μSystems Group
School of Engineering
Newcastle University, UK

Energy-Efficient Mixed-Signal Multiplier Design using Memristive Technologies

Shengqi Yu

Supervisors:

Dr. Rishad Shafik

Prof. Alex Yakovlev

A thesis submitted for the degree of
Doctor of Philosophy

24th March 2022

Abstract

Energy efficiency and performance are two of the most important design considerations for computing applications, e.g., artificial intelligence at the edge and Internet of things empowered by limited energy supply from batteries or energy harvesters. For these applications, arithmetic computation is key, with multiplication and addition being the "must-have" core functionalities. Traditional approaches to these are primarily based on cascaded logic chains with long carry propagation circuits that contribute to high energy consumption and latencies. Additionally, these circuits exploit digital interfaces at both inputs and outputs, which require complex signal conversion circuits when designed using analogue methods. This thesis presents original research focused on developing low-energy and high-speed multiplication hardware. The core technology developed in this work is a novel digital-in/analogue-out mixed signal multiplication method based on a single-bit multiplication cell. The cell consists of a resistive memory bit controlled by a transistor switch. The single-bit memory cell is implemented using memristor devices, which provide non-volatile storage and avoid capacitive or inductive elements. This type of single-bit multiplication cell takes two single-bit input operands (multiplier and multiplicand). One (e.g., the multiplier) is encoded in the form of a Boolean voltage and the other (e.g., the multiplicand) is encoded in the memristor's conductance, also set to Boolean values. The cell current then encodes the Boolean product following the Ohm's Law. The single-bit multiplication cells are then assembled into multi-bit multipliers using a crossbar matrix structure, which directly implements the long-multiplication algorithm. Across the crossbar, Kirchhoff's Current Law ensures that the cell currents are summed up to form the final overall product, forming a digital-in/analogue-out mixed signal design. The entire Ohm's law-Kirchhoff's Current Law operation is instantaneous in the absence of capacitive and inductive elements. With Kirchhoff's Current Law,

this type of mixed-signal multiplier eliminates the need for passing carries to the left. This saves both time and energy compared with conventional digital amplifiers, which need costly and potentially long logic chains for carry handling. By using multiple memristors in an single-bit multiplication cell, costly current mirrors can be avoided from the crossbar. The core digital-in/analogue-out multiplication method can have direct applications in Internet of things nodes, like multiplying digital-to-analogue converters. One advantage of using the proposed multiplier in this application comes from the asymmetry between the two input operands. One of them, saved in memoristor conductances, is the best changed less frequently than the other, represented by voltages, precisely what an multiplying digital-to-analogue converter aims for. This digital-in/analogue-out multiplier is further developed into a digital-in/digital-out multiplier with reduced output precision, with the same bit width for both the operands and the product.

We envisage our design will be useful in applications where multiple multiply-and-add units are assembled into larger structures, such as in neural networks. With the same bit width for both inputs and outputs, multipliers of this design can be cascaded a straightforward manner for larger networks. The multiplier designs are implemented in 65 nm technology using Cadence Virtuoso based analogue simulations. The designs are shown to have significant speed and energy advantages over existing state of the art and the machine learning experiments demonstrate the correctness and usability of the reduced-precision multiplication scheme for artificial intelligence applications.

