = q0$ ($q \rightarrow 0$) implying $s = \bar{q}$. In the second evaluation step (Eva2), input configuration is $ps = p\bar{q}$ ($p \rightarrow \bar{q}$) implying $s_{final} = \overline{\bar{p} + \bar{q}} = \overline{pq}$. Therefore with these two steps, NAND functionality is derived. It is worth noting here that IMPLY NAND gate can be interchangeably used as IMPLY NOR gate if reverse input configuration is assumed *i.e.* Ron and $Roff$ as 0 and 1 state, respectively.

3.2.2 Snider Logic

- **Snider NOT**

Snider *et al.* proposed snider logic designs [81] which can perform NOT, NAND and COPY (buffer) primitive functions. The structure of snider NOT gate is similar to IMPLY gate consisting two memristors, namely P and Q, with applied supply voltages(stored resistance states) as $V_p(p)$ and $V_q(q)$, respectively. Similar to IMPLY, resistor R_g connects the common node (joining their top electrodes) of the two memristors to ground. The truth table of Snider NOT is intuitive ($q = NOT\ p$) and hence not shown.

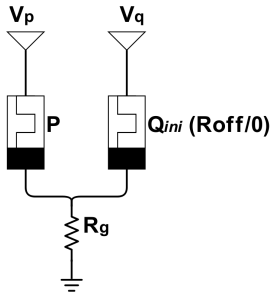


Figure 3.6: Snider NOT.

Cycles	Vp	Vq
Ini		-Vreset
Eva	Vcond	Vset

Table 3.6: Evaluation steps for Snider NOT.

Gate design specification - Evaluation is performed by applying suitable voltages $V_p = V_{cond}$ and $V_q = V_{set}$ connected through a resistor R_g ($Ron << R_g << Roff$) to ground. The appropriate voltages for correct functionality require $V_{cond} < |NVth|$ and $V_{set} = 2 \times V_{cond} > |NVth|$ while keeping $V_{reset} > PVth$. The evaluated result is stored in Q (not an input) and therefore this gate configuration is not destructive.

Working principle - Snider NOT can be considered as a special case of IMPLY with q always equal to 0 in $p \rightarrow q$ *i.e.* $p \rightarrow 0$ implies $q_{new} = \bar{p}$. Evaluation steps are shown in Table 3.6. Memristor Q is initialized to $Roff(0)$ regardless of the input. In the evaluation cycle, case I *i.e.* p=0 implies $V_{qx} \sim V_{set}$, which is enough to switch Q from 0 $\rightarrow$ 1. Case II *i.e.* p=1 implies $V_{qx} \sim V_{set} - V_{cond}$, which is not enough to switch Q and hence Q remains 0. Therefore, NOT gate functionality is achieved.

- **Snider Copy**

Snider COPY primitive gate acts as a buffer *i.e.* it copies the input to the output. The structure as shown in Figure 3.7, consists of two memristors, namely P and Q, with applied supply voltages(stored resistance states) as $V_p(p)$ and $V_q(q)$, respectively. Unlike IMPLY, resistor is not required. The truth table of snider COPY is intuitive ($q = p$) and hence not shown.

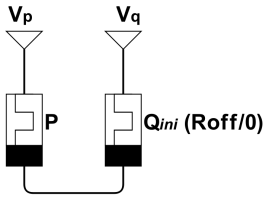


Figure 3.7: Snider COPY.

Cycles	Vp	Vq
Ini		-Vreset
Eva	GND	Vset

Table 3.7: Evaluation steps for Snider COPY.

Gate design specification - Evaluation is performed by applying suitable voltages $V_p = GND$ and $V_q = V_{set}$. The appropriate voltages for correct functionality require $V_{set} > |NVth|$ while $V_{set}/2 < |NVth|$ during the evaluation phase. The evaluated result is stored in Q (not an input) and therefore this gate configuration is not destructive.

Working principle - Evaluation of snider copy can be performed while considering the two memristors P and Q as voltage dividers (in series) with V_{set} across them. Evaluation steps are shown in Table 3.6. Memristor Q is initialized to $Roff(0)$ regardless of the input. In the evaluation cycle, case I *i.e.* p=0 implies $V_{qx} \sim V_{set}/2$, which is not enough to switch Q and hence Q remains 0. Case II *i.e.* p=1 implies $V_{qx} \sim V_{set}$, which is enough to switch Q from $0 \rightarrow 1$. Therefore, copy or buffer functionality is achieved.

- **Snider NAND/NOR**

Snider NOT primitive gate can be cascaded to perform traditional logic functions such as NAND/NOR. The structure as shown in Figure 3.8, which consists of three memristors, namely P, Q and S, with applied supply voltages(stored resistance states) as $V_p(p)$, $V_q(q)$ and $V_s(s)$, respectively. Similar to IMPLY-NAND, resistor R_g connects the common node (joining their top electrodes) of the three memristors to ground. The truth table of Snider NAND is intuitive ($s = NOT\ (p\ AND\ q)$) and hence not shown.

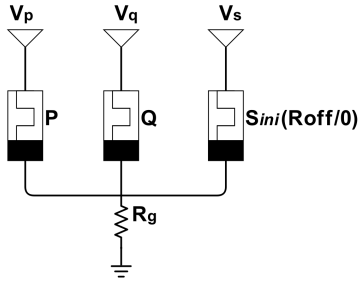


Figure 3.8: Snider NAND/NOR.

Cycles	Vp	Vq	Vs
Ini			-Vreset
Eva	Vcond	Vcond	Vset

Table 3.8: Evaluation steps for Snider NAND.

Gate design specification - Unlike IMPLY-based NAND gate, snider NAND gate is performed in a single evaluation cycle. Table 3.8 shows the application of V_p, V_q and V_s supply voltages during appropriate cycles while connecting the floating node x through a resistor R_g ($Ron << R_g << Roff$) to ground for correct functionality. The required voltages are similar to IMPLY NAND *i.e.* $V_{cond} < |NVth|$ and $V_{set} = 2 \times V_{cond} > |NVth|$ while keeping $V_{reset} > PVth$. The evaluated result is stored in S (not an input) and therefore this gate configuration is not destructive.

Working principle - Evaluation steps are shown in Table 3.8. Memristor S is initialized to $Roff$ regardless of the inputs. Case I, II, III *i.e.* pq=00, 01, 10 require writing with S switching $0 \rightarrow 1$ (write cases) and hence determine the minimum period. Case IV *i.e.*, pq=11 implies S has to endure a cycle without switching (endurance case) and hence determines the maximum period. The evaluation can be simplified as *(p AND q)* $\rightarrow 0$ implying $s = \overline{pq} + 0 = \overline{pq}$ and hence NAND functionality is achieved. It is worth noting here that like IMPLY NAND, Snider NAND gate can be interchangeably used as Snider NOR gate if reverse input configuration is assumed *i.e.* Ron and $Roff$ as 0 and 1 state, respectively.

- **Fast Boolean Logic (FBL)**

Xie *et al.* proposed family of FBL-based logic designs [7] that can perform AND logic and multiple fan-outs for COPY and NAND logics. Single fan-out snider COPY and NAND logic designs are extended to form double fan-out (DFO) FBL-based COPY and NAND gates. The structures of these logic designs are shown in Figures 3.9, 3.10 and 3.11, with COPY and NAND DFO gates consisting of one additional memristor for providing identical second output. Applied supply voltages (stored resistance states) for P, Q, $Q0$, $Q1$, S, $S0$ and $S1$ are $V_p(p)$, $V_q(q)$, $V_{q0}(q0)$, $V_{q1}(q1)$, $V_s(s)$, $V_{s0}(s0)$ and $V_{s1}(s1)$, respectively. Similar to snider-NAND logic design, FBL-NAND requires a resistor R_g, which connects the common node (joining their top electrodes) of the memristors to ground. The truth tables of DFO COPY, AND and DFO NAND are intuitive *i.e.* $q0 = q1 = p$, $s = (p\ AND\ q)$ and $s0 = s1 = NOT\ (p\ AND\ q)$, respectively, hence not shown.

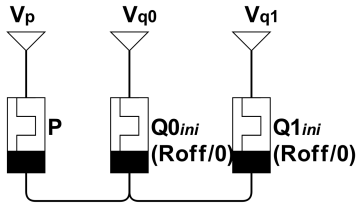


Figure 3.9: FBL-based DFO COPY.

Cycles	Vp	Vq0	Vq1
Ini		-Vreset	-Vreset
Eva	Vw	GND	GND

Table 3.9: Evaluation steps for (DFO) FBL-based COPY.

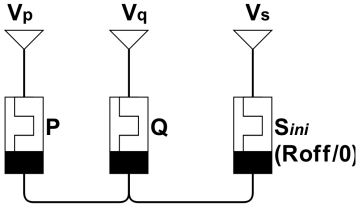


Figure 3.10: FBL-based AND.

Cycles	Vp	Vq	Vs
Ini			-Vreset
Eva	Vw	Vw	GND

Table 3.10: Evaluation steps for FBL-based AND.

Gate design specification - The output memristor(s) is(are) initialized to $Roff$ regardless of the inputs. Table 3.9, 3.10 and 3.11 show the application of supply voltages during appropriate cycles. For FBL COPY and AND designs, node x is kept floating while for FBL NAND design, floating node x is connected via resistor R_g ($Ron << R_g << Roff$) to ground for correct functionality. $V_{reset} > PVth$ for initializing the output memristors to $Roff(0)$ while required voltages V_w and V_{wh} for all these gates are described below. The evaluated result is stored in $Q*(COPY)$, $S*$ (not inputs) and therefore these gate configuration are not destructive.

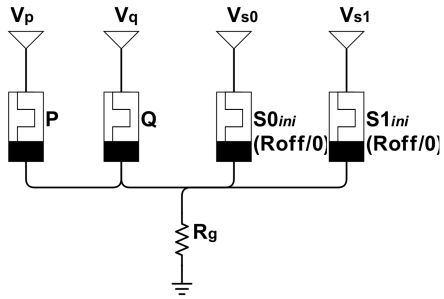


Figure 3.11: FBL-based NAND (DFO).

Cycles	Vp	Vq	Vs0	Vs1
Ini			Vreset	Vreset
Eva	Vwh	Vwh	Vw	Vw

Table 3.11: Evaluation steps for (DFO) FBL-based NAND.

$$\max\left(|V_{reset}|, |V_{set}|\ \right) < V_w < V_{reset}\frac{R_{on}R_g + R_{on}\left(R_{on}+R_g\right)}{R_{on}\left(R_{on}+R_g\right)}$$

$$V_w - V_{wh} < \min\left(|V_{reset}|, |V_{set}|\ \right)$$

Working principle - Evaluation steps for each of these gates are explained individually. Table 3.9 shows the steps for (DFO) COPY function. Case I *i.e.* p=0 implies V_{qx} is not sufficient ($\sim \frac{1}{3}V_w$) to switch $Q*$, but have to endure a cycle without switching (endurance case) and hence determines the maximum period. Case II *i.e.* p=1 implies V_{qx} is sufficient ($\sim V_w$) to switch $Q*$ from $0 \rightarrow 1$ (write case) and hence determine the minimum period.

Table 3.10 shows the steps for AND function. Case I, II and III *i.e.* pq=00, 01 and 10 implies V_{sx} is not sufficient ($\sim \frac{2}{3}V_w$, $\sim \frac{1}{2}V_w$ and $\sim \frac{1}{2}V_w$, respectively) to switch $Q*$, but have to endure a cycle without switching (endurance case) and hence determines the maximum period. Case IV *i.e.*, pq=11 implies V_{sx} is sufficient ($\sim V_w$) to switch S from $0 \rightarrow 1$ (write case) and hence determine the minimum period.

Table 3.11 shows the steps for (DFO) NAND function. Case I, II and III *i.e.* pq=00, 01, 10 implies V_{sx} is sufficient ($\sim V_w$) to switch $S*$ from $0 \rightarrow 1$ (write case) and hence determine the minimum period. Case IV *i.e.*, pq=11 implies V_{sx} is not sufficient ($\sim (V_w - V_{wh})$) to switch $S*$, but have to endure a cycle without switching (endurance case) and hence determines the maximum period.

3.2.3 Memristor-Aided Logic (MAGIC)

- **MAGIC NOT**

Shahar *et al.* proposed family of MAGIC logic logic designs [50] that can perform all the primitive logic functions, namely NOT, AND, OR, NOR and NAND. However, MAGIC-based AND, OR, NAND logic gate configurations does not support crossbar structure and hence not discussed here. The structure of MAGIC-based NOT gate, similar to snider COPY design, is shown in Figure 3.12, which consists of two memristors, namely P and Q, with applied supply voltages(stored resistance states) $V_p(p)$ and $V_q(q)$, respectively. The truth table of MAGIC NOT is intuitive ($q = NOTp$) and hence not shown.

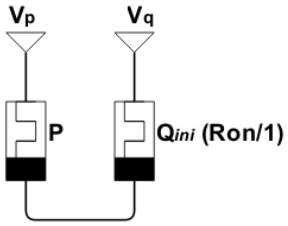


Figure 3.12: MAGIC NOT.

Cycles	Vp	Vq
Ini		Vset
Eva	Vw	GND

Table 3.12: Evaluation steps for MAGIC NOT.

Gate design specification - Evaluation is performed by applying suitable voltages $V_p = V_w$ and $V_q = GND$. Also, the evaluated result is stored in Q (not an input) and therefore this gate configuration is not destructive. The appropriate voltages for correct functionality and ensuring non-destructive are described below.

$$2|V_{reset}| < V_w < \frac{R_{off}}{R_{on}} \left[\min\left(V_{reset}, |V_{set}|\right)\right]$$

Working principle - Evaluation of MAGIC NOT can be performed while considering the two memristors P and Q as voltage dividers (in series) with V_w across them. Evaluation steps are shown in Table 3.12. Memristor Q is initialized to $Ron(1)$ regardless of the input. NOT function is performed by providing $V_p = GND$ and $V_p = V_w$, respectively, abiding to the constraint mentioned above. Case I *i.e.* p=0 implies $V_{xq} \sim 0$, and hence Q remains 1. Case II *i.e.* p=1 implies $V_{xq} \sim> V_{reset}$ (given $V_w > 2 \times V_{reset}$) is equally divided between P and Q), which is enough to switch Q from $1 \rightarrow 0$. Therefore, NOT gate or inversion functionality is achieved.

- **MAGIC NOR**

MAGIC NOT gate configuration can be extended to support more (2^+) inputs, thereby performing an inversion of logical disjunction (OR) of all the inputs *i.e* NOR function. The structure of a MAGIC-based 2-*input* NOR gate is shown in Figure 3.13, which consists of three memristors, namely P, Q and S, with applied supply voltages(stored resistance states) $V_p(p)$, $V_q(q)$ and $V_s(s)$, respectively. The truth table of MAGIC NOR is intuitive ($s = NOT\ (p\ OR\ q)$) and hence not shown.

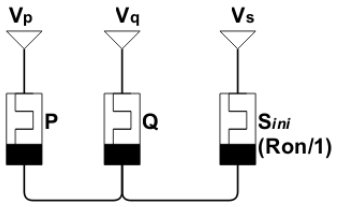


Figure 3.13: MAGIC NOR.

Cycles	Vp	Vq	Vs
Ini			Vset
Eva	Vw	Vw	GND

Table 3.13: Evaluation steps for MAGIC NOR.

Gate design specification - Evaluation is performed by applying suitable voltages $V_p = V_q = V_w$ and $V_s = GND$. Also, the evaluated result is stored in S (not an input) and therefore this gate configuration is not destructive. The appropriate voltages for correct functionality and ensuring non-destructive are described below.

$$2|V_{reset}| < V_w < \min\left[\left(\frac{R_{off}}{2R_{on}}V_{reset}\right), |V_{set}|\right]$$

Working principle - Evaluation of MAGIC NOR can be performed while considering the three memristors as voltage dividers ($(P//Q)$ and S) with V_w across them. Evaluation steps are shown in Table 3.6. Memristor S is initialized to Ron regardless of the inputs and evaluated by providing $V_s = GND$ and $V_p = V_q = V_w$, respectively, abiding to the constraint mentioned above. Case I *i.e.* pq=00 implies $V_{xs} \sim 0$, and hence S remains 1. Case II and III *i.e.* pq=01 and 10, implies $V_{xs} \sim> V_{reset}$ (given $V_w > 2\times V_{reset}$) is equally divided among $(P//Q)$ and S, which is enough to switch S from $1 \rightarrow 0$. Case IV *i.e.* pq=11 implies $V_{xs} \sim> \frac{4}{3}V_{reset}$ is divided among $(P//Q) = R_{on}/2$ and $S = R_{on}$), which is enough to switch S from $1 \rightarrow 0$. Therefore, 2-*input* NOR gate or inverted disjunction functionality is achieved. As mentioned earlier, this configuration can be extended to $2^+ - input$ NOR gate in a similar fashion, but it requires complex constraints.

NOTE - *MAGIC-based logic designs require models with special constraints mentioned above. However, unavailability of analytical models to support such constraints and to avoid mismatch for using models other than TiO_2-based Aachen (analytical) model (behavioral models with materials other than TiO_2-based memristors), MAGIC-based logic designs are not considered for benchmarking.*

CiM-Ah

As described earlier, CiM-Ah primitive logic designs have hybrid locations of inputs *i.e.* atleast one of the inputs as a resistance value in the crossbar array and rest of the inputs as voltage values applied through a periphery circuit (a voltage drivers). However, output is located in the crossbar array as a resistance value. Primitive logic design discussed under this category is programmable logic-in-memory (PLiM) based majority logic (MAJ) implementation. Advantages common to this category of logic designs are -

- Intermediate output in a cascaded set of operations does not require additional read and write operation.
- Sensing circuitry is required only once *i.e.* for the final output after the last evaluation. This feature is exploited for extracting parallelism.

However, these designs have major disadvantages, namely -

- Constant writing during evaluation phase reduces device endurance.
- State machines to regulate control voltages utilize large area and consume high power.

3.2.4 MAJ

Resistive switching logic design based two structures *i.e.* binary resistive switching (BRS) and complementary resistive switching (CRS) implement the majority (MAJ) logic function [51]. BRS logic design is prone to sneak path issue but can be solved by either using a rectifying memristor or using CRS logic design. (The details of the above is illustrated under BRS logic design.) Sneak path issue can be described as an unwanted leakage path formed via half-select memristors storing *Ron* in a crossbar array configuration, as shown on Figure 3.14. Since having a pass transistor (NMOS) in a memristor bit-cell (1T-1R) eliminates this issue, only 1T-1R based BRS logic design is illustrated in detail (and is used for benchmarking later). Nevertheless, CRS design is briefly discussed along with BRS.

- **BRS**

Gaillardon *et al.* proposed BRS logic design that can perform MAJ logic while performing primitive logic functions such as NOT, AND, OR, NOR, NAND, IMPLY etc. in multiple cycles. Here, MAJ logic implementation is illustrated using BRS, whereas other (primitive) logic functions are performed using same (BRS) structure but with some additional cycles operating on specified (and uniquely ordered) input sets. The structure of BRS logic design is shown in Figure 3.15(left). It consists of a single memristor Z, storing one of the inputs (z) as $R_{on}(1)$ or $R_{off}(0)$ resistive value, while the other inputs (p and q) determine the voltage applied as $V_{in}(1)$ or $0(0)$ across the top and bottom electrode (V_p and V_q, respectively). MAJ is a *3-input* (binary) logic function,

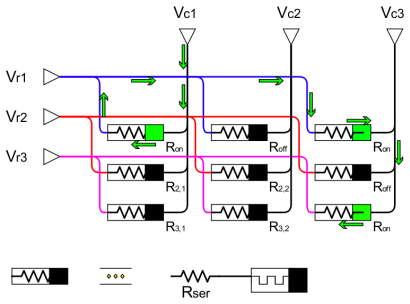


Figure 3.14: Sneak path in suffered in a passive crossbar memory structure. Green memristors correspond to *Ron* (ON) state.

also known as median operator, whose output is the same as the value of input occurring majority of the times (here, ≥ 2). The output (resistance) is represented as (z_{new} =) $MAJ\,(p, \bar{q}, z) = p\bar{q} + \bar{q}z + zp$ in the memristor Z and is shown in Table 3.14.