Contents

List of Figures

List of Tables

List of Acronyms

1T1M - Single Transistor Single Memristor

1TxM - Single Transistor Multiple Memristors

A2D - Analogue to Digital

ABM - Analogue Behavioural Model

ADC - Analogue-to-Digital Converter

AI - Artificial Intelligence

ASM - Alphabet Set Multiplier

CF - Conductive Filament

CL - Column Line

CLA - Carry Look-Ahead

CM - Current Mirror

CMA - Current Accumulation

CMOS - Complementary Metal–Oxide–Semiconductor

Cu:ZnO - Copper doped:Zinc Oxide

DAC - Digital-to-Analogue Converter

DC - Direct Current

DI/AO - Digital-In/Analogue-Out

DI/DO - Digital-In/Digital-Out

GL - Gate Line

HCS - High Conductance State

HRS - High Resistance State

HVS - High Voltage State

IMC - In-Memory Computing

IMP/IMPLY - Material Implication

I/O - Input/Output

IoT - Internet of Things

ITO - Indium Tin Oxide

I-V - Current-Voltage

KCL - Kirchhoff's Current Law

LC - Logic Cell

LCS - Low Conductance State

LRS - Low Resistance State

LSB - Least Significant Bit

LVS - Low Voltage State

MAC - Multiply-Accumulate

MAD - Memristors-as- Drivers

MDAC - Multiplying Digital-to-Analogue Converter

MIG - Majority Inverter Graph

MIM - Metal-Insulator-Metal

MLA - Machine Learning Algorithm

MLP - Multi-Layer Perceptrons

MNIST - Modified National Institute of Standards and Technology

MOSFET - Metal-Oxide-Semiconductor Field-Effect Transistor

MSB - Most Significant Bit

NMOS - N-channel Metal-Oxide-Semiconductor

NN - Neural Network

PMOS - P-channel Metal-Oxide-Semiconductor

PP - Partial Product

PTQ - Post-Training Quantization

QAT - Quantization-Aware Training

ReLU - Rectified Linear Unit

RL - Row Line

ROM - Read Only Memory

RRAM - Resistive Random Access Memory

RS - Resistive Switch

RSV - Reset Voltage

SBMC - Single-Bit Multiplication Cell

SBS - Single Bar Source

SL - Source Line

SV - Set Voltage

TEAM - ThrEshold Adaptive Memristor

TIA - Trans-Impedance Amplifier

TiO_2 - Titanium Dioxide

UT - Unintended Tuning

VMM - Vector-Matrix Multiplication

VTEAM - Voltage ThrEshold Adaptive Memristor

xM - Multiple Memristors

yT - Multiple Transistors

yTxM - Multiple Transistors Multiple Memristors

Acknowledgments

I would like to express my gratitude to Dr. Rishad Shafik, my supervisor, for his passion, knowledge, and guidance throughout my study in Newcastle. Especially the mental support in my toughest period. He educated me to become a better researcher, furthermore, a better man for my family. I would also like to thank my second supervisor, Prof. Alex Yakovlev, for his wisdom and encouragement in improving my research skills. I am also grateful to Petros Missailidis, who supervised me during my MSc studies; he introduced me to the exciting microsystem group.

I also extend my thanks to Dr. Fei Xia, Dr. Ahmed Soltan, and Dr. Thanasin Bunnam, who acted as my unofficial supervisors. Dr. Fei Xia helped me in improving my publications' quality and provided me with invaluable guidance during my postgraduate research. Dr. Ahmed Soltan shared his huge experience in the field of analog circuit design. Dr. Thanasin Bunnam closely collaborated with me, validating the proposed design in Chapter 5. I would like to thank Dr. Kaiyuan Gao and Dr. Yuqing Xu, who gave me support while EDA tool training.

I am thankful to my wife, Dr. Xue Han, for all her love, strength, patience, and support throughout my Ph.D. She gave birth to and raised our first child during my absence, which is an invaluable gift in my tough period. She supported me in any difficulty; also, she shared me with all happiness. I am grateful to my family for their concern and support. My thanks also to all my friends, especially to Dr. Adrian Wheeldon, who made my Ph.D. journey a much more fun journey with various social activities.

This research was supported by EPSRC IAA project Whisperable AI Power Management (ref: NU-007755, Newcastle University, UK) and Northern Accelerator grant on Low-Power AI Circuits (NU-009397, Newcastle University). I am grateful for their support.

Publications

The following is a list of publications related to this thesis.

S. Yu, A. Soltan, R. Shafik, T. Bunnam, F. Xia, D. Balsamo and A. Yakovlev, "Current-Mode Carry-Free Multiplier Design using a Memristor-Transistor Crossbar Architecture," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020, pp. 638-641, doi: 10.23919/DATE48585.2020.9116417.

> This paper presents the titanium oxide (i.e., TiO_2) 1T1M multiplication cell and crossbar multiplier design using the UMC 65 nm library. It is the first publication to introduce a current-mode multiplication with memristive cells. This work appears in Section 3.1.

S. Yu, R. Shafik, T. Bunnam, K. Chen and A. Yakovlev, "Self-Amplifying Current-Mode Multiplier Design using a Multi-Memristor Crossbar Cell Structure," 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020, pp. 1-4, doi: 10.1109/ICECS49266.2020.9294797.

> This paper extend the titanium oxide (i.e., TiO_2) 1T1M multiplication cell to 1TxM multiplication cell. Meanwhile, apply it in a CM eliminated crossbar multiplier design using the UMC 65 nm library. Moreover, this work discusses the performance development by multiplication cell structure variation in Section 4.2.

S. Yu, R. Shafik, T. Bunnam, K. Chen and A. Yakovlev, "Optimized Multi-Memristor Model based Low Energy and Resilient Current-Mode Multiplier Design," 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 1230-1233, doi: 10.23919/DATE51398.2021.9473926.

> This paper compares crossbar multiplier based on the titanium oxide (i.e., TiO_2) 1TxM multiplication cell and doped zinc oxide (i.e., ferroelectric Cu:ZnO) 1TxM

multiplication cell with other multipliers in performance and energy efficiency. The results show the favorable energy efficiency of the 1TxM crossbar multiplier over other designs and motivate the 1TxM crossbar multiplier design in Section 4.2.

S. Yu, F. Xia, R. Shafik, D. Balsamo, and A. Yakovlev, *Approximate Digital-In Analog-Out Multiplier with Asymmetric Nonvolatility and Low Energy Consumption.* In: Integration.

This paper compares crossbar multiplier based on the titanium oxide (i.e., TiO_2) 1TxM multiplication cell and doped zinc oxide (i.e., ferroelectric Cu:ZnO) 1TxM multiplication cell with other multiplication circuits in latency, performance, and energy efficiency. High memristance 1TxM crossbar multiplier stands out in latency and energy efficiency to other designs. Moreover, the transistor memristor crossbar multipliers are summarized in Chapter 4.

Yu, S., T. Bunnam, S. Triamlumlerd, M. Pracha, F. Xia, R. Shafik and A. Yakovlev, *Energy-Efficient Neural Network Design using Memristive MAC Unit.* In: Frontiers in Electronics, 28.

This paper presents a MAC unit based on doped zinc oxide (i.e., ferroelectric Cu:ZnO) yTxM multiplication cell crossbar multiplier. Apply this MAC unit in an NN applications and compare the MNIST classification accuracy. The pure fully-connected layer is acceptable for MNIST classification motivates the MAC unit design in Section 5.4.

The rest papers are not presented in this thesis which I was involved as a contributor:

T. Lan, F. Xia, G. Mao, S. Yu, R. Shafik and A. Yakovlev, "Editable asynchronous control logic for SAR ADCs," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), (in progress).

G. Mao, F. Xia, T. Lan, S. Yu, R. Shafik and A. Yakovlev, "Automated Mapping of Asynchronous Circuits on FPGA under Timing Constraints," In 2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) (pp. 104-109). IEEE.

T. Lan, G. Mao, F. Xia, R. Shafik, A. Yakovlev and S. Yu, "An Asynchronous Tsetlin Automaton Architecture with Integrated Non-volatile Memory," In 2022 International Symposium on the Tsetlin Machine (ISTM) (pp. 37-40). IEEE.

O. Ghazal, S. Singh, T. Rahman, S. Yu, Y. Zheng, D. Balsamo, S. Patkar, F. Merchant, F. Xia, A. Yakovlev, and R. Shafik, "IMBUE: In-Memory Boolean-to-CUrrent Inference ArchitecturE for Tsetlin Machines," In ACM/IEEE International Symposium on Low Power Electronics and Design 2023 (ISLPED 2023).

Chapter 1

Introduction

1.1 Challenges in Emerging Applications

Over the past half-century, the requirement of high energy efficiency and performance in computing has been sustained by the down-scaling of metal-oxide-semiconductor field-effect transistors (MOSFET). This method enabled complementary metal-oxide-semiconductor (CMOS) systems to maintain an exponential increase of the device's density in per unit area at each technology generation [1]. In the recent nano-scale generation, the energy efficiency has stopped commensurately growing with circuit performance. It is partly because the thermal power density from a large number of devices in the unit area leads to device performance degradation. This is further exacerbated by the performance gap between the central processing unit (i.e., data processing part) and the computer memory (i.e., data storing part) increases as the data volume increases. These issues leads to difficulties when trying to meet performance and energy efficiency requirements of emerging electronic applications such as artificial intelligence (AI) and Internet of things (IoT) AI applications usually based on neuron networks (NNs) [2,3].

Machine learning using NNs and other AI methods involves multiple iterations of arithmetic operations, with data flow between processing elements and memory, and is a significant bottleneck for conventional computers [2–4], this phenomenon is known as

the "memory wall" [5, 6]. In order to address the memory wall challenge, researchers have proposed a shift from traditional Von Neumann computing architectures to non-Von Neumann computing architectures. In-memory computing (IMC) is an example of non-Von Neumann computing architectures.

IMC using non-volatile memory technologies, provide ways of reducing the amounts of data flow required for AI applications, including NNs [4, 7], by locating the computation close to or at the memory. Using non-volatile memory, IMC can further reduce the number of data movements. Moreover, non-volatile data storage helps sustain the continuity of computing flow through power cuts or interruptions in edge devices, which are powered by unreliable supplies, such as energy harvester.

As a result, non-Von Neumann architectures have been a popular area of research aimed at improving energy efficiency and performance. An example area of such research is related to the use of resistive memory such as memristors, which has shown promises of significantly improving key performance metrics such as operating frequency (increasing by 15% relative to the scaled supply voltage), energy efficiency (increasing by 35% for a given per switching performance), footprint area cost (reducing by 35% on chip), and scaled die cost (reducing by 20% while no more than 30% increase of wafer cost) [8–10].

Another application area that is seeing a similar rapid development as AI is the Internet of Things (IoT), devices communicate end-to-end to build the machine-to-machine interaction [11].

Arithmetic operations are central to modern AI applications and IoT [12, 13]. In these operations, multiplication plays a crucial role with significant impact on performance and energy efficiency, especially because traditional multiplier circuits feature complex partial product generation and carry propagation logic chains [14]. As such, reducing the energy consumption of multipliers, is an ongoing design challenge.