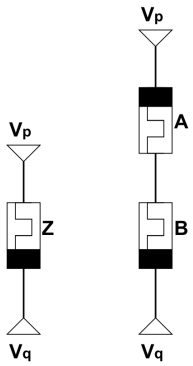


Figure 3.15: BRS (left) and CRS.

z	p	q	z_{final} $MAJ(p, \bar{q}, z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Table 3.14: Truth table MAJ.

Gate design specification - Evaluation is performed by applying inputs p and q as V_p and V_q, respectively, with voltage values of $\max(V_{set}, V_{reset})$ or GND. The appropriate voltages for correct functionality is $V_{reset} > PVth$ and $V_{set} > |NVth|$, implying that voltage applied in either direction should be enough to switch Z to either state.

Working principle - Evaluation of MAJ is a simple write operation, as for $z = 0(1)$, $V_{set}(V_{reset})$ is applied across Z (with the other node at GND). Memristor Z is initialized to $Roff(Ron)$ for $z = 0(1)$. Elaborating on these cases, as shown in Table 3.14 for $z = 0$, Z switches from $0 \rightarrow 1$ only when pq=10. Likewise for $z = 1$, Z switches from $1 \rightarrow 0$ only when pq=01. These cases are the write cases, therefore determine the minimum period. Apart from these cases, the voltage across Z is either 0 (pq=00, 11) or strengthening the already stored data (pq=01 for z=0 and pq=10 for z=1). In other words, BRS configuration can not encounter an unwanted write.

Figure 3.15(right) shows CRS logic design. In brief, state AB = 01 and 10 can be

considered $\sim Z_{BRS} = 0$ and 1, respectively, while the switching cases remain the same and therefore, so does the truth table. However, as already mentioned, using a pass transistor can make the use of CRS redundant and hence is not used for benchmarking later.

CiM-Pa

CiM-Pa primitive logic designs have their inputs located within the crossbar array as resistive values while their output is located (or realized) in the periphery circuit such as a sense amplifier or more generally an analog-to-digital converter (ADC). The primitive logic designs proposed under this category mostly differ in the type of ADC used while the concept remains the same. For example, scouting logic [47] and Pinatabo [82] have a similar approach to execute a logic function, while using current-based and voltage-based sense amplifiers, respectively. Advantages common to this category of logic designs are -

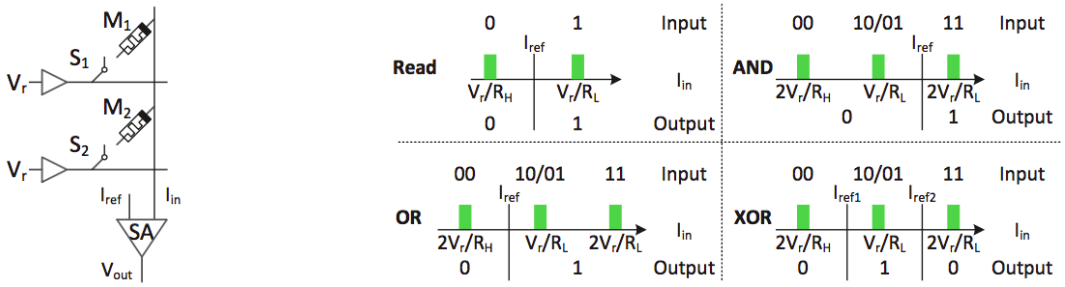
- No writing is involved to execute a logical function in the memristor, thereby increasing endurance.
- Reduction area and power consumption is possible due to reduced control circuitry (state machines) and voltage of operation, particularly since sensing (or reading) voltage can be very low as compared to write (or switching) voltage [61]. Additionally, only a single voltage supply is required for evaluating a logic function.
- Can be easily configured for higher fan-ins.

However, these designs have major disadvantages, namely -

- Intermediate output in a cascaded set of operations does require additional read and write operation.
- Sensing circuitry is required for all intermediate outputs during these set of operations.

3.2.5 Scouting Logic

Xie *et al.* proposed scouting logic design [47] to perform primitive logic functions such as OR, AND and XOR but can be easily extended to NOR, NAND, NOT and XNOR with suitable modifications. Additionally, this design technique allows 2^+ fan-ins as well. The structure of a general $2-input$ scouting logic design is shown in Figure 3.16a. The inputs are stored in memristors P and Q as resistive values p and q, respectively. Output of the intended logic function is a digital output d given by a sensing periphery circuit. Truth tables are not shown since all the performed logic functions are well-known.



(a) Evaluation setup.

(b) Different logic design configurations.

Figure 3.16: Scouting logic design.

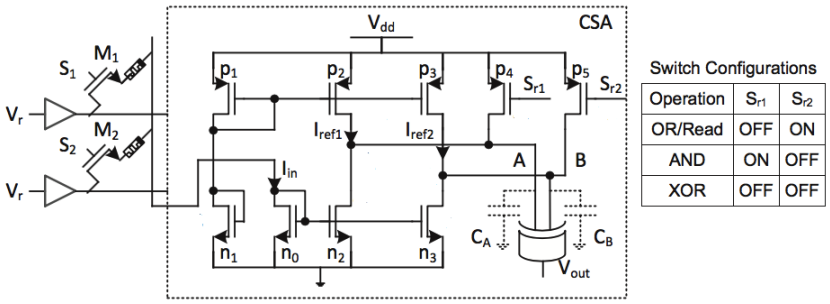


Figure 3.17: Current sense amplifier (CSA).

Gate design specification - Evaluation is performed on resistive input values p and q as $0(Roff)$ or $1(Ron)$. Read voltage (V_r) is applied at the bottom terminals (through a NMOS pass gate, not shown here for clarity), as shown in Figure 3.16a. Total current ($\frac{V_r}{(p//q)}$) is sensed by a current sense amplifier (CSA), as shown in Figure 3.17. The constraints for correct functionality include $V_r << |NVth|$ and ratio $Roff/Ron$ such that CSA can differentiate critical current values of ($\frac{V_r}{Roff/2}$), ($\frac{V_r}{(Ron//Roff)}$) and ($\frac{V_r}{Ron/2}$). This ensures the working of all primitive logic functions (OR, AND, XOR etc.) while using same sized CSA and topology.

Working principle - Evaluation of any scouting logic function is a simple read operation performed simultaneously on two rows (or 2^+ rows for a $2^+ - input$ logic). Subsequently, total current is sensed by CSA, while providing different switches to per-

form different logic functions, as shown in Figure 3.17. In other words, these switches change the reference currents necessary to perform different logic functions, as explained in Figure 3.16.

3.3 Benchmarking of Primitive Logic Designs

This section presents quantitative benchmarking of logic designs illustrated above, while taking into account control and other peripheral circuitry utilized by these designs. The purpose of this section is to identify the best logic design configuration that can be used as a building block for primitive and complex arithmetic and functional units. Key learning and pointers are also discussed.

3.3.1 Benchmarking Criteria and Measuring Methodology

The analyses is performed using Aachen Verilog-A model (Section 2.2). As mentioned earlier, the required regulation of operating voltage during initialization and evaluation phase is still based on CMOS device technology. Therefore, 90nm-TSMC CMOS device models are utilized to simulate these CMOS-based control circuitry. Benchmarking criteria include comparison of logic designs against four key design efficiency metrics, taking into account memristor device, CMOS device and voltage variations. The design metrics and their measuring methodology are as follows.

- **Key Design Metrics** :-
  1. **Latency per operation** (L_t) - Minimum time needed to perform a logic operation.
  2. **Resistance ratio** ($Roff/Ron$) **tolerance** - Tolerated variations in resistance ratio while ensuring correct functionality of a logic design.
  3. **Voltage tolerance** - Tolerated variations in operating voltage(s) while ensuring correct functionality of a logic design.
  4. **Energy per operation** (E_t) - Energy consumption during the logic operation.
- **Setup and Measurement Methodology** :-
  1. Piece-wise-linear (PWL) description of voltage applied during IMPLY (pq=00) is shown (as an example) in Figure **??**. Alternate evaluation followed by read operation is performed with relaxed period of 1μs each.
  2. Read operation is performed by providing $V_q = V_{read}$ and sensing the current through SA. An ON state has a current of nearly $\frac{Roff}{Ron}\times$ OFF state current, thereby distinguishing between the two states.
  3. $Roff$ is normalized with respect to Ron *i.e.* Ron=1 and $Roff = Roff/Ron$. Nominal $Roff/Ron$ chosen for the benchmarking are 10, 50, 100, 200, 500 and 1000. Since resistance variations are taken into account, the analyses is fully exhaustive *i.e.* complete resistance ratio range is covered.

4. Every logic design is evaluated for an exhaustive range of operating voltages (incremental step of 0.1V) applied on all nominal $Roff/Ron$ individually. For any particular $Roff/Ron$ that passes, device parameters N_{min}, N_{max}, l and r_d are varied with equal proportion (%) until it fails. Final variation percentage (for which a logic function passes) provides the extent of device and $Roff/Ron$ variations tolerated by a logic design at a particular voltage and nominal $Roff/Ron$ value.
5. For logic designs that require switching, L_t corresponds to the time required by the output memristor to reach within 2-3% of the destined value. Rest of the cases, it corresponds to minimum time required to successfully read-out the output memristor state.
6. E_t is calculated as the product of all (n) voltages applied (V_i) and time-integral of the current (I_{it}) supplied by these voltages *i.e.* $\sum_1^n (V_i \times \int_0^T I_{it} dt)$ during the time-constrained switching cycle. For read (access) based functions, apart from the voltage (integral) current product mentioned above, energy consumed by SA is also included for the constrained read cycle.

3.3.2 Voltage, Memristive Device and CMOS Variations

As mentioned earlier, voltage variations are accounted (by applying incremental operating voltage of step size 0.1V) while checking for correct functionality. In order to account for CMOS device variations, 100 monte-carlo (MC) are simulated for all process corners *i.e.* TT, SS, SF, FS and FF, while temperature is 27C. However, unlike CMOS models, memristor device models does not support in-built variations. Therefore, accounting for variability in memristor devices in a logic design follows the following approach.

Case $Roff/Ron$=100 is taken as an example to explain the variability approach. This is however done for all six nominal $Roff/Ron$ values. $Roff/Ron$=100 dictates values of physical device parameters N_{min}, N_{max}, l and r_d to be $0.0267(\times 10^{26})cm^{-3}$, $10(\times 10^{26})cm^{-3}$, $1(\times 10^{-9})$m and $32(\times 10^{-9})$m, respectively. Since these parameters are mutually exclusive, the idea is to vary each of these parameters individually yet independently and simulate a logic design with all possible varied set of combinations. However, since changing N_{max} and l by 30% has a negligible effect on key device metrics, as shown in Section 2.3.2, only N_{min} and r_d are varied for this analyses.

A parameter is varied percentage-wise and as a example, 30% variation in N_{min} is presented in Table 3.15 (highlighted in red). The nominal value of N_{min} corresponding to selected $Roff/Ron$ are shown, with 30% variation in N_{min} with resulting range of $Roff/Ron$ variation.

For instance, IMPLY logic design is configured in the manner as shown in Figure 3.18, consisting of two memristors P and Q, with N_{minp} and N_{minq}, respectively. All possible combinations of two extreme parametric values ($N_{min(-30\%)}$ and $N_{min(+30\%)}$) are simulated for IMPLY, as shown in Table 3.16. Parametric values simulated to ensure 30% variation tolerance is highlighted in red.

Variations in r_d are also taken into account in a similar way and is combined with variations in N_{min} to form a complete account of variability. If all such parametric combinations pass for IMPLY, it is substantiated that a logic function can performed

$Roff/Ron$	N_{min}	$N_{min(-30\%)}$	$N_{min(+30\%)}$	$(Roff/Ron)_{max}$	$(Roff/Ron)_{min}$
10	0.09	0.063	0.117	17	7
50	0.0364	0.02548	0.04732	112	29
100	0.0267	**0.01869**	**0.03471**	237	56
200	0.02	0.014	0.026	485	107
500	0.01382	0.009674	0.017966	1212	261
1000	0.0104	0.00728	0.01352	2397	529

Table 3.15: +/-30% variation in N_{min} for each of the selected nominal $Roff/Ron$. $Roff/Ron$=100 highlighted in red.

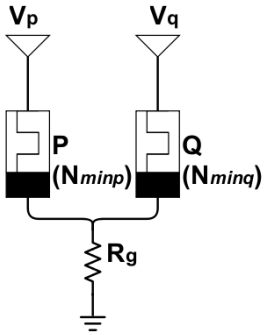


Figure 3.18: IMPLY, as an example, to show the methodology used to account for parametric (here, N_{min}) variations.

against 30% parametric variations with correctly functionality at V_{set} and V_{cond} equal to 1.7V and 0.85V, respectively. As mentioned earlier, this analyses is done for an exhaustive voltage range and results and comparison in detail is presented next.

3.3.3 Results and Critical Analyses

Shmoo plots are compiled for all the logic designs explored in the Section 3.2. Table 3.17 shows the shmoo plot for IMPLY logic design as an example. Each element in the table corresponds to parametric variations tolerance for a given nominal $Roff/Ron$ with a particular voltage(s) of operation. To avoid congestion, rest of the shmoo plots are in appendix.

Final cumulative result is condensed in Table 3.18. The table provides briefly key efficiency metrics for all logic designs illustrated previously. Explanation of the cumulative table, reflection on the final result, justification supporting these results and comparison are presented as follows.

V_{set}	V_{cond}	$Variation\%$	$Roff/Ron$	N_{minp}	N_{ming}
1.7	0.85	10%	100	0.02403	0.02403
1.7	0.85	10%	100	0.02403	0.02937
1.7	0.85	10%	100	0.02937	0.02403
1.7	0.85	10%	100	0.02937	0.02937
1.7	0.85	20%	100	0.02136	0.02136
1.7	0.85	20%	100	0.02136	0.03204
1.7	0.85	20%	100	0.03204	0.02136
1.7	0.85	20%	100	0.03204	0.03204
1.7	0.85	30%	100	**0.01869**	**0.01869**
1.7	0.85	30%	100	**0.01869**	**0.03471**
1.7	0.85	30%	100	**0.03471**	**0.01869**
1.7	0.85	30%	100	**0.03471**	**0.03471**
1.7	0.85	40%	100	0.01602	0.01602
1.7	0.85	40%	100	0.01602	0.03738
1.7	0.85	40%	100	0.03738	0.01602
1.7	0.85	40%	100	0.03738	0.03738

Table 3.16: 10%, 20%, 30% and 40% variation in N_{min} for V_{set} and V_{cond} (IMPLY in Figure3.18) equal to 1.7V and 0.85V, respectively.

- **Interpretation of the cumulative comparison table** :-

  1. For each design, minimum possible nominal $Roff/Ron$ providing maximum tolerance is referred, with corresponding operating voltage range.
  2. Only scouting logic designs involve CMOS devices (in SA) to execute logic functions. *i.e.* CMOS-based CSA distinguishes the total current through the memristors storing inputs to provide an output of a logic function. Therefore, 100 MC simulations (at TT-27C) are run to ensure for correct functionality of CSA (Figure 3.17).
  3. Latency (in ns) indicates the evaluation time required for a logic design for the nominal $Roff/Ron$ (case mentioned in the second column). For scouting logic designs, it is the resolving time for CSA while for the rest, it indicates the time to write the destined value in the output memristor(s).
  4. Energy (in pJ) indicates the energy consumption corresponds to the nominal $Roff/Ron$ (case mentioned in the second column). For scouting logic designs, it is the average read energy plus energy consumed by CSA to resolve the current while for the rest, it is the energy consumed during output memristor's state switching.

- **Key reflections on comparison** :-

  1. Lowest voltage that can support scouting logic designs is determined by the ability of CSA to accurately distinguish the current values. In other words, voltage supply should be sufficient enough such that CSA differentiates among

V_{set} \ $Roff/Ron$	10	50	100	200	500	1000
1.1	F	F	F	F	F	F
1.2	F	F	F	F	F	F
1.3	F	F	F	F	F	F
1.4	F	F	F	F	F	F
1.5	F	F	F	F	F	F
1.6	F	F	F	F	F	F
1.7	P (2%)	P (5%)	P (35%)	P (20%)	P (2%)	F
1.8	F	F	P (15%)	P (35%)	P (20%)	P (3%)
1.9	F	F	P (3%)	P (35%)	P (30%)	P (15%)
2	F	F	F	P (20%)	P (40%)	P (25%)
2.1	F	F	F	P (3%)	P (40%)	P (35%)
2.2	F	F	F	F	P (30%)	P (45%)
2.3	F	F	F	F	P (20%)	P (40%)
2.4	F	F	F	F	P (3%)	P (40%)
2.5	F	F	F	F	P (1%)	P (30%)
2.6	F	F	F	F	F	P (20%)
2.7	F	F	F	F	F	P (15%)
2.8	F	F	F	F	F	P (3%)
2.9	F	F	F	F	F	F
3	F	F	F	F	F	F

Table 3.17: Shmoo plot for IMPLY. V_{set} is shown in the first column while $V_{cond} = V_{set}/2$, since it provides the best result.

Logic Design	Nom. Roff/Ron, Max. Tolerance	Voltage, Range	CMOS Variation	Latency ns	Energy pJ
Scouting OR	>3.4	0.3 - 1.7	Pass***	0.38	0.22
Scouting AND	>4	0.8 - 1.7	Pass***	1.6	0.7
Scouting XOR	>4.5	0.8 - 1.7	Pass***	1.6	0.67
IMPLY	~500, 40%	1.7 - 2.4	NA	375	53
Snider NOT	~500, 40%	1.7 - 2.4	NA	375	53
Snider COPY	>10, 35%	0.7 - 1.2*	NA	75 / 8**	15 / 5**
Snider NAND	>100, 35%	1.3 - 2.5*	NA	32 / 4**	9/2**
FBL COPY	>10, 35%	1 - 1.7*	NA	7 / 3**	4 / 2**
FBL NAND	>70, 40%	1.3 - 2.5*	NA	100 / 30**	35 / 6**
MAJ	7, 60%	>0.8	NA	40 / 5**	40 / 6**

Table 3.18: Cumulative comparison for all primitive logic designs. *Average range voltage for any $Roff/Ron$ is $\sim 600mV$, **Low/High Operating Voltage, ***Pass with 100 Monte Carlo at TT 27C, NA - No CMOS device used for logic design functionality.

the voltage levels at the CMOS device (NMOS $N0$ in Figure 3.17), where this voltage level depends on the current flowing through it. The size of this NMOS

is kept as large as possible to provide minimum resistance in the path. This ensures minimal influence of $N0$ resistance on the current under consideration (basic current sense amplifier requirement).