For low-complexity multiplication, reducing precision is a promising method. For this, pruning the carry chains to a minimum proportion while also maintaining an acceptable precision has been proposed by numerous approximate and speculative circuit designs [15]. These designs require careful synergy of operating voltages and frequencies to balance energy and performance trade-offs [16]. Moreover, the accumulation of

imprecision and errors in cascaded workloads needs mitigation strategies which adds more complexity to the logic chains [17]. Consequently, the usability of voltage-mode proportional carry pruning schemes is still limited.

Many IoT applications, such as neuromorphic, signal processing and control, require the multiplier output in an analogue form with digital input interfaces [18, 19]. This is conventionally satisfied by attaching a digital-to-analogue converter (DAC) device to the output of a digital circuit [20]. Meanwhile, the increase of real-time data produced by relative sensors in edge devices and the number of edge devices set a much higher requirement for the processing speed in IoT applications. However, DAC circuits add to the energy and performance costs that depend on the precision of the digital multipliers. Therefore, resistive switch (RS) emerging devices bring analogue domain data processing in hardware back to the forefront [8,9,21].

1.2 Energy-efficient Multiplier Design

Since pure digital multiplier design needs positive related scale of DACs, the digital multiplier will be costly in high density analogue to digital (A2D) conversion applications. Therefore, the inevitable A2D conversion in IoT edge devices and the higher requirement of processing speed and energy efficiency makes a limited space for pure digital design in IoT.

Multiplication with mixed-signal arithmetic circuits is a potential alternative for achieving low-cost analogue output directly [22] and has a successful academic and commercial history. An example is the multiplying digital-to-analogue converter (MDAC) circuit, which multiplies a digital number by a usually analogue reference signal to produce an analogue output [23–25]. Digital-in/analogue-out (DI/AO), where both operands are digital, but the product is analogue, has remained under explored. One of the main areas of contribution by this thesis is in this area.

Table 1.1 lists different types of multipliers (including MDACs) by the digital and analogue nature of their input and output signals. In digital design of DI/AO, both multiplication operands are in digital, and product is initially in digital. Thus DAC is needed for the analogue product. Conversely, analogue design has analogue

Table 1.1: Digital-in/Analogue-out Multiplier Designs

Design	Multiplier	Multiplicand	Product	Non-volatility
Digital	Digital	Digital	Digital+DAC [20]	Symmetric
Analogue [22]	Analogue	Analogue	Analogue	Symmetric
MDAC [23]	Digital	Analogue (Ref.)	Analogue	Symmetric
This Work (Chapters 3&4)	Digital (Voltage)	Digital (Memductance)	Analogue (Current)	Asymmetric

multiplication operands and analogue product. The MDAC design has digital multiplier, analogue multiplicand and analogue product. The proposed work in Chapter. 3 and 4 have digital multiplication operands and analogue product.

The proposed designs have the both operands in digital form which removes the need for maintaining an analogue reference or other type of analogue input. These analogue signals will be costly in edge computing including IoT applications. Research in pure-digital input, pure-analogue output is, therefore, relevant for serving one of the important needs in the rapidly developing edge computing area.

This thesis presents a design approach for mixed-signal DI/AO multipliers. These multipliers are based on transistor-memristor cells located at the nodes of a crossbar for fast and efficient operation. With one of the operands (inputs) held in non-volatile memory, such a multiplier is suitable for use in applications for which one of the operands has a relatively stable value, for instance a reference input. Such a multiplier can be used as a replacement for or an improvement on an MDAC.

For AI applications such as NNs, on the other hand, the input and the output of a multiply-accumulate (MAC) unit should all be of the same format, e.g., digital, because the output of the multiplication usually is re-used as input for other MAC units in the NN [2, 3]. In order to extend the multipliers (presented in Chapters 3 and 4) for use in NNs, the analogue output needs to be converted to digital format. One strong reason for adapting these DI/AO multipliers is that they are based on the transistor-

memristor crossbar structure. In such multipliers, one of the operands is represented by memductance (memristor conductance), which is non-volatile. This is a good match for such applications as NNs and reference-based arithmetic, where one of the operands (e.g., the weight or the reference) tends to be relatively stable and requires only sporadic change [2–4,7]. Having that operand in non-volatile storage help reducing system energy consumption and operating latency.

1.3 Research Questions and Contributions

The energy-efficient multiplier design has set several fundamental requirements. However, implementations of multiplier demands more than the basics.

The arithmetic requirements from IoT nodes and NN cells on the edge pose the following research questions:

Research Question (RQ): Can a method be found for designing hardware multipliers that satisfy the following:

1. Both the operands (inputs) are digital and the product is analogue (DI/AO).
2. Operands and product are all digital, and have the same bit-width (DI/DO).
3. One of the operands is maintained in non-volatile memory (asymmetric non-volatility).
4. Low latency and low energy operations.
5. High precision and high bit resolution is not an important concern.

Note that for both of these application areas, high precision and high bit resolution (bit-width) are not a major concern [23–26], and lower precision can be traded for complexity, energy and speed gains.

This thesis seeks to answer these research questions and presents methods for making use of memristors to improve performance metrics including speed and energy efficiency of multipliers. The specific contributions of this thesis are as follows:

- A new, mixed-signal multiplier design method for multiplying two digital numbers and directly obtaining an analogue product without carry-chain and DAC complexities. (Addressing RQ. 1, 4 and 5)
- Comparative analysis of energy/performance against state of the art existing work, demonstrating the advantages of this work through extensive theoretical and experimental investigations. (Addressing RQ. 1, 3, and 4)
- Optimisation methods such as the elimination of current mirror (CM) by changing the topologies of memristor cells and investigating different memristor technologies resulting in an order of magnitude improvements in accuracy, speed and energy for lower complexity design when compared with the high complexity structure with CM. (Addressing RQ. 3 and 4)
- A high energy efficiency end-to-end multiplication accumulation unit based on the transistor-memristor crossbar multiplier with mode transition for such applications as classification NNs. (Addressing RQ. 3, 4 and 5)
- Validation of the MAC design using it as a perception in a non-Von Neumann NN implementation with quantization-aware training (QAT) solving a machine learning problem of non-trivial size (MNIST hand-writing classification). (Addressing RQ. 2, 3, 4, and 5)

1.4 Thesis Layout

This thesis is organised as follows:

Chapter 1 - Introduction. This chapter briefly presents the motivation for the thesis and summarises its contributions.

Chapter 2 - Background and Literature Review. This chapter gives background theory of the technologies used in the designs in this thesis. These include amplification implementations, the theoretical base of CMs, methods of high energy efficiency AI hardware design, as well as the properties of the memristor.

Chapter 3 - Transistor-memristor Crossbar Multiplier with Current Amplifiers. In this chapter, a design of crossbar array multiplier based on one transistor one memristor

(1T1M) is presented. The performance and characteristics are investigated. (Addressing RQ. 1, 3, and 4)

Chapter 4 - Memristive Multiplier Design with In-cell Current Multiplication. This chapter presents a multiplication cell which amplifies current in cell without CM circuit, and its use in multipliers. The performance and characteristics are investigated. (Addressing RQ. 1, 3, and 4)

Chapter 5 - Memristive Multiply-accumulate Unit Applied for Neural Network. The multiplier presented in the previous chapter is further developed into a MAC unit and an NN is constructed using such MAC units as perceptrons. The use of these types of NNs is investigated with real-world example machine learning applications. (Addressing RQ. 2, 3, 4, and 5)

Chapter 6 - Conclusions and Future Work. The contributions of this thesis are summarised, and future research areas for the development of memristor-based design solutions for computing performance in AI applications are suggested.

Chapter 2

Background and Literature Review

In this chapter, the technology baseline and related work are discussed, and appropriate literature survey is also carried out.

2.1 High Performance Arithmetic Circuit Design Techniques

2.1.1 Multiplication Circuit Design

Multipliers have been a computational building block or programming element in different computing and signal processing applications. These include filters, NNs, communication mixers, and communication modulators.

Multiplication is traditionally implemented through a sequence of logic AND, addition, subtraction, and shift operations. In other words, multiplication is a series of repeated additions [27]. The multiplicand is the number in addition, and the multiplier is the number of addition repetitions. Usually, multiplication is divided into several steps: partial product generation, partial product addition for two rows final addend and augend, and final product generation by adding row final addend and augend. Besides initial partial product generation procedure, each step of addition also generates a partial product. Carry propagation is along with the entire addition procedures [28]. The partial products addition procedure usually performed by digital adders. These circuits

generate delay and consume energy in the carry propagating procedure. Therefore, reducing the delay caused by carry propagation has been set as high-priority task about multiplication optimisation and widely investigated.

Partial products are conventionally generated by adders in various logic operations. For instance, in radix b notation, integer $x = (\dots x_2 x_1 x_0)_b$ and $y = (\dots y_2 y_1 y_0)_b$, sum with them will generate two integers, $p_{xy} = (\dots p_2 p_1 p_0)$ and $c_{xy} = (\dots c_2 c_1 c_0)$, and these two new integers has the relation as Eq. (2.1) [29] shows:

$$0 \leq s_i = x_i + y_i - b_{c_{i+1}} < b \; (i \geq 0) \tag{2.1}$$

In Eq. (2.1), c_{xy} are the "carry" digits with $c_0 = 0$ as the least significant bit (LSB) cannot get carry from a lower significant bit. As the speed of addition be affected by carry propagating time, a single sum usually is operated in a single adder with additional circuits running the carry propagating procedure. In a multiplier, the scale of the additional circuit, which contributes hugely to logic complexity, will increase with the bit-width and this increase not be proportional.