2. Highest voltage supporting scouting logic designs is determined by read disturb failure of the memristors storing the resistance input values. Therefore, for given parametric values of the memristor, voltage applied should be less than $|NVth|$.
3. Scouting AND and XOR logic designs support less voltage range when compared to Scouting OR. This is due to the fact that differentiating $pq = 01/10$ ($Roff//Ron \sim Ron$) and $pq = 11$ ($Ron//Ron = Ron/2$) case is more challenging as compared to differentiating $pq = 00$ ($Roff//Roff = Roff/2$) and $pq = 01/10$ ($\sim Ron$).
4. After a certain mem-resistance is reached, increasing $Roff/Ron$ (through varying parametric values) may not impact in scouting tolerance. It is due to the fact that once ($Roff//Ron \sim Ron$) and ($Ron//Ron = Ron/2$) or ($Roff//Roff = Roff/2$) and ($\sim Ron$) can be differentiated by CSA, a higher $Roff/Ron$ will certainly be differentiated.
5. Worst (extreme) cases for CiM-A logic designs, say IMPLY or IMPLY-NAND (or for that matter any stateful logic), are write and endurance cases. Write (switching) case *i.e.* when $pq = 00$ tends to fail at low voltages for high $Roff/Ron$ values, while in the endurance case *i.e.* when $pq = 10$, tends to fail at high voltages for low $Roff/Ron$ values. It is apparent due to the fact that memristors with low $Roff/Ron$ values have lower $|NVth|$ and vice versa. This effectively provides an upper and lower bound of working voltages of operation. The above explanation is reflected in the result of IMPLY logic design for $Roff/Ron$=500, as shown in Table 3.19.
6. As proposed by Xie, FBL-COPY(MFO version of Snider-COPY) does not require R_g, since it does not provide any added value.
7. Worst (extreme) possible cases for Snider/FBL-COPY share the same explanation as point 5. In case of coping 1 *i.e.* $pq = 10$, the design tends to fail at low voltages for high $Roff/Ron$ values, while in case of coping 0 *i.e.* $pq = 00$, it tends to fail at high voltages for low $Roff/Ron$ values.
8. Range of operating voltage for FBL-COPY(MFO) is more as compared to Snider-COPY(SFO). Although, lower voltage limit of FBL-COPY is slightly increased (corresponding limiting case is copying 1 *i.e.* $pq = 10$), as initial values of q (effective $q_{FBL} = Roff/2$ and $q_{Snider} = Roff$) implies that for a given applied voltage (V_q), less voltage across Q_{FBL} is available to readily switch Q from $0 \rightarrow 1$. But, on the other hand, higher voltage limit of FBL-COPY is greatly increased (corresponding limiting case is coping 0 *i.e.* $pq = 00$), implies that less voltage across Q_{FBL} makes Q less prone to switch from $0 \rightarrow 1$. Quantitatively, $V_{min(FBL)} \sim 2 \times V_{min(Snider)}$ but $V_{max(FBL)} \sim 2 \times V_{max(Snider)}$, and since $V_{max(Snider)} > V_{min(Snider)}$, range $(V_{min(FBL)}, V_{max(FBL)}) > (V_{min(Snider)}, V_{max(Snider)})$. This is substantiated from their respective shmoo plots (in appendix).

V_{set}	V_{cond}	$pq = 00$ ***(write case)***	$pq = 10$ ***(endurance case)***
1.3	0.65	F	P
1.4	0.7	F	P
1.5	0.75	F	P
1.6	0.8	F	P
1.7	0.85	P	P
1.8	0.9	P	P
1.9	0.95	P	P
2	1	P	P
2.1	1.05	P	P
2.2	1.1	P	P
2.3	1.15	P	P
2.4	1.2	P	P
2.5	1.25	P	F
2.6	1.3	P	F
2.7	1.35	P	F
2.8	1.4	P	F
2.9	1.45	P	F
3	1.5	P	F

Table 3.19: IMPLY logic design with extreme cases *i.e.* write case ($pq = 00$) and endurance case ($pq = 10$) at $Roff/Ron$=500.

9. FBL-NAND (MFO) and Snider-NAND (SFO) have nearly similar voltage and resistance tolerance range.

10. Stateful logic designs *i.e.* IMPLY/Snider/FBL are slow as voltage across output memristor reduces when it shifts from $0 \rightarrow 1$ logic state. In other words, say in case of IMPLY with pq=00, as it approaches its destined value ($q_{final} = Ron$), the available write voltage across Q diminishes as compared to the beginning ($q_{ini} = Roff$) of the cycle.

- **Logic Design Selection for building arithmetic and functional units** :- Scouting logic designs does not require writing (switching) a memristor. This results in the following improvements.

  1. **Endurance** - Read endurance is three orders higher as compared to write endurance [61]. Therefore, switching a memristor reduces its endurance significantly as compared to simply accessing (reading) it. This is due to the fact that operating voltage applied in scouting logic is relatively small $V_{read} << V_{write}$. For scouting OR logic design, V_read can be as low as 0.3-0.4V, which is enough to warrant to quantify the current under consideration *i.e.* distinguishing current flowing through $V_{read}/(Roff/2)$ and

$V_{read}/(Ron//Roff)(\sim Ron)$. A higher voltage or larger NMOS is required if higher fan-in ($2^+ - input$ scouting logic) is built.

2. **Speed** - Read operation is much faster than write operation [61]. As mentioned earlier, writing voltage reduces across the target memristor during the switching process, thereby slowing the process. Additionally, besides switching to get the output resistance in a memristor, the final output needs sensing (reading out) anyway, to be used in the next cycle (in case cascading is not desired).
3. **Energy** - Since scouting logic designs require less time and lower operating voltage, these designs are highly energy(power)-efficient.
4. **Area** - Scouting logic designs do not require extra control (peripheral) circuitry that regulates different set of voltages to ensure functionality. Hence, these designs are highly area-efficient as well.
5. **Reliability** - Increased parametric variation tolerance. The working principle of scouting logic designs is such that the functionality has negligible dependency on $Roff/Ron$ values (say for $Roff/Ron > 5$).

The quantitative benchmarking concludes that **scouting OR** (effectively including AND and XOR) logic design is the most suitable primitive logic design to be utilized as the building block for arithmetic and functional circuit designs.

Building Primitive Arithmetic Circuit Designs

4

Primitive arithmetic functions in modern computing systems are addition, subtraction, multiplication and division. All computing hardware units can be essentially broken down to these primitive circuit designs as their building blocks. However, if the algorithms of the later three functions are observed, their fundamental functional unit is addition. Subtraction (while working with most widely used binary/radix-2 input representation), say $A-B$, is a special case of addition i.e. adding A with 2's complement of B and a carry-in at LSB. Multiplication, say $A \times B$, involves multiple shift-and-add operations on partial products. Division, say $A \div B$, is performed by algorithms such as reciprocation and successive multiplication ($A \times \frac{1}{B}$), restoring/non-restoring division (shift-and-subtract) or division by convergence ($q = \frac{z}{d}$, converging $z \to q$ such that $d \to 1$), with addition and multiplication (which in turn is addition based) as fundamental functional units. Hence, it can be substantiated that addition is the prime arithmetic function for any computing system. Consequently, building memristor-based efficient adder circuit designs would strengthen the promise of memristor device technology to support in-memory-computing architectures.

Several memristor based adder circuits are proposed, with primitive logic designs described in the Section 3.2 as building blocks to perform addition. However, these solutions are inefficient in terms of speed, peak power, overall energy consumption, area utilization and/or reliability against device variations. In this regard, a novel adder circuit design is proposed that outperforms existing circuit design solutions in terms of key efficiency metrics.

The organization of this chapter is as follows. Section 4.1 illustrates existing CiM-A and CiM-P logic design-based adder circuit designs, while pointing out major inefficacies. Section 4.2 introduces a novel adder circuit, with detailed description of each of the key features incorporated, that improves overall efficiency metrics. Section 4.3 compares the design efficiency of proposed 4-bit adder design (as a case study) with existing 4-bit adder designs. Also, extension to multi-operand addition (3, 4 operands added simultaneously) is presented to showcase the scalability of the proposed design.

4.1 Overview of Existing Adder Circuit Designs

This section illustrates existing adder circuit designs, classified based on the class of logic design used as a building block. The main idea and key features presiding each circuit design is covered briefly while describing .

4.1.1 Using CiM-A (Non-Scouting) type Logic Designs

- **Using Boolean Logic**

Xie *i.e.* proposed *1-bit* full adder (FA) based on fast boolean logic (FBL) logic design [7] (illustrated in Section 3.2). The premise of the subsequently built *4-bit* adder circuit is concatenating four such FAs in a ripple carry fashion. *1-bit* FA follows the approach of sum-of-products (SOP) *i.e.* adding appropriate min-terms to give sum and carry bits as outputs. The following equations describe well-known FA operation of adding two *1-bit* inputs and a carry in *i.e.* $A + B + C_{in}$ with sum(S) and carry(C_o) as outputs.

$$S = \overline{\bar{A}\bar{B}C_{in}} \cdot \overline{\bar{A}B\bar{C}_{in}} \cdot \overline{A\bar{B}\bar{C}_{in}} \cdot \overline{ABC_{in}} \tag{4.1}$$

$$C_o = \overline{\bar{A}BC_{in}} \cdot \overline{AB\bar{C}_{in}} \cdot \overline{A\bar{B}\bar{C}_{in}} \cdot \overline{ABC_{in}} \tag{4.2}$$

Figure 4.1 shows the implementation of one FA within memristor-based crossbar structure. Bit-wise inputs and their inverted valued inputs *i.e.* A, $\bar{A}$, B, $\bar{B}$, C and $\bar{C}$ are programmed (as mem-resistances) in the first row and are broadcasted within the array (MFO COPY) by column write drivers at appropriate locations. Thereafter, the required minterms *i.e.* $\bar{A}\bar{B}C_{in}$, $\bar{A}B\bar{C}_{in}$, $A\bar{B}\bar{C}_{in}$, $\bar{A}BC_{in}$, $AB\bar{C}_{in}$, $AB\bar{C}_{in}$ and ABC_{in} are obtained by performing conjunction operation (using FBL-based $3-input$ AND logic) with appropriate bit-wise input combinations. Finally, these minterms are combined by performing inverted disjunction (using FBL-based $4-input$ NOR logic) to provide $\bar{S}$ and $\bar{C}_o$. The final outputs are inverted (using FBL-based NOT logic) to get S and C_o.

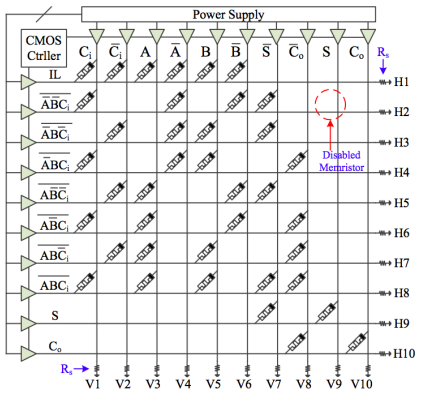


Figure 4.1: FBL-based *1-bit* FA [7].

Table 4.1 shows the configuration of control voltages to perform *1-bit* FA. All possible elements ($1-bit$ inputs and inverted values) A, $\bar{A}$, B, $\bar{B}$, C, and $\bar{C}$ are initialized during INA. RIN ensures memristors in the crossbar array have received $1-bit$ inputs and inverted values, CFM involves configuration of min-terms, EVM corresponds to evaluation of the min-terms, GER and INR generates the inverted and un-inverted final outputs S and C_o, respectively. SOU sends the outputs to the next FA as inputs, enabling ripple carry addition.

Phase	Control Voltages					
	Row			Column		
	Input Latch	Logic Block	Output Latch	Inputs	Outputs	
	H1	H2-8	H9,10	V1-6	V8,10	V7,9
INA	Vw	Vw	Vw	G	G	G
RIN	G	Vh	Vh	F	Vh	Vh
CFM	Vw	G	Vh	F	Vh	Vh
EVM	Vh	F	Vh	Vh	Vh	Vw
GER	Vh	Vw	G	Vh	Vh	F
INR	Vh	Vh	F	Vh	Vw	Vh
SOU	Vh	Vh	Vw	Vh	Vh	Vh

Table 4.1: Control voltages to evaluate 1-bit FA [7].

Figure 4.2 shows a well-known sequentially operated *4-bit* ripple carry adder structure. Final sum array ($S[3:0]$) and carry out (C_o) is obtained by repeating (sequentially) the above operation four times.

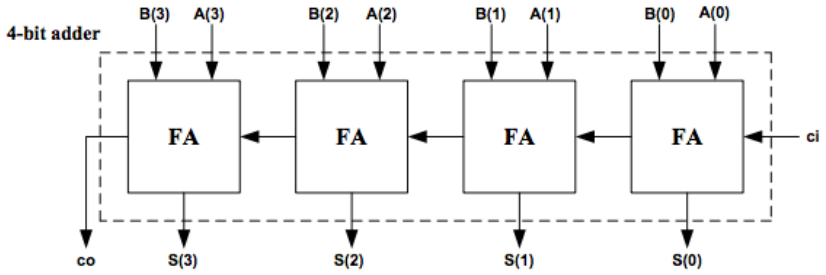


Figure 4.2: *4-bit* ripple carry adder.

- **Using IMPLY**

Shahar *et al.* proposed *1-bit* full adder (FA) based on material implication logic design (IMPLY) [8] (illustrated in Section 3.2). Two designs approaches *i.e.* serial and parallel design techniques proposed to perform *8-bit* addition claims to outperform other stateful implication logic family based adder designs. Unlike FBL-based adder with ripple carry design approach, serial *8-bit* addition is performed in a bit-serial fashion. Whereas, parallel adder circuit consists of 8 FAs connected in a complex fashion (differs from crossbar structure) to communicate in a parallel manner. *1-bit* FA follows the topology of basic CMOS-based FA with two XOR gates, two AND gates and an OR gate *i.e.* $S = A \otimes B \otimes C_i$ and $C_o = (A \cdot B) + (C_i \cdot (A \otimes B))$. Essentially, these gate functionalities are built solely using IMPLY and FALSE (a function that always yields 0 at output using IMPLY logic). As an example, XOR logic is performed in the following manner.

$$A \otimes B = FALSE(M1), FALSE(S), A \rightarrow S, S \rightarrow M1 \Rightarrow M1 = A$$

$$FALSE(M2), FALSE(S), B \rightarrow S, S \rightarrow M2 \Rightarrow M2 = B \tag{4.3}$$

$$B \rightarrow M1, FALSE(S), B \rightarrow S, M1 \rightarrow S \Rightarrow S = \overline{A + \bar{B}}$$

$$A \rightarrow M2, M2 \rightarrow S \Rightarrow S = \overline{\overline{\bar{A} + B} + \overline{A + \bar{B}}} = A\bar{B} + \bar{A}B$$

This execution of XOR function takes 13 computational steps, and since computing S requires two sequential XOR gates, total computation takes 26 steps. Intermediate $A \otimes B$ is also used for C_o, which is executed in the following manner.

$$C_o = \underbrace{\underbrace{(A \rightarrow (B \rightarrow 0))}_{NAND(\overline{A \cdot B})} \rightarrow \underbrace{((C_i \rightarrow ((A \otimes B) \rightarrow 0))}_{NAND(\overline{C_i \cdot (A \otimes B)})} \rightarrow 0)}_{NAND(\overline{\overline{A \cdot B} \cdot \overline{C_i \cdot (A+B)}}) = (A \cdot B) + (C_i \cdot (A \otimes B))} \tag{4.4}$$

Figure 4.3 shows the implementation of a serial *8-bit* adder circuit using 27 memristors (26 steps with 1 memristor each + 1 output memristor) within a single row of a crossbar structure. As parallel approach used complex non-crossbar structure with 8×27 memristors, it is excluded from the discussion.

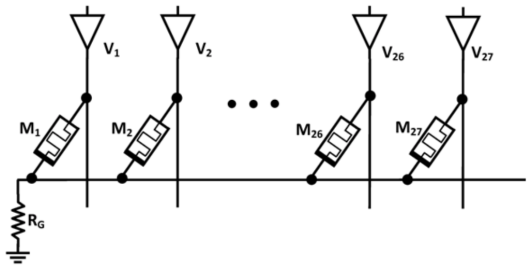


Figure 4.3: Serial IMPLY based $8-bit$ adder [8].

- **Using MAJ**

Gaillardon *et al.* proposed *1-bit* full adder (FA) based on majority function (MAJ, a 3-binary input median function) using programmable logic-in-memory [51] (PLiM, illustrated in Section 3.2). Similar to IMPLY-based serial adder, MAJ-based *2-bit* adder is built in a bit-serial fashion. However, MAJ-based *2-bit* adder does not quite follow conventional CMOS-based FA topology and is built solely using MAJ. Table 4.2 provides the steps to perform *2-bit* serial-addition.

Steps	MAJ *inputs* (p, q, z)	MAJ *output*$=z_{final}$ $MAJ(p, \bar{q}, z)$	Output Expression
1	0, 1, @C;	$C = C_{in} = 0$	Initializing C
2	0, 1, @X;	$X = 0$	Initializing X
3	1, @B_0, @X;	$X = \bar{B_0}$	Inverting B_0
4	@A_0, @X; @C;	$C = C_1$	A_0B_0
5	0, 1, @S_0;	$S_0 = 0$	Initializing S_0
6	@A_0, @B_0, @S_0;	$S_{0int} = S_0$	$A\bar{B}$
7	@B_0, @C, @S_0;	$S_0 = S_{0int}$	$A + AB + A\bar{B}$
8	0, 1, @X;	$X = 0$	Initializing X
9	1, @B_1, @X;	$X = \bar{B_1}$	Inverting B_1
10	@A_1, @X; @C;	$C = C_2$	$A_1B_1 + B_1C_1 + C_1A_1$
11	0, 1, @X;	$X = 0$	Initializing X
12	1, @C_1, @X;	$X = \bar{C_1}$	Inverting C_1
13	@C, @X; @S_1;	$S_1 = C_1$	Copying C_1
14	@A_1, @B_1, @S_1;	$S_{1int} = S_1$	$A_1\bar{B_1} + \bar{B_1}C + CA$
15	@B_1, @C, @S_1;	$S_1 = S_{1int}$	$\bar{A_1}\bar{B_1}C_1 + \bar{A_1}B_1\bar{C_1} +$ $A_1\bar{B_1}\bar{C_1} + A_1B_1C_1$

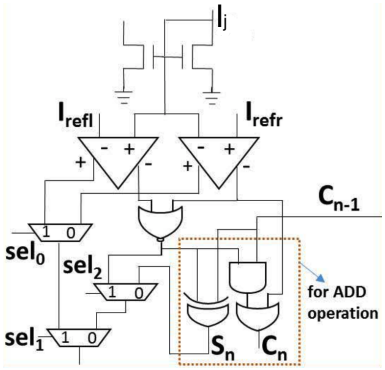
Table 4.2: Computational Steps for MAJ-based *2-bit* adder.

In the table above, A_0, B_0 and $C_0(=0)$ are at 0th position and A_1 and B_1 are at 1st position, with usual meaning. $S_0(C_1)$ and $S_1(C_2)$ are sum (carry-outs) at 0th(1st) and 1st(2nd) positions, respectively. The total computational steps required for this *2-bit* addition is 15.

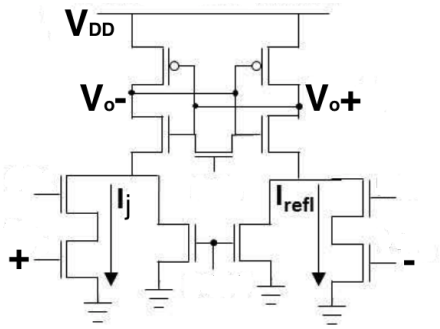
4.1.2 Using CiM-P (Scouting) type Logic Designs

- **Using Voltage Sense Amplifier**

Jain *et al.* proposed a voltage-based sensing circuitry (VSA) [9] as opposed to current-based sensing circuitry (CSA) (illustrated in Figure 3.17) proposed by Xie. The proposed design performs all well-known bit-wise logic functions and read operation, along with in-built *1-bit* FA functionality. The complete circuit, along with the sense amplifier used is shown in Figure 4.4. All the above functionalities can be achieved by toggling I_{refl} and I_{refr} as reference currents as inputs to the two comparators and sel_0, sel_1 and sel_2 as *1-bit* MUX selectors. The current references can assume any combination of $V_{read}/Roff$, V_{read}/Ron and $v_{read}/Rref$ ($Ron < Rref < Roff$) current values. However, only control signal selection required for an adder is illustrated here.



(a) Modified sensing circuitry to perform bit-wise and adder operations [9].



(b) VSA [9].

Figure 4.4: VSA-based sensing circuitry to perform multiple operations [9].

To perform a FA operation, the execution of the following expression (logic functions) narrows down the selection of control signals and reference currents. All equations have parameters with their usual meaning.