One way of reducing the carry propagation overhead is to reduce the number of addends and augends. In addition, Bedrij proposed a carry identification adder [30]. This design generates two sub-sums for each addition with repeating sub-addends and sub-augends addition twice in the same addition sequence. One is forced with carry digits in these two sub-sums, and the other is not. Therefore, the selection of addition results can be directly forwarded without heavy back-propagation [30]. Therefore, the multiplier can be much faster with this light carry propagation.

These conventional multipliers built with different adders show respective advantages for faster partial product generation. High-performance multiplier design needs to consider simplifying the number of addition operands, accelerating the generation of addition operands, and adding up all operators faster [31]. Wilkes tried to iterate the multiplication process for cutting down the number of addition operands [32]. This method is able to approximate the multiplication operands and shorten the digits for a quicker result than a full multiplication. As addition operands have the same amount as multiplier digits do, all addition operands need to be generated simultaneously. The efficient recording needs to be local operation with digit-shifted multiplicand. By doing

this, the number of addition operands will be halved. Normally, an addition operation is performed by a single adder that can only generate a single sum. Carry propagation grows when the number of bits increases, usually not in proportion. And the logic complexity also grows disproportionally with the increase of the number of bits. One way of mitigating this is to use carry-save schemes which reduce the horizontal passing of carry bits by delaying their resolution.

Carries must exist in digital multiplication because a single bit cannot represent a numerical value higher than 1, but adding two such bits produce a higher than 1. Furthermore, time and energy costs are inevitably caused by carry processing. Thus, digital no-carry/carry-mitigation/carry-optimisation schemes are ultimately incapable of completely removing the complexity of dealing with carries in digital multiplication. On the other hand, analogue arithmetic does not need to deal with carries because an analogue signal is able to represent a range of values large enough to contain all possible arithmetic results at that digit position. The current in each column may be amplified according to the column's bit significance. For instance, a current value stands for LSB can be amplified to the digit with respective significance. Simultaneously, KCL circuit adds up all currents and generates one current stands for the final calculation result. And this result naturally contains all carries. This will be discussed in detail in subsequent chapters (Chapter 3 and Chapter 4) as the multipliers presented in this thesis take advantage of this principle to eliminate carry processing.

2.2 High-speed Analogue-to-digital Converter

2.2.1 Flash Analogue-to-digital Converter

In Chapter 5 of this thesis, there is a requirement for high-performance low-energy analogue to digital (A2D) conversion. Small size is also an important requirement as the analogue-to-digital converter (ADC) represents a significant part of the hardware design.

A Flash ADC structure is presented in Fig. 2.1. With very high-speed architecture, flash ADC has its performance dominated by matching issues [33]. Also, the flash ADC

is a good fit for these requirements. Meanwhile, parallelism exists in both the current comparison and encoding operation by inputting the current into multiple comparisons while encoding multiple comparison results in one shot. This parallelism makes the flash ADC one of the fastest ADC schemes.

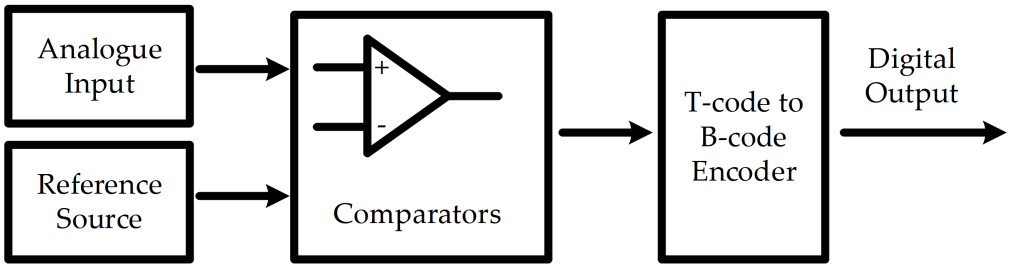
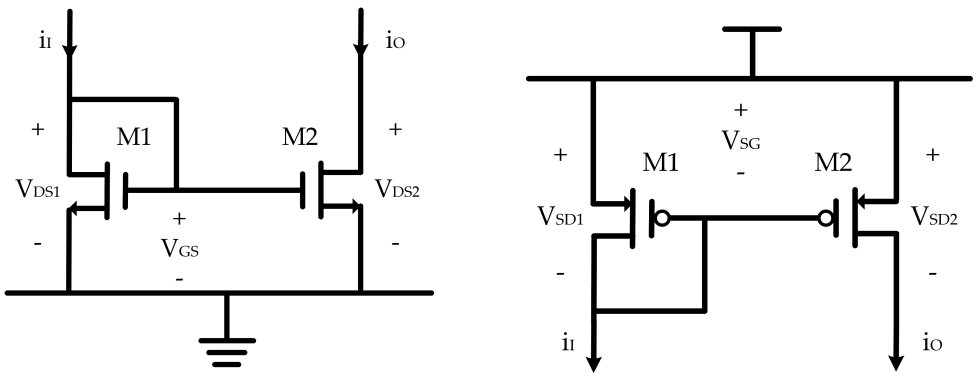


Figure 2.1: Block diagram of flash ADC.

2.2.2 Current Mirror

The CM was originally named after the equal channel current, which was generated by two identical MOS transistors with the same gate-source potentials [34]. Fig. 2.2 shows two CM structures, in them, (a) is n-channel CM structure, and (b) is p-channel CM structure.



(a) n-channel CM

(b) p-channel CM

Figure 2.2: Circuit diagram of CM.

In Fig. 2.2, i_I is defined by an input current source, and i_O is the output with the name "mirrored current". In the n-channel CM, transistor M1 has the drain connected with

the gate. Therefore, V_{DS1}=V_{GS} which means M1 is working in saturation. Similarly, M2 also needs to be set in saturation by V_{DS2}+V_{T2}≥V_{GS}. In this way, the ratio of i_O to i_I can be written as Eq. (2.2). In the following equations, i_I and i_O are the input current and output current, L_1 and L_2 are channel length of M_1 and M_2, W_1 and W_2 are channel width of M_1 and M_2, V_{GS} is the gate-source voltage on transistor, V_{T1} and V_{T2} are the threshold voltage of M_1 and M_2, V_{DS1} and V_{DS2} are drain-source voltage of M_1 and M_2, $K_1^{'}$ and $K_2^{'}$ are the process transconductance parameter of M_1 and M_2, λ is the device parameter of transistor.

$$\frac{i_O}{i_I} = \left(\frac{L_1 W_2}{W_1 L_2}\right)\left(\frac{V_{GS} - V_{T2}}{V_{GS} - V_{T1}}\right)^2 \left[\frac{1 + \lambda v_{DS2}}{1 + \lambda v_{DS1}}\left(\frac{K_2^{'}}{K_1^{'}}\right)\right] \tag{2.2}$$

Normally, the same physical parameters of MOS components in the same integrated circuit are identical. These include gate threshold voltage V_T and process transconductance K'. Therefore, Eq. (2.2) simplifies to Eq. (2.3).

$$\frac{i_O}{i_I} = \left(\frac{L_1 W_2}{W_1 L_2}\right)\left(\frac{1 + \lambda v_{DS2}}{1 + \lambda v_{DS1}}\right) \tag{2.3}$$

In other words, the value of i_O is proportional to the value of i_I, achieving pure unidirectional current amplification. In other words, putting a CM on an input current to generate a proportional output current does not modify the former.

Similarly, in the p-channel CM, transistor M1 has the drain connected with the gate. Therefore, $V_{SD1} = V_{SG}$ which means M1 is working in saturation. Similarly, M2 also needs to be set in saturation by $V_{SD2} + V_{T2} \geq V_{SG}$. In this way, the ratio of i_O to i_I can be written as Eq. (2.4).

$$\frac{i_O}{i_I} = \left(\frac{L_1 W_2}{W_1 L_2}\right)\left(\frac{V_{SG} - V_{T2}}{V_{SG} - V_{T1}}\right)^2 \left[\frac{1 + \lambda v_{SD2}}{1 + \lambda v_{SD1}}\left(\frac{K_2^{'}}{K_1^{'}}\right)\right] \tag{2.4}$$

It is reasonable to assume that the physical parameters, including the gate threshold voltage V_T, and process transconductance K', are the same for the same p-type transistor. Then Eq. (2.4) simplifies to Eq. (2.5).

$$\frac{i_O}{i_I} = \left(\frac{L_1 W_2}{W_1 L_2}\right)\left(\frac{1 + \lambda v_{SD2}}{1 + \lambda v_{SD1}}\right) \tag{2.5}$$

If $V_{DS1} = V_{DS2}$ ($V_{SD1} = V_{SD2}$), then the ratio of i_O/i_I becomes Eq. (2.6).

$$\frac{i_O}{i_I} = \left(\frac{L_1 W_2}{W_1 L_2}\right) \tag{2.6}$$

To increase the current, the size ratio of the CM transistor can be adjusted [34]. According to Eq. (2.6), a much larger multiplying transistor (M2) with W/L ratio greater than that of the reference transistor (M1) can be used in the CM to amplify current. This technique can also be applied to a p-type CM for current amplification.

CMs and other purely transistor-based methods are not the only ways in which current can be tuned. In low-energy and high-speed applications, the latency and energy consumption of charging and discharging transistors are need to be avoided. In some of the multipliers presented in this thesis CMs are not used for this reason.