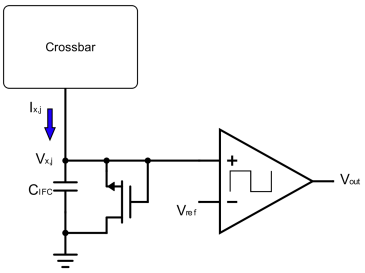
$$S_n = \underbrace{A_n \otimes B_n}_{bitwise-operation} \otimes C_{n-1} \Rightarrow S_n = O_{(\otimes)} \otimes C_{n-1} \tag{4.5}$$

$$C_n = \underbrace{(A_n \otimes B_n)}_{bitwise-operation} \cdot C_{n-1} + \underbrace{(A_n \cdot B_n)}_{bitwise-operation} \Rightarrow C_n = O_{(\otimes)} \cdot C_{n-1} + O_{(\cdot)} \tag{4.6}$$

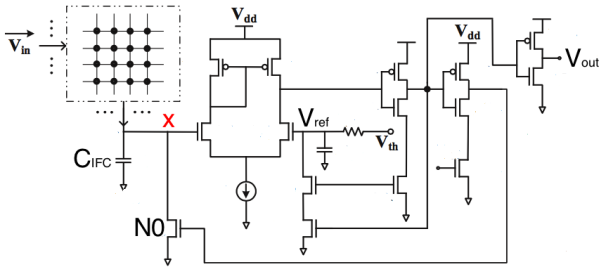
Where, an *N-bit* adder is built in a ripple carry fashion with A_n, B_n (stored as resistances in memristor crossbar array) and C_{n-1} is feed from $(n-1)$th position. All select signals sel_0, sel_1 and sel_2 are 0, while current references I_{refl} and I_{refr} are $V_{read}/(Roff//Rref)$ and $V_{read}/(Ron//Rref)$, respectively. Such control signals allow bitwise-operations mentioned in the equations above. Three additional operations are performed by CMOS-based XOR, AND and OR gates in the periphery circuit, as marked in Figure 4.4a.

- **Integrate and Fire Circuit (IFC)**

Liu *et al.* proposed modified integrate and fire circuit (IFC) [12] to perform in-memory multi-operand arithmetic operations. Figure 4.5 shows the general idea of an IFC-based arithmetic design. The working principle of IFC-based arithmetic unit is to provide number of spikes (or pulses) at the output, depending on the current flowing through a column in a memristor crossbar array. These spikes are counted using a CMOS-based counter, say Johnson counter of bit-size/length $[\log_2(\text{rows}+1)]$. Subsequently, shift-and-add circuit is deployed to integrate the results (# of spikes) from all the columns into one unified digital output.



(a) Analog to digital conversion using IFC.



(b) Detailed IFC sensing circuit [12].

Figure 4.5: Muti-operand addition using IFC as a sensing circuit.

The implementation of an IFC-based arithmetic unit is illustrated as follows. A read/access voltage $V_{read} = V_{in}$ is applied to all the rows of the memristor crossbar array for which arithmetic operation is desired. Current flowing through each column $\sim \frac{V_{in}}{G_{j(eff)}}$ (where $G_{j(eff)}$ effective parallel combination of all active memristor rows in column j) is sensed using an IFC. A gate-drain shorted (node x) NMOS $N0$ receives the column current, which is in parallel with a capacitance C_{IFC} that stores the voltage (V_x) appearing on this node x. Once the rising node V_x reaches V_{ref}, differential amplifier (DA) (V_x, V_{ref} as +,- DA inputs) triggers a spike at the output, while also resetting the node x. This process is repeated again, with the frequency of the spike formation directly proportional to the current flowing through the column. The following expressions relate the number of spikes and the current flowing through a given column j with a general N number of rows per column.

$$I_{x,j}(t) = \sum_{i=0}^{N-1} g_{ij} \left[V_{in,i}(t) - V_{x,i}(t)\right] \tag{4.7}$$

$$I_{x,j}(t) = C_{IFC} \frac{dV_{x,j}(t)}{dt} \tag{4.8}$$

$$\frac{dV_{x,j}(t)}{dt} \approx \frac{1}{C_{IFC}} \sum_{i=0}^{N-1} g_{ij} V_{in,i}(t) \tag{4.9}$$

$$n_{y,j}(t) \propto \int_{\tau=0}^{T_{total}} \sum_{i=0}^{N-1} g_{ij} V_{in,i}(\tau) d\tau \tag{4.10}$$

Here, $V_{in,i}$ and $V_{x,j}$ are voltages at WL_i and BL_j, respectively. The current $I_{x,j}$ flows through BL_j with g_{ij} as mem-resistances at the cross-section of WL_i and BL_j along column j. $n_{x,j}$ is the number of spikes produced for column j. Below is the expression for number of spikes produced by IFC for a given time interval T_{total}.

$$n_{y,j}(t) = \frac{T_{total}}{\Delta t + t_o} \tag{4.11}$$

Here, t_o is the time take by the current to charge C_{IFC} (or node V_x) from $0 \rightarrow V_{ref}$ and Δt is the time taken for IFC to reset the node V_x ($\sim$ time taken from V_x having reached $V_{ref} \rightarrow$ spike generation). The term Δt de-linearizes the relationship between current value ($\propto t_o$) and corresponding number of spikes. The standard IFCs [58, 83, 84] is modified in [12] to improve the accuracy and linearity of outputs with the inputs, as shown in Figure 4.5b. In the proposed modified circuit, M7-M9 provides a positive feedback loop to minimize Δt.

The illustration above is for a general number of rows (N, # of operands) and columns (# of bits per operand). *4-bit* adder design can be built with crossbar memory size of 2×4 (replacing equations with $N = 2$ and $0 \leq j \leq 3$). This implies that four IFCs are employed *i.e.* one for each column, along with individual digital counters of bit-size $[\log_2 3]$.

4.2 Proposed Adder Circuit Design

The previously discussed adder circuit designs, besides IFC-based design, suffer from serious limitations owing to the fact that they are built on inefficient logic designs. Some of the major aspects are qualitatively discussed as follows.

- **Latency** - Besides the fact that several steps are required to compute one FA operation, ripple carry and bit-serial based designs are highly sequential. Hence, these implementations are anticipated to be extremely slow.
- **Area** - Significant area is utilized for a single FA operation. Cascading such units is likely to be highly area-inefficient.
- **Energy** - High operating voltage is required as these designs involve switching of memristor devices. This results in significant power consumption per operation. Coupled with high latency, these designs are expected to be extremely energy inefficient.
- **Tolerance against variations** - Since the building block of such designs *i.e.* CiM-A type logic designs are prone to voltage and memristor device variations (see Section 3.3), arithmetic operations are expected to be susceptible as well.
- **Endurance** - Write endurance is three orders of magnitude less than read endurance [61]. Since these designs involve switching, a diminished lifetime of memristor devices is expected.

The adder circuit based on IFC technique suffers from high latency, area utilization and energy consumption to perform arithmetic functions. For a *4-bit* adder circuit, key drawbacks are discussed qualitatively.

- **High latency** - Following the generation of output spikes by IFC, counting these output spikes and aggregating the results (from four columns) with shift-and-add *(S+A)* operation makes the design technique inherently slow. Additionally, it does not support pipelining of operations.
- **Large area utilization** - Analog-based IFC and the associated digital counter is installed in every column. *S+A* and large capacitor (C_{IFC}) used in proposed IFC configuration takes up addition area as well.
- **High power consumption** - The circuit needs to be ON *i.e.* V_{read} is applied for the complete computational process. Consequently, huge current flows through all the rows for the entire evaluation phase and thus contributing to high power consumption.
- **Accuracy with CMOS-device variations** - Unlike digital computing, accuracy of analog computing largely depends on the strength of CMOS devices *i.e.* spike generation while analog-based DA resolves V_x. Thus, this technique is susceptible to device and voltage variations.

Following are the proposed features that aim to improve the IFC-based adder design. The structure, design specification and functionality of the design features are discussed one-at-a-time. Improved design efficiency solely due to that added feature is qualitatively presented. *4-bit* adder circuit is taken as a case study.

Relative Efficiency of Adders

Metrics \ Design	FBL	IMPLY	MAJ	VSA	IFC
Speed	- - - - -	- - -	- - - - -	- - - -	+
Area	- - -	- - -	- - -	- - -	- - -
Energy	- - - -	- - -	- - -	- - -	- -
Variation Tol.	- - -	- - -	+	+	- -

Table 4.3: Relative overview of dot-product engines.

4.2.1 In-Build Sample-and-Hold

Sample-and-hold *(S+H)* feature is introduced in IFC design to allow pipelining of operations. However, instead of using area inefficient large capacitance to implement *S+H* functionality, a novel circuit is proposed. Figure 4.6 shows the circuit that provides *S+H* feature.

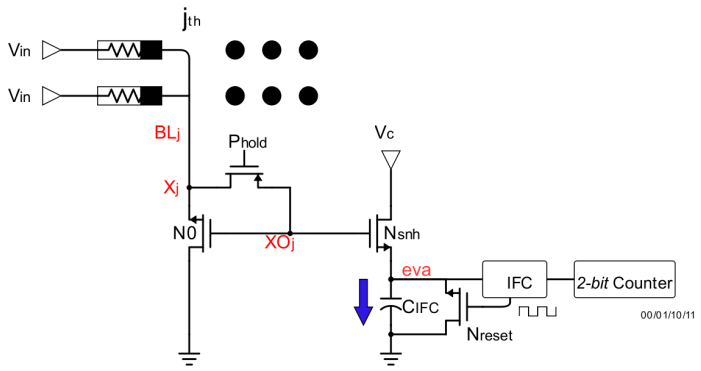


Figure 4.6: In-built sample-and-hold circuit.

The proposed circuit is a current mirror with input current same as the current I_{BLj} received by the original NMOS $N0$ is Figure 4.5b. The current flowing through $N0$ is now mirrored at output NMOS N_{snh} and rest of the IFC design is now connected to N_{snh}. This implies that N_{snh} is now responsible for charging the input node of DA. Evidently, the gate voltage V_{xo} (at node xo) of N_{snh} is not influenced by charging and discharging of node eva throughout the evaluation phase. Given the fact that voltage V_{xo} quantifying the current I_{BLj} does not vary, memristor crossbar can be disconnected from the peripheral evaluating circuit. Hence, *S+H* feature is achieved by disconnecting node xo from x. A strong (2-3μm) PMOS P_{snh} is introduced to disconnect these nodes after a pre-deterministic time.

Key design advantages of the proposed in-built *S+H* technique are highlighted as follows.

1. **Improved latency** - *S+H* allows pipelining *i.e.* new inputs (resistive values) can be programmed to the memristor crossbar after a small pre-deterministic hold time. Assuming evaluation time is t_E and configuration time required for next cycle is t_C then expected improvement in speed is roughly $\frac{t_E+t_C}{t_E}X$.

2. **Improved area utilization** - Huge capacitor (C_{IFC}) is removed which saves significant area. Large NMOS $N0$ ($\sim 10u$m) receiving the crossbar current is kept while N_{snh} is kept extremely small ($\sim 0.1u$m). Hence, additional mirror circuit has negligible area impact.

3. **Improved power-efficiency** - *S+H* reduces the time duration for which large current I_{BLj} is flown through the memristor array. Thereby, reduction in dynamic power of nearly 100X ($\sim \frac{size(N0)}{size(N_{snh})}X$) is expected.

4.2.2 In-Build Bit Weights

Shift-and-add *(S+A)* technique is utilized to aggregate the outputs (# of spikes counted by a digital counter) received from different columns. Since such a sequential approach is a slow process, a novel technique of weighting based on the bit position is proposed to allow parallel processing of different columns. Weighting is achieved by sizing N_{snh} for each column accordingly, to get weight-proportional currents flowing through the *eva* node *i.e.* current responsible for charging C_{IFC}. Figure 4.7 shows the implementation of the novel weighting concept for a *4-bit* adder.

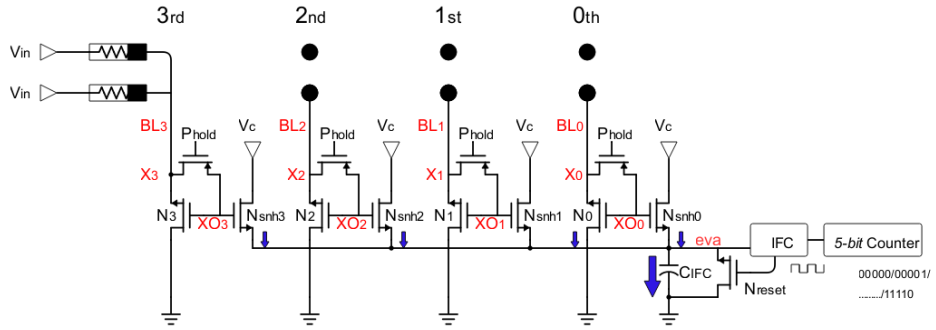


Figure 4.7: In-built bit-wise weighting circuit.

The in-built weighting technique utilizes the premise of IFC design *i.e.* obtain a digital output (# of spikes) linearly proportional to current flowing through a memristor crossbar column. For a *4-bit* adder, four current mirrors are used with sizes N_{snh0} : N_{snh1} : N_{snh2} : N_{snh3} in the ratio 1:2:4:8 while keeping the same input NMOS $N0$ size. The source terminals of N_{snh0}, N_{snh1}, N_{snh2} and N_{snh3} are connected together (as node *eva*), and therefore the summation of these currents is used to charge C_{IFC} (DA's input voltage V_{eva}). Hence, this technique makes the use of *S+A* circuit redundant as the final aggregated digital result is obtained by analog computation before the analog-to-digital conversion. This feature makes the following contributions.

1. **Improved latency** - *S+A* performs the aggregation sequentially. This technique allows parallel processing of different columns. If evaluation time is t_E and aggregation time of *S+A* is t_{SnA}, then expected improvement in speed is roughly $\frac{t_E+t_{SnA}}{t_E}X$.

2. **Improved area utilization** - Huge analog-based IFC is only used once per (here) four columns, which saves significant area. Prior design requires four IFCs which include four large analog-based DAs, four digital counters and a common *S+A* circuit. Instead, only one DA and one counter is required. Although a larger counter is needed, *4-bit* counter instead of a *2-bit* counter. Expected improvement of atleast $\frac{4\times(A_{DA}+A_{2counter})+A_{S+A}}{A_{DA}+A_{4counter}}X$ in area utilization is expected, where area of any component is given by $A_{component}$.

4.2.3 Self-Timing Path (STP)

Analog computation using IFC critically depends on the strength of CMOS devices used. Based on the equations, the digital output (# of spikes) depends on the current flowing through a memristor crossbar column. However, analog components such as current mirrors receiving the current and DA are highly sensitive to CMOS device, voltage and temperature variations. Consequently, the number of spikes vary significantly due to such variations. In this regard, to improve the accuracy of an IFC-based adder circuit design (or any analog-based arithmetic unit for that matter), a novel self-timing technique is proposed that adaptively varies the evaluation time (T_{total}, Equation 4.11) *i.e.* total time available to generate spikes. Figure 4.8 shows the basic circuit implementing the self-timing feature.

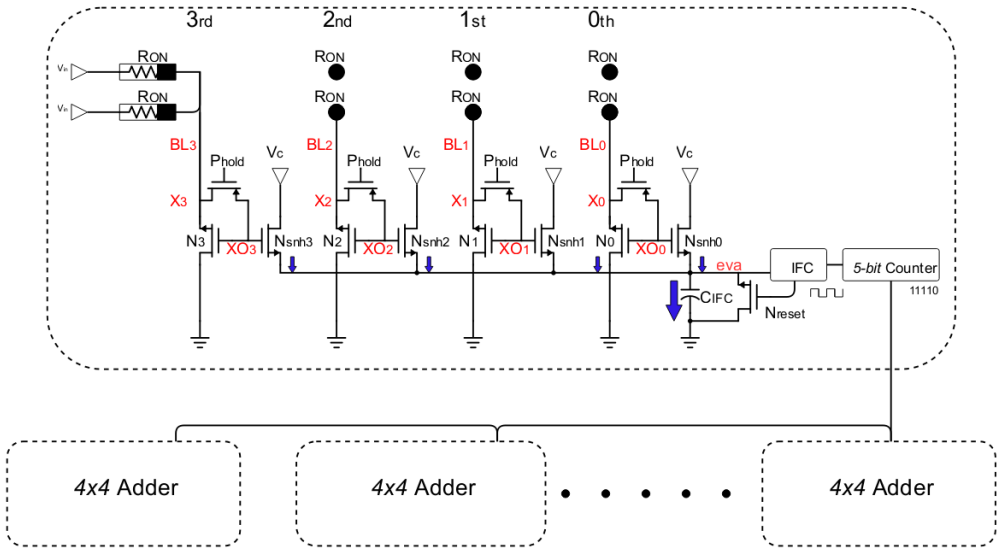


Figure 4.8: STP circuit.

Total time (T_{total}) is key to quantify the digital output. The premise of the proposed STP circuit is to calculate the time an adder takes to produce maximum output, and then use that time interval as T_{total} for the required addition. To implement this technique, a duplicate adder circuit is proposed to count the maximum sum possible *i.e.* a *4-bit* adder with all 1's as input bits (corresponding to all mem-resistance values as *Ron* with maximum sum $S_{max} = 30$) to provide variation-aware T_{total}. The acquired T_{total} is adaptive to global variations of CMOS devices, temperature and fluctuations in the peripheral/core voltage supplies. This feature makes the following contributions.

1. **Improved accuracy** - Analog computing is highly inaccurate. High accuracy against process, temperature and voltage (PVT) variations is achieved using this technique.

2. **Latency** - No impact on latency of the adder circuit.

3. **Energy consumption and area utilization** - Both metrics are almost doubled (2X of what is offered by proposed adder without STP design. Note that area and energy was significantly reduced compared to prior IFC design by using previously discussed design features). However, any computing system would have an array of adders and given that this additional adder is a one-time hardware and energy investment, these penalties are negligible. Nevertheless, it is extremely necessary to provide accurate results against variations.

4.3 Simulation Results and Comparison

This section presents quantitative comparison of proposed *4-bit* adder circuit and prior IFC-based approach described in Section 4.1.2, while taking into account control and other peripheral circuitry utilized by these designs. The comparison of the proposed design is done with two possible IFC-based designs that are optimized for (1) area and (2) speed. The three designs are referred as enhanced-IFC, IFC-Area and IFC-Speed, respectively. This section includes comparison criteria, measuring methodology, variation considerations, followed by comparison results.

4.3.1 Comparison Criteria, Measuring and Variation Methodology

The analyses is performed using Aachen Verilog-A model (Section 2.2). As mentioned earlier, the required regulation of operating voltage during initialization and evaluation phase is still based on CMOS device technology. Therefore, 90nm-TSMC CMOS device models are utilized to simulate these CMOS-based control circuitry. The adder designs are compared against four key design efficiency metrics. The design metrics, corresponding measuring and variation methodology are as follows.

- **Key Design Metrics** :-

  1. **Latency** (L_t) - Minimum time needed to perform *4-bit* adder operation.
  2. **Area** A_t - Area utilized for the entire adder operation.
  3. **Energy** (E_t) - Energy consumption during the entire operation.
  4. **Accuracy** - Computational accuracy of the adder operation against voltage, memristive device and CMOS variations.

- **Setup and Measurement Methodology** :-

  1. All memristors in the crossbar array are initialized to $Roff$. All row voltages *i.e.* $V_{ri} = V_r = 0.7V$. In subsequent (30) cycles, memristors are switched from $Roff \rightarrow Ron$ such that the sum of the two operands linearly increase from 0 (all $Roff$s) to 30 (all Rons). Thereby, all possible cases are simulated.
  2. The adder output *i.e.* sum is obtained by counting the number (#) of spikes obtained through IFC by a digital counter. Hence, accuracy is compared against variations.

3. IFC-Area shares one IFC for the four columns to have an (pseudo) iso-area comparison with enhanced-IFC. This implies that IFC is used in a time-multiplex manner *i.e.* four columns requires four cycles. IFC-Speed however, utilizes IFC in each row to improve speed and requires just one cycle.
4. L_t corresponds to the time taken for the average case *i.e.* sum=15 or average time taken for all the cases (whichever is higher). For IFC-Area(IFC-Speed), shift-and-add circuit optimized for area(speed) is used for comparison.
5. A_t includes area utilized by (90nm) CMOS-based pre and post circuits to perform a complete adder operation. Drivers, registers, counters, shift-and-add circuits are included.
6. E_t is the energy consumption for performing the adder operation. Similar to latency calculation, average case *i.e.* sum=15 or average time taken for all the cases (whichever is higher) is compared.

- **Voltage, Memristive device and CMOS variations** :-

1. **CMOS** - 100 monte carlo (MC) simulations are performed at TT 27C, to investigate accuracy of the adder operation against CMOS device variations.
2. **Voltage** - +/-10% voltage variations is taken into account.
3. **Memristive device** - Three set of nominal and extreme parametric values corresponding to 0%, +30% and -30% memristive device variations are simulated. The parameters are varied in a similar way, as described in Section 3.3.

4.3.2 Comparison and Results

Table 4.4 shows the comparison of IFC-Area, IFC-Speed and enhanced-IFC against previously discussed design metrics. The results substantiate the improvement claims made qualitatively in the previous section. As compared to IFC-Area, enhanced-IFC offers 4X speed, 2.3X area-efficiency and 11X energy-efficiency. As compared to IFC-Speed, enhanced-IFC offers 4.2X area-efficiency and 12.2X energy-efficiency. Besides these improvements, enhanced-IFC also offers pipelining of operation, that IFC-based adders do not offer. Figure 4.9 show simulation results of the sum outputs obtained.

Efficiency comparison of *4-bit* adder designs

Metrics / Design	Latency (ns)	Area (um^2)	Energy (pJ)
IFC-Area	12	17.8	9.78
IFC-Speed	2.25	31.56	10.6
Enhanced-IFC	2.9	7.6	0.87

Table 4.4: Comparison of IFC-Area, IFC-Speed and enhanced-IFC.

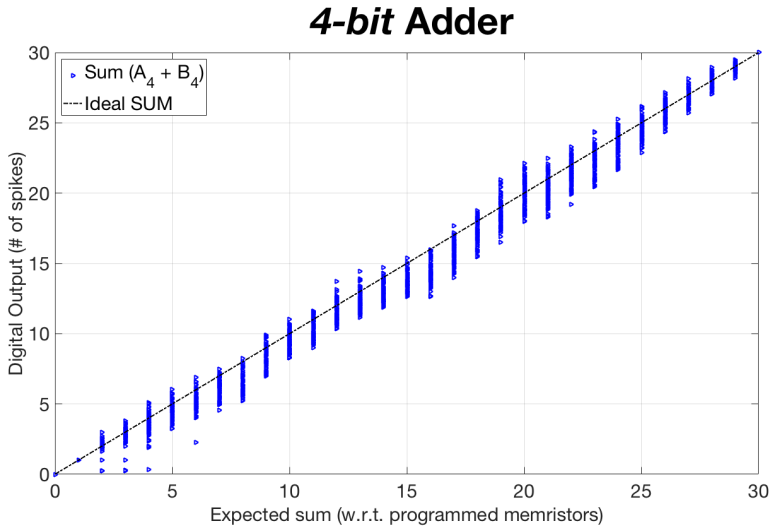


Figure 4.9: *4-bit* adder output.

- **4 operand *4-bit* Adder :-**

Multi-operand addition is achieved by enabling more number of rows at the same time. Rest of the setup, measuring and variation methodology is similar to a 2 operand *4-bit* adder. Figure 4.10 are the simulation results for a 4 operand *4-bit* adder operation, thus showcasing scalability of the proposed design with accurate computation.

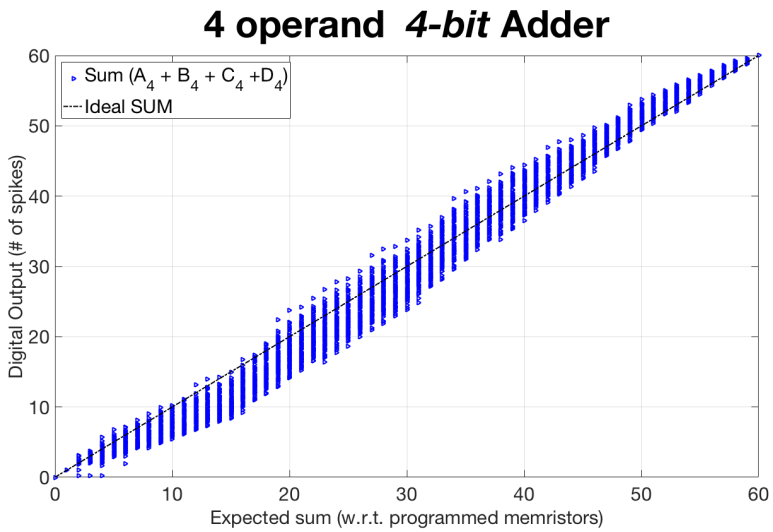


Figure 4.10: 4 operand *4-bit* adder output.

Building Complex Arithmetic Circuit Designs for Data-Intensive Applications

5

A complex arithmetic function can be described as a combination of more than one primitive arithmetic functions. Such complex functions are abundant in today's computing systems. Logarithmic/exponential functions, trigonometric functions, differential/integral functions, dot-products, matrix multiplications are to name a few. However, if algorithm of any such function is investigated and thus segregated into finer functions, multiple-and-accumulate is found to be the key building block.

As discussed in Section 1.1, CMOS-based Von-Neumann architectures are incapable to support data-intensive applications. Image processing, image recognition, hand-writing recognition, DNA-sampling are some of the prime data-intensive applications that require (novel) efficient hardware implementations. Such applications are based on neuromorphic computing i.e. neural computing network to imitate brain-like computation. Functionally, a neural (computing) network involves multi-layered vector matrix multiplications i.e. successive dot product operations, performed on exascale amount of data. Therefore, hardware implementation of a dot product engine is chosen (as a case study) to showcase the promise of memristor-based in-memory computing.

Several dot product engines are proposed, making utmost use of the memristor-based highly dense crossbar memory structure. However, these engines are highly inefficient in terms of speed, area utilization, energy consumption and accuracy of the arithmetic functions performed. This chapter presents a novel circuit design, an extension of the IFC-based adder design proposed previously, that outperforms existing engines.

The organization of this chapter is as follows. Section 5.1 begins with a brief discussion about the hardware implementation of neural networks and memristor-based dot product engines. Thereafter, it illustrates existing memristor-based engines that perform data-intensive dot products, while pointing out their major inefficacies. Section 5.2 introduces a novel dot product circuit design, with detailed description of each of the key features incorporated. Qualitative improvements corresponding to each of the introduced feature are also discussed. Section 5.3 provides a detailed comparison of key designs efficiency metrics with existing dot product engines. Section 5.4 explores commonly employed arithmetic and functional units to showcase the versatility of the proposed circuit design. Thereby, design prepositions of a 4x4 multiplier and a 4-bit comparator are presented.

5.1 Overview of Existing Dot Product Engines

This section briefly introduces the concept of convolutional and deep neural networks and puts forth the motivation behind the choice of such applications. Thereafter, this section illustrates existing dot product engines (or multiply-accumulate circuit designs), which is the fundamental computational unit of such applications. The main idea of these circuit design solutions are described briefly, while discussing their key features, benefits and limitations.

- **Convolution and Deep Neural Networks in Hardware**

Convolution [85] and deep neural networks [86] (CNN/DNNs) are state-of-the-art machine-learning algorithms. From a functional point of view, CNN/DNNs typically consist of multiple layers of vector matrix multiplications performed by a series of dot-product operations. The dot product is computed at the neuron while the result is broadcasted via synaptic weights. From a hardware perspective, the inherent structure of these networks offer an opportunity to design highly specialized and efficient circuit design solutions. Considering that solutions based on conventional Von-Neumann based accelerators [34, 87, 88] are impeded by memory access bottleneck (see Section 1.1), the hardware implementation of such applications has been the prime focus of research.

The subsequently discussed existing memristor-based dot-product engines map the memory and required computation directly to on-chip storage *i.e.* in-memory computing, are designed as an attempt to solve the above issue. However, these engines suffer from high area utilization and energy consumption while also lack the accuracy needed to perform such data-intensive dot products.

NOTE - *These solutions also lay emphasis on implementing multi-layered convolutions and thereby have proposed architectural optimization to perform a full neural network operation. Since this work only focuses on optimizing the dot-product engine, overall architecture can be assumed to be the same. Nevertheless, optimized architecture to be build around the proposed dot-product is considered as a future research work.*

- **Dot product engine in a crossbar array**

The existing implementations of a dot product engine generally follow the same approach *i.e.* multiply-accumulate operation performed via analog computation. The implementations of analog-to-digital converter are however different and are illustrated in detailed subsequently. The shared working principle of dot product engines is shown in Figure 5.1.

Dot product is performed between synaptic weights and an input vector [11]. Training a neural network provides fixed synaptic weights which are stored as G_{ij}, where G_{ij} (G_{on} or G_{off}) is the mem-conductance stored in memristors placed at the cross-section of ith row and jth column of a crossbar array. Each neuron has n elements stored in n rows in a $n \times m$ crossbar array, with each element of size w-bits *i.e.* m/w different neurons are stored, with one neuron occupying n rows and w columns. (For simplicity, as shown in Figure 5.1, $w = 4$ implying that each neuron is stored in four consecutive

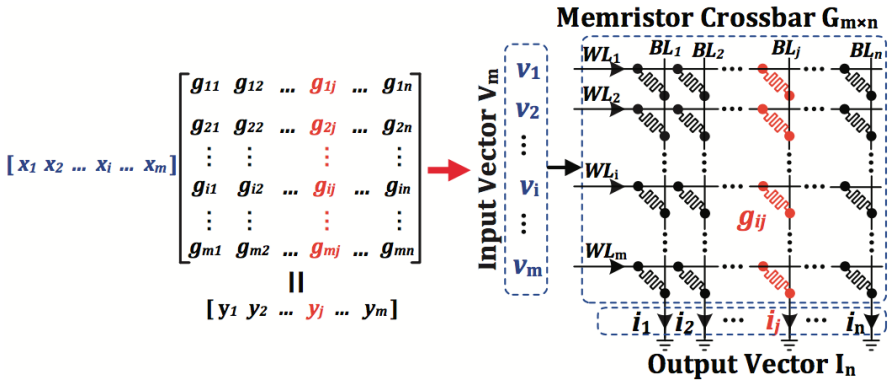


Figure 5.1: Dot product engine [10].

columns.) Memristor M_{ij} has non-linear I-V characteristics [11], which connects the horizontal select word line (WL_i) and vertical bit-line (BL_j).

Input vector ($=n$ elements, each element size of p-bits) is applied as an array of voltages at each of the (n) rows as V_{ik}, where kth significant bit of voltage representation is applied to ith row. Therefore, at any instant for each column j, partial dot product operation with operands G_{1j}, G_{2j}, G_{3j} ... G_{nj} and V_{1k}, V_{2k}, V_{3k} and V_{nk} is performed, given by the following expression.

$$\sum_{i=1}^{n} V_{ik} \cdot G_{ij} = V_{1k} \cdot G_{1j} + + V_{2k} \cdot G_{2j} + V_{3k} \cdot G_{3j} .... + V_{nk} \cdot G_{nj}$$

The input voltage is applied to all (m) columns *i.e.* V_{ik} is applied to all memristors M_{i1}, M_{i2} .... M_{im} placed in the ith row during the kth cycle. (For simplicity, each voltage vector is kept of size $1-bit$ *i.e.* 0 or V_{in}, therefore utilizing just one cycle. Hence, suffix k is dropped ($V_{ik} = V_i$) as to represent $1-bit$ voltage.) Since a neuron is stored as a set of mem-conductance values in four consecutive columns (or bit-lines), the current flowing through all such columns represent the output corresponding to that neuron for a given (applied) input vector. This allows parallel dot product evaluation of $m/4$ different neurons in $p(=1)$ step(s), which implies that $n \times m$ (bit-wise) vector matrix multiplication is performed in $1(p)$ step(s). The current $\sum_{i=1}^{n} V_i \cdot G_{ij}$ flowing through a column j is quantified using an analog-to-digital converter (ADC) and the aggregation of digital results obtained associated with a neuron (from set of four columns for each neuron) is done with help of shift-and-add ($S + A$) circuits. The existing circuit design implementations using different ADCs are now presented.

5.1.1 ISAAC using SAR ADC

Shafiee *et al.* proposed in-situ analog arithmetic in crossbar (ISAAC) [11] as a CNN accelerator. ISAAC deploys modified structure of the dot product engine discussed previously, as shown in Figure 5.2.

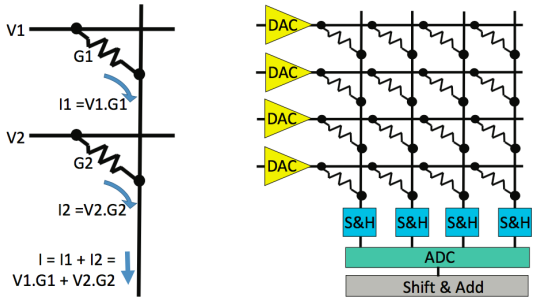


Figure 5.2: Dot product engine using SAR-ADC [11].

The voltage (input vector) is applied to every row (WL_i) using a $1-bit$ digital-to-analog converter (DAC). Therefore, the voltage V_i applied to ith row is either 0 or V_{in} based on digital inputs 0 or 1, respectively. However, prior to analog-to-digital conversion of current flowing through BL_j *i.e.* $\sum_{i=1}^{n} V_i \cdot G_{ij}$, sample-and-hold circuit $(S+H)$ is employed to extract parallelism and hence support pipelining of operations. $S+H$ is implemented using a large capacitance with a switch (generally NMOS) providing isolation between computation performed by peripheral circuits and initialization of inputs in the memristor crossbar array for the next cycle. State-of-the-art 8-bit SAR-ADC (optimized for area) [89] is used to implement the conversion unit. Figure 5.3 shows the block diagram of a well-known SAR-ADC. $S+A$ circuit is used to aggregate the digital values obtained from columns representing a neuron (here, a set of four columns) which utilizes the classic multiplicative algorithm of adding next-higher bit product to left-shifted version of the prior partial product.

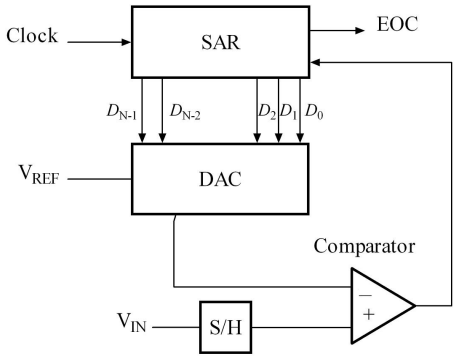
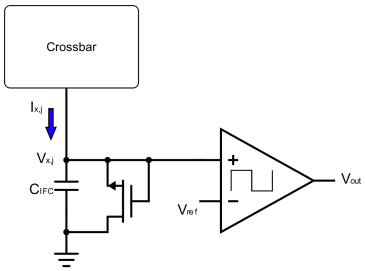


Figure 5.3: SAR-ADC block diagram.

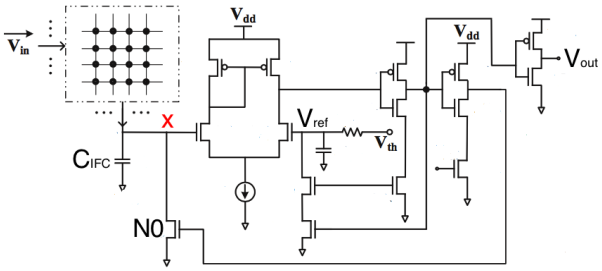
The paper, however points out a major disadvantage that SAR-ADC alone contributes to 58% power and 31% area of the entire chip. The next approach attempts to solve this inefficacy.

5.1.2 Integrate and Fire Circuit

Similar to the circuit design to perform addition as illustrated in section 4.1, Liu implemented dot product engine via analog computing [12]. The structure of IFC-based ADC is shown in Figure 5.4.



(a) IFC-based dot product engine in memristor crossbar.



(b) Circuit design of IFC [12].

Figure 5.4: Dot product engine using IFC.

There are certain similarities and dis-similarities between circuit design implementation of based on SAR-ADC and IFC. The application of input vector and storage of synaptic weights is same as ISAAC. However, instead of directly converting the total current $\sum_{i=1}^{n} V_i \cdot G_{ij}$ into digital outputs by SAR-ADC, IFC converts (quantifies) the current as number of pulses (or # of spikes). Thus, a digitalized output ensures good noise immunity and high energy efficiency in data communication. Similar to ISAAC, $S + A$ circuits are used to aggregate the digital values obtained from four consecutive columns representing a neuron. The clear advantage of dot product engine based on IFC over SAR-ADC is low area utilization and power consumption.

However, as claimed by the authors in [12] and shown in Figure 5.5, the output *i.e.* # of spikes losses its linearity at around 20-30, even after relaxing the total evaluation time to quantify the current. In other words, # of spikes does not increase linearly (and thus saturates) after reaching 30. Therefore, maximum rows supported by IFC-based dot product is around 30. Additionally, this technique can not support parallelism through pipelining since there is no $S + H$ circuit. The next approach attempts to solve the non-linearity problem.

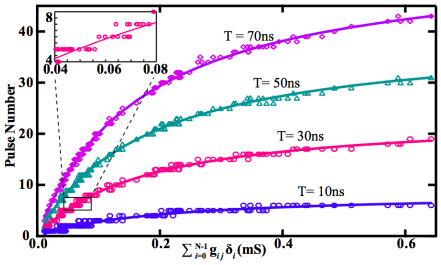
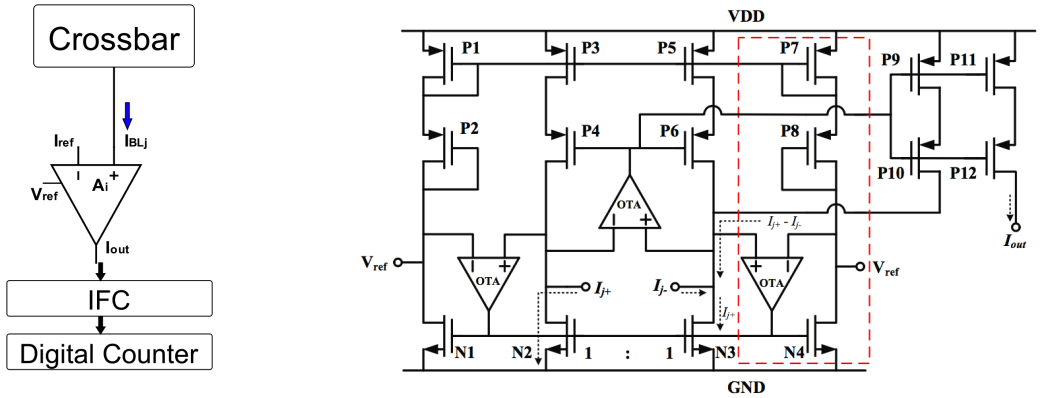


Figure 5.5: Non-linearity faced using IFC-based dot product engine [12].

5.1.3 Current Sense Amplifier Circuit

Liu *et al.* proposed current sense amplifier (CSA) [10] based ADC to improve the linearity between current flowing through BL_j, given by *i.e.* $\sum_{i=1}^{n} V_i \cdot G_{ij}$ and output spiking number. The prime feature of this circuit design is to fix the voltage of BL_j and thus allow a linear increase in current with increase in $\sum_{i=1}^{n} V_i \cdot G_{ij}$. The structure of the dot product engine with detailed circuit design of CSA is shown in Figure 5.6a.



(a) Modified IFC-based dot product engine.

(b) Detailed design of current sense amplifier (CSA) [10].

Figure 5.6: Dot product engine using CSA for high linearity.

The structure is similar to the one utilized in IFC-based dot product engine, but with one major modification. Instead of directly connecting BL_i to charge and discharge a capacitor (also, positive input voltage V_x of DA in IFC), CSA is bridged between the two. CSA clamps a constant voltage at BL_j, ensuring a fixed voltage across memristor devices. The working of the circuit is explained as follows.

The labels given to MOSFETs starting with P(N) are PMOS(NMOS) devices and current through BL_j is I_{BLj}. $P1$, $P2$ and $N1$ provide a reference voltage V_{ref}, which fixes the voltage (drain-source and gate-source voltage) of $N2$, $N3$ and $N4$ as they are connected by a chain of operational trans-conductance amplifiers (OTAs). Thus, it enables current mirror functionality with paths $P3-P4-N2$, $P5-P6-N3$ and $P7-P8-N4$ following the current flowing through $P1-P2-N1$ *i.e.* I_{BLj}. The output current follows the input current I_{BLj} provided by a cascode structure of PMOS devices $P9-P12$ to improve the accuracy of the current mirror.

Apart from the high linearity provided by CSA-based ADC, it also allows the use of $1R$ bit-cell instead of large $1T1R$ bit-cell used in previously illustrated circuit designs. This is due to the fact that since the voltage of BL_j node is fixed, there is no sneak-path issue, thus making the use of a pass transistor futile. Therefore, densely packed memristor crossbars can be built with $1R$ bit-cell of feature size $4F^2$. However, these bulky and power hungry CSA structures are installed in every column, apart from IFC and $S+A$ circuits. Additionally, this technique can not support parallelism through pipelining since there is no $S+H$ circuit. Therefore, low speed, high area utilization and power consumption weakens the prospect of this circuit technique.

5.2 Proposed Dot Product Engine

The proposed novel circuit design attempts to solve the inefficiencies suffered by dot product engines discussed in the previous section. Following are their key drawbacks (also summarized in Table 5.1), which form the prime targets of the proposed design solution illustrated subsequently.

- **High latency** - Following the generation of output spikes by IFC, counting these output spikes and aggregating the results (from $w(= 4)$ columns) with shift-and-add *(S+A)* operation makes the design technique inherently slow. Additionally, it does not support pipelining of operations.

- **Large area utilization** - SAR-ADC utilize extremely large area. IFC and additional CSA in the modified-IFC technique, plus the associated digital counter is installed in every column. *S+A* and large capacitor (C_{IFC}) used in IFC configuration takes up addition area as well. IFC-based techniques are relatively more area-efficient than SAR-ADC, but nevertheless utilize a considerable area per column.

- **High energy consumption** - SAR-ADC alone consumes 58% power of the entire chip design and more than 74% area utilized by the entire dot product engine. Relatively power-efficient IFC-based dot product engines still needs to be ON *i.e.* V_{read} is applied for the complete computational process (no $S+H$ circuits). Therefore, huge current flows through all the columns for the entire evaluation phase. Even more so for the power hungry CSA employed by modified-IFC technique.

- **Reduced read endurance** - An array of voltage values is applied to all the rows of memristor-based crossbar as an input vector. The voltage of operation is typically $1.2V$ with an average voltage of $\sim 0.9V$ across a memristor device. A voltage of this magnitude reduces the read endurance of memristor devices considerably. Especially in case of a neural network application, given that once mem-resistance (synaptic weight) is fixed it is not changed *i.e.* write operations are very infrequent, endurance related to read operation is crucial.

- **Non-linearity** - SAR-ADC is linear but highly area and power inefficient (mentioned above). IFC is non-linear, and although modified-IFC is linear, it is other major disadvantages mentioned above.

- **Accuracy with CMOS-device and voltage variations** - Digital output acquired using analog computing largely depends on strength of the CMOS devices *i.e.* # of spikes depend on the speed with which V_x is charged to V_{ref} and analog-based DA resolves V_x.

Following are the proposed **(additional)** features that aim to improve IFC-based dot-product engine. *(The proposed dot product engine is inclusive of the features discussed in Section 4.2 i.e. proposed features for the adder circuit design. Since those features are elaborated previously, they are not discussed here.)* The structure,

design specification and functionality of the design features are discussed one-at-a-time. Improved design efficiency solely due to a particular feature is qualitatively presented. Dot-product of one neuron of size 64×4 is taken as a case study, with a voltage array (input vector) applied across all 64 rows to present the working of the proposed novel circuit design.

Relative Efficiency of ADCs

Metrics \ Design	SAR	IFC	CSA
Speed	+	-	-
Area	- - - -	-	- -
Energy	- - - -	-	- -
Variation Tol.	-	- - -	- -
Accuracy	+	- - - -	+

Table 5.1: Relative overview of dot-product engines.

5.2.1 High Gain Differential Amplifier

An advanced version of proposed adder circuit design (Figure 4.7) offers a highly area- and power-efficient solution to fix BL_j voltage. A high gain differential amplifier (DA) is placed between BL_j and node x, as shown in Figure 5.7. A detailed illustration regarding the use of high-gain DA is described next.

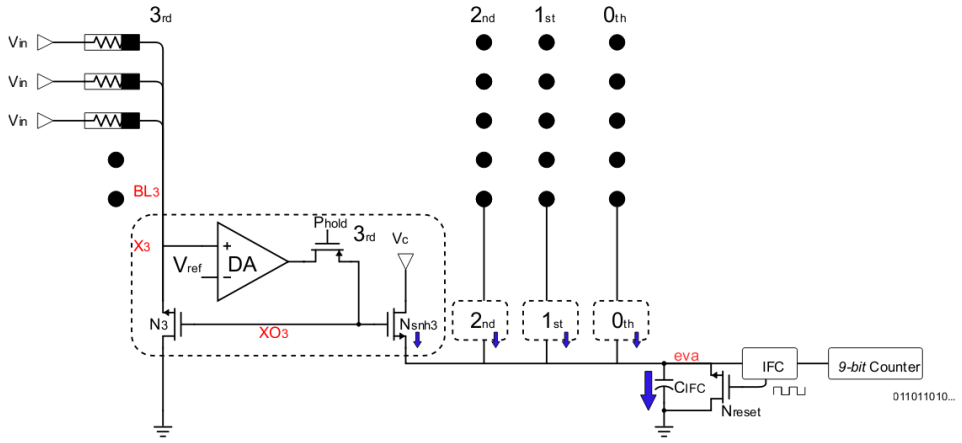


Figure 5.7: Proposed high-gain DA based dot product engine.

The premise of the proposed technique is that the two inputs of a high-gain DA connected through in a negative feedback loop tend to converge at equal voltages. Here, since a fixed V_{ref} is provided as a negative terminal input of DA, BL_j tends to reach ($\sim V_{ref}$), while accordingly tuning xo to allow appropriate current through $N0$ (I_{BLj}). To make sure that V_{BLj} reaches $\sim V_{ref}$ at all possible scenarios, a high-gain DA is implemented using $350nm$ devices to provide a large range of V_{xo} for a very small differential input *i.e.* ($V_{BLj} - V_{ref}$). With appropriate sizing and biasing voltage, a mere $+/-15mV$ shift in V_{BLj} is achieved covering almost the entire range of possible cases supporting 64 rows. The extreme cases considered are all memristors with mem-conductance $Goff$ or Gon in a column. $S+H$ feature is implemented by just using a PMOS switch (P_{hold}) that isolates xo and x *i.e.* isolates BL_j or the memristor crossbar structure from the computing structure (IFC). Similar to the proposed adder circuit, parallelism (pipelining of operations) is provided by P_{hold}, which disconnects xo from BL_j after a small fixed time interval, given that V_{xo} does not change (V_{gate} of N_{snh} is un-affected) while IFC generate spikes.

Associated advantages of this feature are highlighted as follows.

1. **Improved Linearity** - Once V_{BLj} is virtually fixed at V_{ref}, I_{BLj} follows linearly with $\sum_{i=1}^{n} V_i \cdot G_{ij}$. This results in a linearized charging time of the capacitance C_{IFC} by the final evaluation current I_{eva}. (hence, linearized # of spikes).

2. **Improved latency** - P_{hold} enables pipelining

3. **Improves Power** - For a single operation, since P_{hold} provides isolation between the evaluation circuit and the memristor crossbar, the voltage supply can be removed to save power.

4. **Improved area** - This technique enables the use of $1R$ bit-cell (feature size $4F^2$) instead of $1T1R$ ($67F^2$), resulting a $\sim$ 17X increase in memristor crossbar storage density.

This technique is the key to enable implementation/working of the subsequent features that makes the proposed design highly power-efficient. Furthermore, read endurance and power/energy consumption is drastically improved along with linearized output spike number.

5.2.2 Modified In-Build Bit Weights

The aggregation of outputs (# of spikes counted by a digital counter) received from all the columns associated with a neural is usually done using a shift-and-add *(S+A)* circuit. Since such a sequential addition is a slow process, two novel techniques of weighting based on bit position are proposed to allow parallel processing of different columns. Thus, making the use of $S + A$ circuitry redundant. Weighting achieved by sizing N_{snh} accordingly, to get weight-proportional currents at the output is already illustrated in Section 4.2 (Figure 4.7). Weighting can also be achieved by having different reference voltages (V_{ref}) connected to the negative input to the high-gain DA. Figure 5.8 shows the implementation and working of the proposed weighting technique.

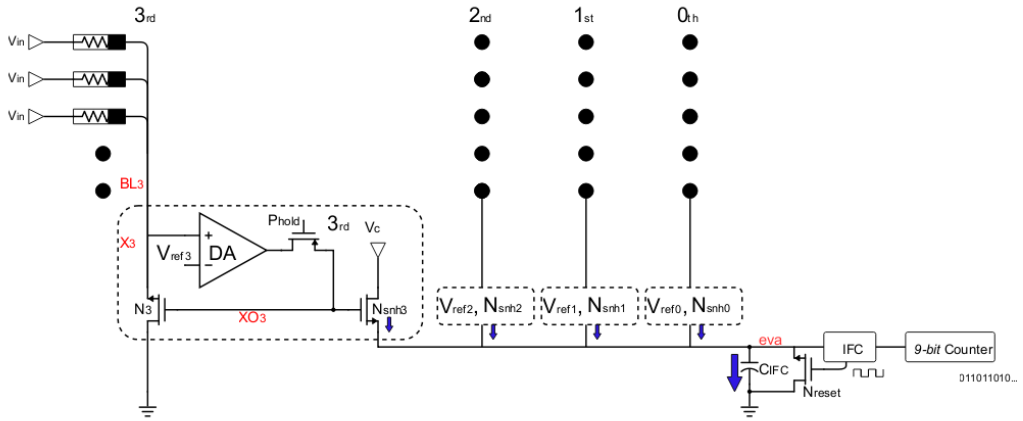


Figure 5.8: Proposed in-built weighting technique using different voltage references. Output is combined for a neuron represented in 64×4 memristor crossbar array.

The premise of this technique is to maintain appropriate and virtually fixed voltage across the memristors in a particular column. In this manner, corresponding to a column's weight, different voltages across memristors in different columns can be maintained accordingly. Let voltage applied to each row i be V_i, such that V_i can assume

values 0V or V_{in} for an input 0 or 1, respectively. Also, let the consecutive columns with bit positions(weighting factor) and reference voltages be $0th(1)$, $1st(2)$, $2nd(4)$ and V_{ref1}, V_{ref2}, V_{ref4}, respectively. Then for input voltage, say $V_{in} = 0.7V$, V_{ref1}, V_{ref2}, V_{ref4} is kept at $0.65V$, $0.6V$ and $0.5V$, respectively. The fact that high-gain DA virtually fixes the BL voltage (V_{BLj}) at V_{ref} (negative input voltage of DA), voltage across the memristors in a column is virtually maintained at $50mV$, $100mV$ and $200mV$, respectively. This allows a current multiplication corresponding to the weights represented by each column position, similar to the outcome achieved by sizing N_{snh} previously illustrated.

Since the two weighting techniques are exclusively implemented, they can be integrated together to combine more columns as well. An intuitive combination for integrating these techniques is illustrated with $\{S_{N0} : V_{in}\} = \{10um : 0.7V\}$ for aggregating a neuron with four column representation. One such combination with different pairing of $\{S_{Nsnhi} : V_{refi}\}$ is $\{0.1um : 0.65V\}$, $\{0.1um : 0.60V\}$, $\{0.2um : 0.60V\}$ and $\{0.4um : 0.60V\}$ for $0th$, $1st$, $2nd$ and $3rd$ bit position, respectively.

Associated advantages of this feature are highlighted as follows.

1. **Improved latency** - Apart from the pipelining offered by in-built $S + H$ circuit, this technique allows parallel aggregation of digital outputs obtained from different columns.

2. **Improved read endurance** - Since a very small virtually fixed voltage is maintained across the memristors, read endurance improves drastically.

3. **Improved power** - Since a very small virtually fixed voltage is maintained across the memristors, an extremely low current flows through all the columns of a memristor crossbar, not to mention that $S + H$ circuit reduces the duration of applied (input row) voltages.

4. **Area overhead** - Additional area include 2 MOSFETs per reference voltage, therefore n different reference voltages will require $2n$ MOSFETs. Hence, a negligible area overhead to implement this technique.

5.2.3 Improved IFC (PMOS DA)

The differential amplifier (DA) used in the original IFC design is NMOS-based *i.e.* input voltages are applied at gate terminal of a NMOS device. To further improve the linearity of the proposed ADC, PMOS-based DA is used instead. Figure 5.10 shows the implementation of this proposed alteration. The associated design changes, working and the claim of improving the linearity is described next.

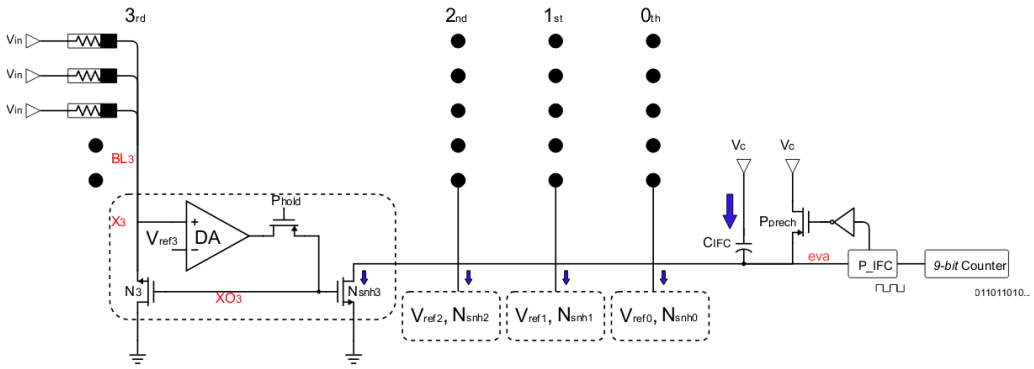


Figure 5.9: Proposed PMOS-DA based IFC design.

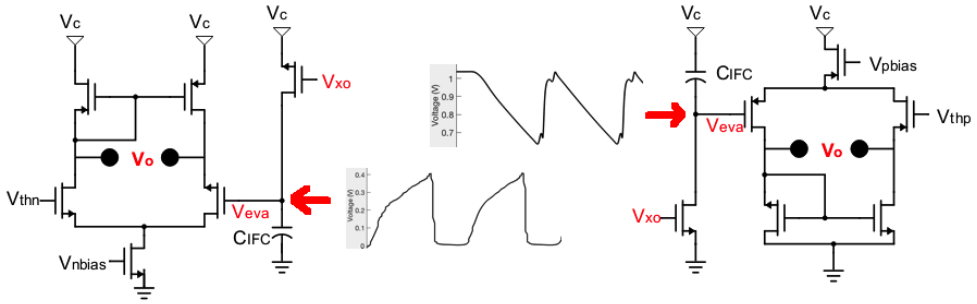
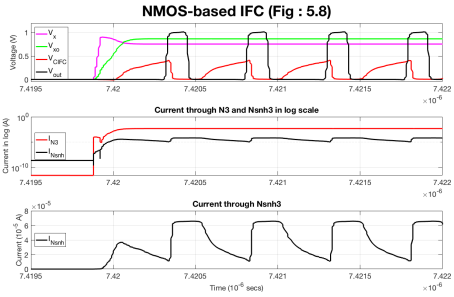


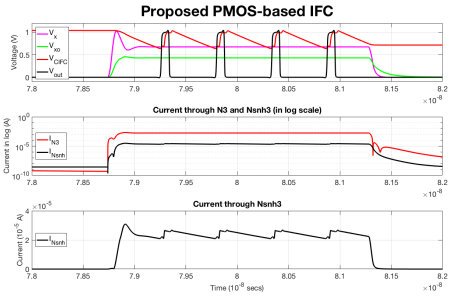
Figure 5.10: NMOS (left) vs PMOS-based DA utilized in IFC.

The modification proposed involves discharging capacitor C_{IFC} from, say $VDD \rightarrow Vthp$ instead of charging it from, say $0 \rightarrow Vthn$, to trigger an output spike. Here, VDD is supply voltage of IFC, $Vthp(Vthn)$ is threshold or trigger voltage for PMOS(NMOS)-based DA. Due to this modification, resetting the node *eva* involves pre-charging it to VDD instead of pre-discharging it to GND. Not to mention, the reset signal is inverted for this altered arrangement.

The intention is to provide the current charging or discharging C_{IFC} to be nearly constant throughout the evaluation phase. Figure 5.11a and Figure 5.11b illustrates the working of previous and the proposed redesign, respectively. While charging C_{IFC} (placed between N_{snh} and GND) through NMOS (N_{snh}), change occurs in both V_{DS}



(a) NMOS-based DA (Figure 5.8).



(b) Proposed PMOS-based DA.

Figure 5.11: Current comparison of NMOS-based and proposed PMOS-based DA for improved accuracy.

and V_{GS} of N_{snh}. This implies that current provided by N_{snh} fluctuates considerably as its V_{GS} switches between V_{xo} and $V_{xo} - Vthn$. However, if C_{IFC} is discharged (placed between VDD supply and N_{snh}) through N_{snh}, change during the evaluation only occurs in V_{DS} (V_{GS} remains constant at V_{xo}). Since N_{snh} is always in strong saturation mode, a V_{DS} change of VDD to $VDD - Vthp$ does not change the discharging current considerably. The I-V curves at two extreme ends of the evaluation phase are inscribed in Figure **??** with their respective design choices. This ensures a more linear increase in number of output spikes with increase in $\sum_{i=1}^{n} V_i \cdot G_{ij}$ for a given column j with n number of rows. Another point worth mentioning here is that keeping $Vthp = V_{ref}$ ensures a highly accurate current mirror.

5.3 Simulation Results and Comparison

This section presents quantitative comparison of proposed dot product engine and prior engines described in Section 5.1, while taking into account control and other peripheral circuitry utilized by these designs. The comparison of enhanced-IFC is done with IFC-Area (optimized for area), SAR-ADC and CSA which are referred to as the same, respectively. This section includes comparison criteria, measuring methodology, variation considerations, followed by comparison results.

5.3.1 Comparison Criteria, Measuring and Variation Methodology

The analyses is performed using Aachen Verilog-A model (Section 2.2). As mentioned earlier, the required regulation of operating voltage during initialization and evaluation phase is still based on CMOS device technology. Therefore, 90nm-TSMC CMOS device models are utilized to simulate these CMOS-based control circuitry. The dot product operation is performed between 2 operands, each of length 64-elements but each element of width *1-bit* and *4-bits*, respectively *i.e.* $(64_1 \cdot 64_4)$. Hence, corresponding engines performing $64_1 \cdot 64_4$ are compared against four key design efficiency metrics. The design metrics, corresponding measuring and variation methodology are as follows.

- **Key Design Metrics** :-
  1. **Latency** (L_t) - Minimum time needed to perform dot product operation.
  2. **Area** A_t - Area utilized for the entire dot product operation.
  3. **Energy** (E_t) - Energy consumption during the entire operation.
  4. **Accuracy** - Computational accuracy of the operation against voltage, memristive device and CMOS variations.
  5. **Range** (# of rows) - Number of rows supported (simultaneously) for a given design.
  6. **Endurance** $(V_{tb(avg)})$ - Average voltage across memristors in the crossbar array.
- **Setup and Measurement Methodology** :-
  1. All memristors in the crossbar array are initialized to $Roff$. All voltage inputs *i.e.* $V_{ri} = V_r = 0.7V$. In subsequent (64) cycles, memristors are switched from $Roff \rightarrow Ron$ such that the dot product linearly increase from 0 (all $Roff$s) to 64 (all Rons). Thereby, all possible cases are simulated.
  2. The dot product output *i.e.* sum is obtained by counting the number (#) of spikes obtained through IFC by a digital counter. Hence, accuracy is compared against variations.
  3. IFC-Area(SAR-ADC, CSA) shares one IFC(ADC, CSA) for the four columns to have an (pseudo) iso-area comparison with enhanced-IFC. This implies that IFC(ADC, CSA) is used in a time-multiplex manner *i.e.* four columns requires four cycles.

4. L_t corresponds to the time taken for the average case *i.e.* product=32 or average time taken for all the cases (whichever is higher). For IFC-Area, SAR-ADC and CSA, shift-and-add circuit optimized for area is used for comparison.
5. A_t includes area utilized by (90nm) CMOS-based pre and post circuits to perform a complete dot product operation. Drivers, registers, counters, shift-and-add circuits are included.
6. E_t is the energy consumption for performing the adder operation. Similar to latency calculation, average case *i.e.* product=32 or average time taken for all the cases (whichever is higher) is compared.

- **Voltage, Memristive device and CMOS variations** :-

1. **CMOS** - 100 monte carlo (MC) simulations are performed at TT 27C, to investigate accuracy of the adder operation against CMOS device variations.
2. **Voltage** - +/-10% voltage variations are taken into account.
3. **Memristive device** - Three set of nominal and extreme parametric values corresponding to 0%, +30% and -30% memristive device variations are simulated. The parameters are varied in a similar way, as described in Section 3.3.

5.3.2 Comparison and Results

Table 5.2 shows the comparison of IFC-Area, SAR-ADC, CSA and enhanced-IFC against previously discussed design metrics. The results substantiate the improvement claims made qualitatively in the previous section.

As compared to IFC-Area, enhanced-IFC offers 2X speed, > 3X range, 3.6X area-efficiency, 54X energy-efficiency and an improved endurance. As compared to SAR-ADC, enhanced-IFC offers 200X area-efficiency, 270X energy-efficiency with reduced speed of nearly 0.4X but with similar range and accuracy. As compared to CSA, enhanced-IFC offers 2X speed, 6X area-efficiency, 108X energy-efficiency and an improved endurance with similar range and accuracy. Besides these improvements, enhanced-IFC also offers pipelining of operation, that IFC-Area and CSA do not offer which is a key feature to support neural networks algorithms.

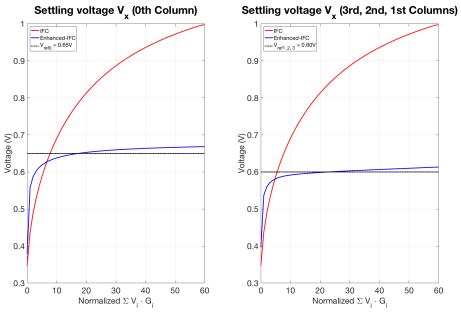
Efficiency comparison of $64_1 \cdot 64_4$ dot product engines

Metrics / Design	Latency (ns)	Area (um^2)	Energy (nJ)	Range (#)	Endurance ($V_{tb(avg)}$)
IFC-Area	400	102.6	6244	~ 20	$\sim 1V$
SAR-ADC	80	5500	30233	> 64	$\sim 1V$
CSA	400	171	12640	> 64	$\sim 1.1V$
Enhanced-IFC	200	28	116	> 64	$\sim 0.2V$

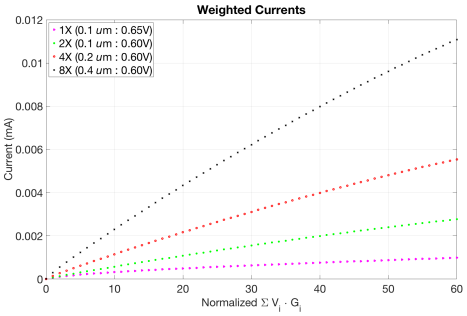
Table 5.2: Comparison of IFC-Area, SAR-ADC, CSA and enhanced-IFC.

Figure 5.12 and 5.13 shows simulation results of the dot product outputs obtained with the proposed dot product engine. Figure 5.12a shows node x being held at V_{ref},

as supported by circuits presented in Section 5.2.1 and 5.2.2. Figure 5.12b shows the weighted current flowing through different columns *i.e.* 1X, 2X, 4X and 8X current flowing through column $0th$, $1st$, $2nd$ and $3rd$, as supported by weighting technique presented in Section 5.2.2. Figure 5.13a shows digital outputs obtained for a single column with 64 rows. Figure 5.13b shows digital outputs obtained for four columns combined using the proposed circuit design in Figure 5.10.

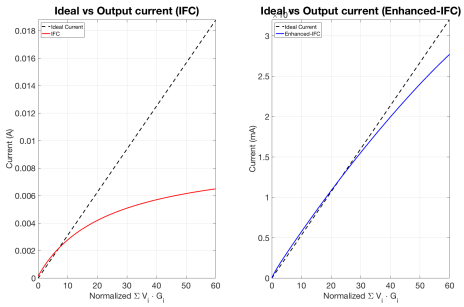


(a) Settling voltage (V_x) comparison.

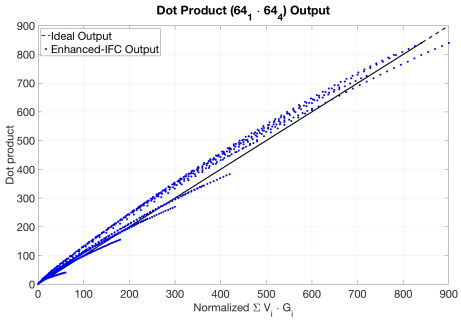


(b) Weighted currents.

Figure 5.12: IFC vs enhanced-IFC comparison.



(a) Current comparison ($64_1 \cdot 64_1$) with IFC-Area.



(b) Digital output *i.e.* # of spikes ($64_4 \cdot 64_1$).

Figure 5.13: Enhanced-IFC results.

5.4 Advanced Designs for Arithmetic and Logic Functions

This section includes design preposition of two of the most common operations utilized in computing systems *i.e.* *4×4* multiplier and *4-bit* comparator. The circuit designs are inclusive of all the fixtures previously discussed.

5.4.1 *4×4* Multiplier

Circuit design of a *4×4* multiplier is an extension of four operand *4-bit* parallel adder design presented in Section 4.3. The multiplier design performs in-built shift-and-add mechanism within the memristor crossbar structure, following the standard multiplication algorithm used in modern computing units. Figure 5.14 shows the proposed circuit design.

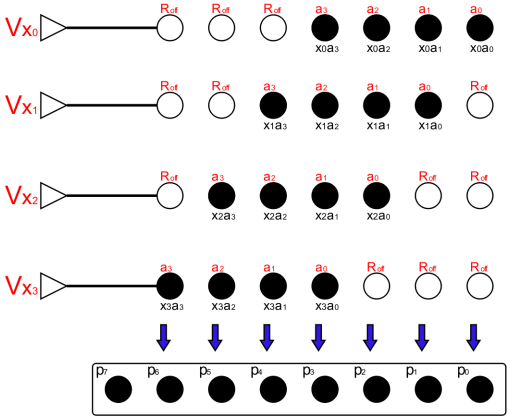


Figure 5.14: Proposed *4×4* multiplier circuit design.

Here, $x_3x_2x_1x_0(X_4)$ and $a_3a_2a_1a_0(A_4)$ are *4-bit* multiplier and multiplicand, respectively, while the *8-bit* product A_4X_4 is given by $p_7p_6p_5p_4p_3p_2p_1p_0(P_8)$. Array X_4 is represented as bit-wise voltage $V_i(0 \leq i \leq 3)$, with voltage values $GND(0)$ or $V_{in}(1)$ applied to row i. Array A_4 is represented as bit-wise mem-conductance $G_{ij}(0 \leq j \leq 6)$, stored in the memristor crossbar with conductance values $Goff(0)$ or $Gon(1)$ in the manner as shown in Figure 5.14. The multiplicand A_4 is placed according to its weight/bit-position in rows 0-3, while the unused positions in the array are initialized to $0(Goff)$.

Due to the introduction of in-built weighting through positional storing of multiplicand bits, external shift-and-add mechanism is essentially not required. Current $I_j = \sum_{i=0}^{3} V_i \cdot G_{ij}$ represents weighted analog accumulation of bit-wise AND function elements $x_i \cdot a_j$ for a given column j. As discussed earlier, in-built weighting can be achieved independently by progressive sizing and/or variable voltage referencing. For this case study, the optimal configuration of $\{S_{Nsnhi} : V_{refi}\}$ for a given $\{S_{N0} : V_{in}\} = \{10um : 0.7V\}$ is found to be $\{0.1um : 0.65V\}$, $\{0.2um : 0.65V\}$, $\{0.4um : 0.65V\}$, $\{0.8um : 0.65V\}$, $\{0.8um : 0.60V\}$, $\{1.6um : 0.60V\}$ and $\{3.2um : 0.60V\}$ at bit position $0th$, $1st$, $2nd$, $3rd$, $4th$, $5th$ and $6th$, respectively.

5.4.2 *4-bit* Comparator

Circuit design of a *4-bit* comparator is an extension of a *4-bit* parallel adder design presented in Section 4.3. A *2-input* comparator provides which is greater of the two inputs or whether both are equal. Working principle of the proposed comparator design is to compare voltage equivalent of the accumulated current corresponding to individual operands. Figure 5.15 shows the proposed circuit design.

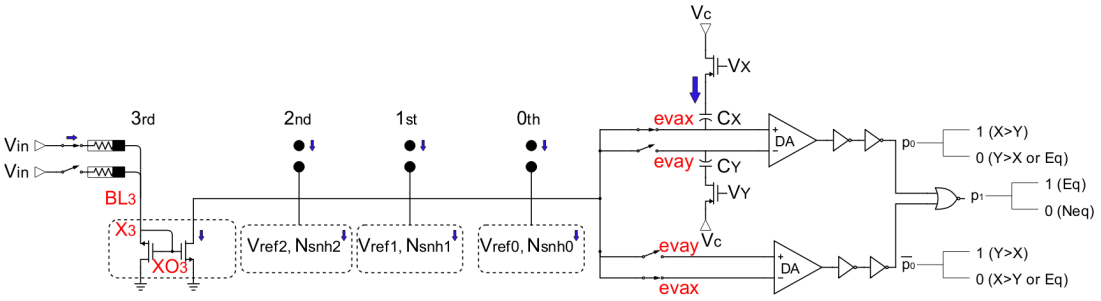


Figure 5.15: Proposed *4-bit* comparator circuit design.

The expected output of the comparator is given by

$$p_1p_0 = \begin{cases} 00, & X_4 = Y_4 \\ 01, & X_4 < Y_4 \\ 10, & X_4 > Y_4 \\ 11, & \text{Invalid} \end{cases}$$

Where, $x_3x_2x_1x_0(X_4)$ and $y_3y_2y_1y_0(Y_4)$ are two *4-bit* inputs, while the *2-bit* output is given by $p_1p_0(P_2)$. Array X_4 (row 0) and Y_4 (row 1) are stored in the memristor crossbar as bit-wise mem-conductance values of $Goff(0)$ or $Gon(1)$, in the fashion as shown in Figure 5.15. $X_4(Y_4)$ is evaluated, providing a voltage equivalent at $evax(evay)$ as $V_{evax}(V_{evay})$. These voltages are feed into two differential amplifiers DAs, with opposite polarities. Combined output of the two DAs is given by P_2 *i.e.* p_0 and p_1, respectively. Note that since each output bit is obtained separately, IFC is not required.

6 Conclusion

This chapter summarizes the contributions made as part of this thesis work and highlights the direction to be pursued for future work. Section 6.1 presents a short summary of all the preceding chapters. Key conclusions inferred regarding each chapter are thereby stated briefly. Section 6.2 lays down the recommended future research directions.

6.1 Summary

Chapter-wise summarization is presented as follows.

- **Introduction** : The first chapter put forth the primary motivation of this work, which is to provide in-memory computing solutions based on novel memristor devices. Well-known walls faced by CMOS-based computing systems were described to highlight the need of new computing paradigm. Thereafter, it drew attention to existing in-memory computing solutions at each abstraction level, *i.e.* device, primitive logic design, circuit design and architecture level, while citing associated research opportunities. Topics of research covered in this work were described, listing existing related research work and their limitation. Key contributions made as part of this work were described briefly *i.e.* quantitative benchmarking of existing primitive logic designs and propose efficient primitive and complex arithmetic functional units.

- **Background** : The second chapter introduced novel non-volatile memristor devices. The basic theory, structure and working principle of three most widely researched memristive devices *i.e.* redox-oxide, spin-torque and phase change based memristive devices were illustrated. Potential analog and digital applications offered by highly scalable, CMOS-compatible non-volatile devices with densely packed crossbar structures were tabulated. Next, different analytical and behavioral memristor models were illustrated in detail. TiO_2-based Aachen Verilog-A model was selected based on conclusive justifications, which was used to perform further analyses. Important device parameters that influence the behaviour of a memristor device were extracted through device simulations. Critical device efficiency metrics were defined and evaluated under device and voltage variations.

- **Overview and Benchmarking of Memristor-based Primitive Logic Designs** : The third chapter provided classification of existing primitive logic designs or gates supporting crossbar structures. The structure and working principle of these existing primitive logic designs were illustrated briefly. Simulation methodology covered the key design efficiency metrics taken into consideration while performing quantitative benchmarking. Results suggested that scouting logic design offered minimum latency and energy consumption per operation while provided maximum tolerance to voltage and device variations.

- **Building Primitive Arithmetic Circuit Designs** : The fourth chapter discussed the importance of an adder in modern computing systems. Overview of existing memristor based adder circuits was presented based on primarily all primitive logic designs described in the third chapter. Scouting based novel adder circuit design was proposed, illustrating in detail its key features and associated improvements qualitatively. Thereafter, results and comparison showed that proposed *4-bit* adder circuit design outperformed existing memristor based circuit design solutions. Circuit designs of a 3, 4 operand *4-bit* adders were presented to showcase the scalability of the proposed adder circuit design.

- **Building Complex Arithmetic Circuit Designs for Data-Intensive Applications** : As established in the first chapter, data-intensive applications have exposed the memory bottleneck issue faced by CMOS-based Von-Neumann computing architectures. This chapter introduced data-intensive neural networks from a hardware perspective, focusing on the dot product engines which forms the building block of such applications. Existing dot product engines were illustrated while pointing out their design inefficiencies. Modified version of the previously proposed adder circuit design was presented. The key features along with their expected improvements over existing solutions were thoroughly discussed. Result section provided comparison analyses where it was concluded that proposed dot product engine comprehensively outperforms existing design solutions. Circuit designs of *4*×*4* multiplier and *4-bit* digital comparator were presented to showcase the versatility of the proposed circuit designs.

6.2 Future Research Directions

Suggested directions of research classified under two broad topics of research are discussed below.

- **Primitive Logic Designs** :-
  1. There are numerous applications which require multiple known primitive gate functionalities to be performed. Therefore, research related to logic design exploration capable of performing multiple logic functions in a single operating cycle is suggested.
  2. New logic designs performing unprecedented logic functions can simplify countless arithmetic operating units. Research on such a topic is highly recommended.
  3. More accurate models are expected to be established with in-built variation-aware feature, similar to highly accurate and extensive monte carlo simulations for CMOS devices. This would replace the manually performed variation analyses done as part of this work, while offering a more comprehensive benchmarking of primitive logic designs.
- **Arithmetic Circuit Designs** :-
  1. Primitive arithmetic operations, especially a divider, still need to be evaluated to find an optimal memristor based circuit design solution.
  2. Additional research is required to incorporate of other features offered by memristive devices, such multi-bit storage, in circuit design solutions.
  3. Exploring the inherent properties of memristive devices to perform non-traditional ways to compute known arithmetic functions or establish whole new arithmetic functions is a excellent topic of research. For example, logarithmic, exponential or trigonometric functions may not require traditional successive approximation techniques, but rather straightforward utilization of a unique property of memristive device.

4. A processing unit developed around the proposed dot product engine to compute a full CNN/DNN algorithm is encouraged, laying pathways to new instruction sets and architectures.
5. CMOS-based control circuitry is still responsible for significant area utilization and energy consumption in the proposed memristor based in-memory computing solutions. Efforts to reduce such a major influence is recommended.

Bibliography

[1] D. E. Nikonov and I. A. Young, "Benchmarking of beyond-cmos exploratory devices for logic integrated circuits," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 1, pp. 3–11, 2015.

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, p. 80, 2008.

[3] Redox-based tera-bit memories. [Online]. Available: http://www.emrl.de/r_a_1.html#Artikel_1

[4] Power trends through 2020. [Online]. Available: http://www.itrs2.net/

[5] B. Mohammad, M. A. Jaoude, V. Kumar, D. M. Al Homouz, H. A. Nahla, M. Al-Qutayri, and N. Christoforou, "State of the art of metal oxide memristor devices," *Nanotechnology Reviews*, vol. 5, no. 3, pp. 311–329, 2016.

[6] B. K. You, M. Byun, S. Kim, and K. J. Lee, "Self-structured conductive filament nanoheater for chalcogenide phase transition," *ACS nano*, vol. 9, no. 6, pp. 6587–6594, 2015.

[7] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, and K. Bertels, "Fast boolean logic mapped on memristor crossbar," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*. IEEE, 2015, pp. 335–342.

[8] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014.

[9] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic ram," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 470–483, 2018.

[10] C. Liu, Q. Yang, B. Yan, J. Yang, X. Du, W. Zhu, H. Jiang, Q. Wu, M. Barnell, and H. Li, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 110–115.

[11] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "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.

[12] C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Y. Chen, H. Li, Q. Wu, and H. Jiang, "A spiking neuromorphic design with resistive crossbar," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[13] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.

[14] R. Gregorian and G. C. Temes, "Analog mos integrated circuits for signal processing," *New York, Wiley-Interscience, 1986, 614 p.*, 1986.

[15] B. Sklar, *Digital communications*. Prentice hall Upper Saddle River, NJ, USA:, 2001, vol. 2.

[16] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital integrated circuits*. Prentice hall Englewood Cliffs, 2002, vol. 2.

[17] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *International Conference on High Performance Computing for Computational Science*. Springer, 2010, pp. 1–25.

[18] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, vol. 15, 2008.

[19] H. Wong, C. Ahn, J. Cao, H. Chen, S. Fong, Z. Jiang, C. Neumann, S. Qin, J. Sohn, Y. Wu *et al.*, "Stanford memory trends," *tech. report*, 2016.

[20] K. Roy, M. Sharad, D. Fan, and K. Yogendra, "Beyond charge-based computation: Boolean and non-boolean computing with spin torque devices," in *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*. IEEE Press, 2013, pp. 139–142.

[21] S. Hamdioui, S. Kvatinsky, G. Cauwenberghs, L. Xie, N. Wald, S. Joshi, H. M. Elsayed, H. Corporaal, and K. Bertels, "Memristor for computing: Myth or reality?" in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 722–731.

[22] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[23] X. Bi, C. Zhang, H. Li, Y. Chen, and R. E. Pino, "Spintronic memristor based temperature sensor design with cmos current reference," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 1301–1306.

[24] Shifting bathtub. [Online]. Available: https://ocw.tudelft.nl/wp-content/uploads/Module_14_Reliability.pdf

[25] S. Hamdioui, L. Xie, H. A. D. Nguyen, M. Taouil, K. Bertels, H. Corporaal, H. Jiao, F. Catthoor, D. Wouters, L. Eike *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *Proceedings of the 2015 design, automation & test in Europe conference & exhibition*. EDA Consortium, 2015, pp. 1718–1725.

[26] L. Wang and K. Skadron, "Implications of the power wall: Dim cores and reconfigurable logic," *IEEE Micro*, vol. 33, no. 5, pp. 40–48, 2013.

[27] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "A case for intelligent ram," *IEEE micro*, vol. 17, no. 2, pp. 34–44, 1997.

[28] H. Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobbs journal*, vol. 30, no. 3, pp. 202–210, 2005.

[29] J. Chhugani, A. D. Nguyen, V. W. Lee, W. Macy, M. Hagog, Y.-K. Chen, A. Baransi, S. Kumar, and P. Dubey, "Efficient implementation of sorting on multi-core simd cpu architecture," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1313–1324, 2008.

[30] M. J. Mayfield, F. P. O'connell, and D. S. Ray, "Cache prefetching of l2 and l3," Sep. 3 2002, uS Patent 6,446,167.

[31] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Transactions on Computers*, vol. 100, no. 12, pp. 1145–1155, 1975.

[32] R. Lin and M. Margala, "Multiplier-based processor-in-memory architectures for image and graphics processing," Jan. 23 2007, uS Patent 7,167,890.

[33] X. Liu, M. Mao, B. Liu, H. Li, Y. Chen, B. Li, Y. Wang, H. Jiang, M. Barnell, Q. Wu *et al.*, "Reno: A high-efficient reconfigurable neuromorphic computing accelerator design," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[34] T. Luo, S. Liu, L. Li, Y. Wang, S. Zhang, T. Chen, Z. Xu, O. Temam, and Y. Chen, "Dadiannao: A neural network supercomputer," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 73–88, 2017.

[35] C. Merkel, R. Hasan, N. Soures, D. Kudithipudi, T. Taha, S. Agarwal, and M. Marinella, "Neuromemristive systems: Boosting efficiency through brain-inspired computing," *Computer*, vol. 49, no. 10, pp. 56–64, 2016.

[36] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nature communications*, vol. 9, no. 1, p. 2331, 2018.

[37] P. Koeberl, Ü. Kocabaş, and A.-R. Sadeghi, "Memristor pufs: a new generation of memory-based physically unclonable functions," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 428–431.

[38] A. Mazady, M. T. Rahman, D. Forte, and M. Anwar, "Memristor pufa security primitive: Theory and experiment," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 222–229, 2015.

[39] M. Kund, G. Beitel, C.-U. Pinnow, T. Rohr, J. Schumann, R. Symanczyk, K. Ufert, and G. Muller, "Conductive bridging ram (cbram): An emerging non-volatile memory technology scalable to sub 20nm," in *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.* IEEE, 2005, pp. 754–757.

[40] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung *et al.*, "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, vol. 52, no. 4/5, p. 465, 2008.

[41] Y. Huai *et al.*, "Spin-transfer torque mram (stt-mram): Challenges and prospects," *AAPPS bulletin*, vol. 18, no. 6, pp. 33–40, 2008.

[42] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie, "Design implications of memristor-based rram cross-point structures," in *2011 Design, Automation & Test in Europe.* IEEE, 2011, pp. 1–6.

[43] A. L. Lacaita, "Phase change memories: State-of-the-art, challenges and perspectives," *Solid-State Electronics*, vol. 50, no. 1, pp. 24–31, 2006.

[44] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Mossé, "Increasing pcm main memory lifetime," in *Proceedings of the conference on design, automation and test in Europe.* European Design and Automation Association, 2010, pp. 914–919.

[45] B. Del Bel, J. Kim, C. H. Kim, and S. S. Sapatnekar, "Improving stt-mram density through multibit error correction," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2014, pp. 1–6.

[46] G. W. Burr, M. J. Brightsky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis *et al.*, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, 2016.

[47] L. Xie, H. A. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. AlFailakawi, and S. Hamdioui, "Scouting logic: A novel memristor-based logic design for resistive computing," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI).* IEEE, 2017, pp. 176–181.

[48] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, D. B. Strukov *et al.*, "Memristor- cmos hybrid integrated circuits for reconfigurable logic," *Nano letters*, vol. 9, no. 10, pp. 3640–3645, 2009.

[49] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "memristiveswitches enable statefullogic operations via material implication," *Nature*, vol. 464, no. 7290, p. 873, 2010.

[50] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magicmemristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.

[51] P.-E. Gaillardon, L. Amarú, A. Siemon, E. Linn, R. Waser, A. Chattopadhyay, and G. De Micheli, "The programmable logic-in-memory (plim) computer," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Ieee, 2016, pp. 427–432.

[52] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Memristor-based imply logic design procedure," in *2011 IEEE 29th International Conference on Computer Design (ICCD)*. IEEE, 2011, pp. 142–147.

[53] L. Guckert and E. E. Swartzlander, "Optimized memristor-based multipliers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 2, pp. 373–385, 2017.

[54] F. Merrikh-Bayat and S. B. Shouraki, "Memristor-based circuits for performing basic arithmetic operations," *Procedia Computer Science*, vol. 3, pp. 128–132, 2011.

[55] X. Wang, Q. Wu, Q. Chen, and Z. Zeng, "A novel design for memristor-based multiplexer via not-material implication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1436–1444, 2018.

[56] O. A. Olumodeji and M. Gottardi, "Memristor-based comparator with programmable hysteresis," in *2015 11th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*. IEEE, 2015, pp. 232–235.

[57] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 27–39.

[58] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, p. 384010, 2013.

[59] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[60] J.-M. Lee, C.-M. Lee, L.-X. Ye, J.-P. Su, and T.-H. Wu, "Switching properties for mgo-based magnetic tunnel junction devices driven by spin-transfer torque in the nanosecond regime," *IEEE Transactions on magnetics*, vol. 47, no. 3, pp. 629–632, 2011.

[61] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges," *Advanced materials*, vol. 21, no. 25-26, pp. 2632–2663, 2009.

[62] B. Linares-Barranco and T. Serrano-Gotarredona, "Memristance can explain spike-time-dependent-plasticity in neural synapses," 2009.

[63] M. Marinella, "The future of memory," in *2013 IEEE Aerospace Conference*. IEEE, 2013, pp. 1–11.

[64] B. Muthuswamy, "Implementing memristor based chaotic circuits," *International Journal of Bifurcation and Chaos*, vol. 20, no. 05, pp. 1335–1350, 2010.

[65] Y. V. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 8, pp. 1857–1864, 2010.

[66] S. Smaili and Y. Massoud, "Studying the effect of memristor state variability on the gain of memristor-based tunable amplifiers," in *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2013, pp. 912–915.

[67] M. Itoh and L. O. Chua, "Memristor oscillators," *International journal of bifurcation and chaos*, vol. 18, no. 11, pp. 3183–3206, 2008.

[68] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature nanotechnology*, vol. 3, no. 7, p. 429, 2008.

[69] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, no. 7, p. 074508, 2009.

[70] J. G. Simmons, "Generalized formula for the electric tunnel effect between similar electrodes separated by a thin insulating film," *Journal of applied physics*, vol. 34, no. 6, pp. 1793–1803, 1963.

[71] A. Hardtdegen, C. La Torre, F. Cüppers, S. Menzel, R. Waser, and S. Hoffmann-Eifert, "Improved switching stability and the effect of an internal series resistor in hfo 2/tio x bilayer reram cells," *IEEE Transactions on Electron Devices*, vol. 65, no. 8, pp. 3229–3236, 2018.

[72] X. Guan, S. Yu, and H.-S. P. Wong, "A spice compact model of metal oxide resistive switching memory with variations," *IEEE electron device letters*, vol. 33, no. 10, pp. 1405–1407, 2012.

[73] S. H. Szczepankiewicz, J. A. Moss, and M. R. Hoffmann, "Slow surface charge trapping kinetics on irradiated tio2," *The Journal of Physical Chemistry B*, vol. 106, no. 11, pp. 2922–2927, 2002.

[74] A. Ascoli, F. Corinto, V. Senger, and R. Tetzlaff, "Memristor model comparison," *IEEE Circuits and Systems Magazine*, vol. 13, no. 2, pp. 89–105, 2013.

[75] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Team: Threshold adaptive memristor model," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211–221, 2013.

[76] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "Vteam: A general model for voltage-controlled memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, 2015.

[77] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: properties of basic electrical circuits," *European Journal of Physics*, vol. 30, no. 4, p. 661, 2009.

[78] Z. Biolek, D. Biolek, and V. Biolkova, "Spice model of memristor with nonlinear dopant drift." *Radioengineering*, vol. 18, no. 2, 2009.

[79] T. Prodromakis, B. P. Peh, C. Papavassiliou, and C. Toumazou, "A versatile memristor model with nonlinear dopant kinetics," *IEEE transactions on electron devices*, vol. 58, no. 9, pp. 3099–3105, 2011.

[80] H. A. Du Nguyen, J. Yu, L. Xie, M. Taouil, S. Hamdioui, and D. Fey, "Memristive devices for computing: Beyond cmos and beyond von neumann," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2017, pp. 1–10.

[81] G. Snider, "Computing with hysteretic resistor crossbars," *Applied Physics A*, vol. 80, no. 6, pp. 1165–1172, 2005.

[82] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 173.

[83] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, vol. 4. IEEE, 2003, pp. IV–IV.

[84] M. Chu, B. Kim, S. Park, H. Hwang, M. Jeon, B. H. Lee, and B.-G. Lee, "Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2410–2419, 2015.

[85] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[86] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8609–8613.

[87] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3. IEEE Computer Society, 2012, pp. 356–367.

[88] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2012, pp. 449–460.

[89] L. Kull, T. Toifl, M. Schmatz, P. A. Francese, C. Menolfi, M. Braendli, M. Kossel, T. Morf, T. M. Andersen, and Y. Leblebici, "A 3.1 mw 8b 1.2 gs/s single-channel asynchronous sar adc with alternate comparators for enhanced speed in 32 nm digital soi cmos," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 3049–3058, 2013.

Shmoo Plots

A

V_r \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
0.2	F	F	F	F	F	F
0.3	P (2%)	P (161%)	P (256%)	P (375%)	P (587%)	P (813%)
0.4	P (60%)	P (296%)	P (439%)	P (620%)	P (942%)	P (1285%)
0.5	P (120%)	P (444%)	P (642%)	P (890%)	P (1333%)	P (1804%)
0.6	P (120%)	P (444%)	P (642%)	P (890%)	P (1333%)	P (1804%)
0.7	P (130%)	P (469%)	P (675%)	P (935%)	P (1398%)	P (1890%)
0.8	P (190%)	P (790%)	P (1113%)	P (1520%)	P (2244%)	P (3015%)
0.9	P (70%)	P (345%)	P (507%)	P (710%)	P (1072%)	P (1458%)
1	F	P (98%)	P (170%)	P (260%)	P (421%)	P (592%)
1.1	F	P (37%)	P (87%)	P (150%)	P (262%)	P (381%)
1.2	F	F	P (20%)	P (60%)	P (132%)	P (208%)
1.3	F	F	P (1%)	P (35%)	P (95%)	P (160%)
1.4	F	F	F	F	P (37%)	P (83%)
1.5	F	F	F	F	P (16%)	P (54%)
1.6	F	F	F	F	F	P (30%)
1.7	F	F	F	F	F	P (19%)
1.8	F	F	F	F	F	F

Table A.1: Shmoo plot for Scouting OR.

V_r \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
0.2	F	F	F	F	F	F
0.3	F	F	F	F	F	F
0.4	F	F	F	F	F	F
0.5	F	F	F	F	F	F
0.6	F	F	F	F	F	F
0.7	F	F	F	F	F	F
0.8	P (35%)	P (235%)	P (359%)	P (510%)	P (802%)	P (1073%)
0.9	P (10%)	P (172%)	P (271%)	P (395%)	P (616%)	P (852%)
1	F	P (98%)	P (170%)	P (260%)	P (421%)	P (592%)
1.1	F	P (37%)	P (87%)	P (150%)	P (262%)	P (381%)
1.2	F	F	P (20%)	P (60%)	P (132%)	P (208%)
1.3	F	F	P (1%)	P (35%)	P (95%)	P (160%)
1.4	F	F	F	F	P (37%)	P (83%)
1.5	F	F	F	F	P (16%)	P (54%)
1.6	F	F	F	F	F	P (30%)
1.7	F	F	F	F	F	P (19%)
1.8	F	F	F	F	F	F

Table A.2: Shmoo plot for Scouting AND/XOR.

V_{set} \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
0.6	F	F	F	F	F	F
0.7	P (3%)	F	F	F	F	F
0.8	P (20%)	F	F	F	F	F
0.9	P (40%)	F	F	F	F	F
1	P (40%)	P (5%)	F	F	F	F
1.1	P (30%)	P (40%)	P (3%)	F	F	F
1.2	P (3%)	P (40%)	P (25%)	P (3%)	F	F
1.3	F	P (40%)	P (30%)	P (10%)	F	F
1.4	F	P (40%)	P (40%)	P (25%)	P (2%)	F
1.5	F	P (30%)	P (40%)	P (40%)	P (10%)	F
1.6	F	F	P (35%)	P (40%)	P (25%)	P (3%)
1.7	F	F	P (25%)	P (40%)	P (25%)	P (10%)
1.8	F	F	F	P (30%)	P (40%)	P (30%)
1.9	F	F	F	P (10%)	P (40%)	P (40%)
2	F	F	F	F	P (35%)	P (40%)
2.1	F	F	F	F	F	P (40%)
2.2	F	F	F	F	F	P (35%)
2.3	F	F	F	F	F	P (30%)
2.4	F	F	F	F	F	P (10%)
2.5	F	F	F	F	F	F
2.6	F	F	F	F	F	F

Table A.3: Shmoo plot for Snider-COPY. V_{set} is shown in the first column while $V_{cond} = V_{set}/2$, since it provides the best result.

V_{set} \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
1.1	F	F	F	F	F	F
1.2	F	F	F	F	F	F
1.3	F	F	F	F	F	F
1.4	F	F	F	F	F	F
1.5	F	F	F	F	F	F
1.6	F	F	F	F	F	F
1.7	P (1%)	P (5%)	P (35%)	P (20%)	P (2%)	F
1.8	F	F	P (15%)	P (35%)	P (20%)	P (3%)
1.9	F	F	P (3%)	P (35%)	P (30%)	P (15%)
2	F	F	F	P (20%)	P (40%)	P (25%)
2.1	F	F	F	P (3%)	P (40%)	P (35%)
2.2	F	F	F	F	P (30%)	P (45%)
2.3	F	F	F	F	P (20%)	P (40%)
2.4	F	F	F	F	P (3%)	P (40%)
2.5	F	F	F	F	P (1%)	P (30%)
2.6	F	F	F	F	F	P (20%)
2.7	F	F	F	F	F	P (15%)
2.8	F	F	F	F	F	P (3%)
2.9	F	F	F	F	F	F
3	F	F	F	F	F	F

Table A.4: Shmoo plot for Snider-NOT. V_{set} is shown in the first column while $V_{cond} = V_{set}/2$, since it provides the best result.

V_{set} \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
1.1	F	F	F	F	F	F
1.2	F	F	F	F	F	F
1.3	F	P (20%)	P (10%)	F	F	F
1.4	P (3%)	P (25%)	P (20%)	P (15%)	F	F
1.5	F	P (15%)	P (30%)	P (20%)	P (3%)	F
1.6	F	F	P (20%)	P (30%)	P (10%)	F
1.7	F	F	P (5%)	P (30%)	P (20%)	P (10%)
1.8	F	F	F	P (10%)	P (30%)	P (15%)
1.9	F	F	F	P (3%)	P (40%)	P (30%)
2	F	F	F	F	P (30%)	P (30%)
2.1	F	F	F	F	P (10%)	P (40%)
2.2	F	F	F	F	P (3%)	P (30%)
2.3	F	F	F	F	F	P (20%)
2.4	F	F	F	F	F	P (10%)
2.5	F	F	F	F	F	P (3%)
2.6	F	F	F	F	F	F
2.7	F	F	F	F	F	F
2.8	F	F	F	F	F	F
2.9	F	F	F	F	F	F
3	F	F	F	F	F	F

Table A.5: Shmoo plot for Snider NAND/NOR. V_{set} is shown in the first column while $V_{cond} = V_{set}/2$, since it provides the best result.

V_p/V_q \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
0.6	F	F	F	F	F	F
0.7	F	F	F	F	F	F
0.8	P (50%)	F	F	F	F	F
0.9	P (55%)	F	F	F	F	F
1	P (60%)	P (20%)	F	F	F	F
1.1	P (65%)	P (30%)	P (10%)	F	F	F
1.2	P (70%)	P (50%)	P (30%)	P (10%)	F	F
1.3	P (75%)	P (55%)	P (40%)	P (20%)	F	F
1.4	P (>75%)	P (60%)	P (50%)	P (30%)	P (2%)	F
1.5	P (>75%)	P (65%)	P (55%)	P (40%)	P (20%)	P (2%)
1.6	P (>75%)	P (70%)	P (60%)	P (50%)	P (30%)	P (10%)
1.7	P (>75%)	P (75%)	P (65%)	P (55%)	P (40%)	P (20%)
1.8	P (>75%)	P (>75%)	P (70%)	P (60%)	P (45%)	P (20%)
1.9	P (>75%)	P (>75%)	P (75%)	P (65%)	P (50%)	P (30%)
2	P (>75%)	P (>75%)	P (>75%)	P (70%)	P (55%)	P (40%)
2.1	P (>75%)	P (>75%)	P (>75%)	P (75%)	P (60%)	P (45%)
2.2	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (65%)	P (50%)
2.3	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (70%)	P (55%)
2.4	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (75%)	P (60%)
2.5	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (65%)
2.6	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (>75%)	P (70%)

Table A.6: Shmoo plot for MAJ.

V_{set} \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
0.6	F	F	F	F	F	F
0.7	F	F	F	F	F	F
0.8	F	F	F	F	F	F
0.9	F	F	F	F	F	F
1	P (3%)	F	F	F	F	F
1.1	P (7%)	P (5%)	F	F	F	F
1.2	P (15%)	P (20%)	P (3%)	F	F	F
1.3	P (20%)	P (30%)	P (5%)	P (3%)	F	F
1.4	P (25%)	P (30%)	P (20%)	P (10%)	P (2%)	F
1.5	P (20%)	P (30%)	P (20%)	P (20%)	P (10%)	F
1.6	P (10%)	P (30%)	P (20%)	P (20%)	P (25%)	P (3%)
1.7	P (5%)	P (30%)	P (30%)	P (25%)	P (25%)	P (10%)
1.8	F	P (20%)	P (20%)	P (35%)	P (30%)	P (15%)
1.9	F	P (15%)	P (20%)	P (20%)	P (20%)	P (20%)
2	F	P (10%)	P (20%)	P (20%)	P (15%)	P (25%)
2.1	F	P (5%)	P (15%)	P (15%)	P (15%)	P (20%)
2.2	F	F	P (10%)	P (15%)	P (10%)	P (20%)
2.3	F	F	P (4%)	P (10%)	P (10%)	P (15%)
2.4	F	F	F	P (10%)	P (10%)	P (15%)
2.5	F	F	F	P (3%)	P (10%)	P (15%)
2.6	F	F	F	F	P (5%)	P (10%)
2.7	F	F	F	F	P (3%)	P (10%)
2.8	F	F	F	F	F	P (3%)
2.9	F	F	F	F	F	F
3	F	F	F	F	F	F

Table A.7: Shmoo plot for FBL-COPY (MFO). V_{set} is shown in the first column while $V_{cond} = V_{set}/2$, since it provides the best result.

V_{set} \ $Roff/Ron$	**10**	**50**	**100**	**200**	**500**	**1000**
1.1	F	F	F	F	F	F
1.2	F	F	F	F	F	F
1.3	P (5%)	P (20%)	P (2%)	F	F	F
1.4	F	P (25%)	P (25%)	P (2%)	F	F
1.5	F	P (5%)	P (30%)	P (20%)	P (2%)	F
1.6	F	F	P (10%)	P (30%)	P (10%)	F
1.7	F	F	F	P (25%)	P (20%)	P (2%)
1.8	F	F	F	P (10%)	P (35%)	P (15%)
1.9	F	F	F	F	P (35%)	P (30%)
2	F	F	F	F	P 15%)	P (30%)
2.1	F	F	F	F	P (10%)	P (40%)
2.2	F	F	F	F	F	P (30%)
2.3	F	F	F	F	F	P (10%)
2.4	F	F	F	F	F	P (8%)
2.5	F	F	F	F	F	F
2.6	F	F	F	F	F	F

Table A.8: Shmoo plot for FBL-NAND/NOR (MFO). V_{set} is shown in the first column while $V_{cond} = V_{set}/2$, since it provides the best result.