2.2.3 Complementary Metal–Oxide–Semiconductor Current Comparator

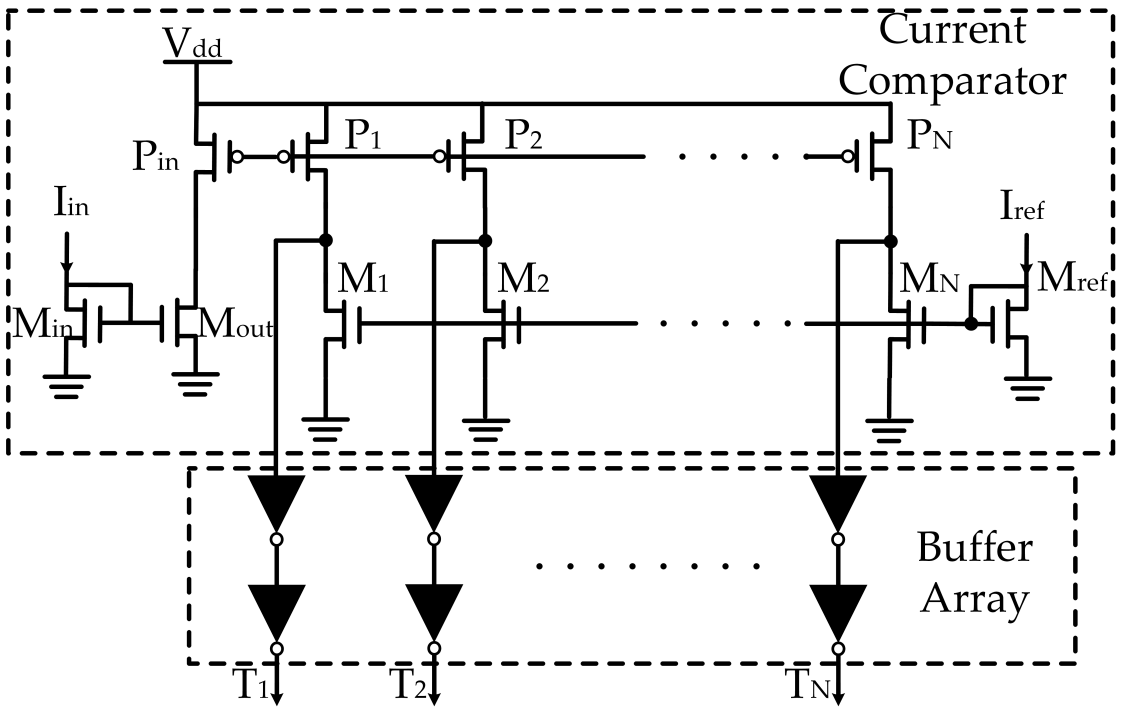


Figure 2.3: Circuit diagram of a current comparator thermometer code generator [35].

In ADC designs, comparing an analogue input signal with threshold features is a

prominent operation. When the input signals are in the form of voltages, this comparison requires a respective circuit structure to build a feedback path to sources such as source-coupled pairs and complementary device pairs of common-gate and common-drain. In comparison, if the input analogue signals are in the form of currents, the comparison can be implemented in a much more straightforward manner, resulting in faster responses in some cases [36].

The current comparator incorporates CM circuits, which are shown in Fig. 2.3. The input is on the left, and the reference is on the right. In this design, p-MOSFETs (P_1 to P_N) function as current sources, while n-MOSFETs (M_1 to M_N) act as current sinks. The p-MOSFET source current is mirrored from the input, while the n-MOSFET sink current is mirrored from the reference source. As a result, the voltage at the junction point between the p-MOSFET source and the n-MOSFET sink increases to Vdd when the p-MOSFET source current is greater than the n-MOSFET sink current. Conversely, if the n-MOSFET sink current is greater than the p-MOSFET source current, the junction point voltage drops to ground. The comparison of currents is therefore represented in voltages.

To detect a small reference current, the sink can be constructed with multiple same channel length n-MOSFETs connected in series. When the differences between the input and reference currents are minimal, the output may not be resolved to logic levels. To address this, dual series-connected inverters in the buffer array amplify the comparator output to standard logic levels. With this setup, the gain inverter array produces a thermometer code where the boundary between 0 and 1 indicates the input current value.

2.2.4 Thermometer Code to Binary Encoder

After the current comparator generates its output in a thermometer code, the encoder needs to translate the thermometer code to binary code for output. The thermometer code to binary encoding consists of 2 procedures. First, it generates a one-hot code from the thermometer code. Second, it converts one-hot code to binary code. A 16-bit thermometer code to 4-bit binary code encoder is illustrated in Fig. 2.4. As can be seen, the one-hot code is generated by the AND gates. Then, the one-hot code is input

into a binary encoded ROM pattern. Finally, the binary output is buffered and sent out.

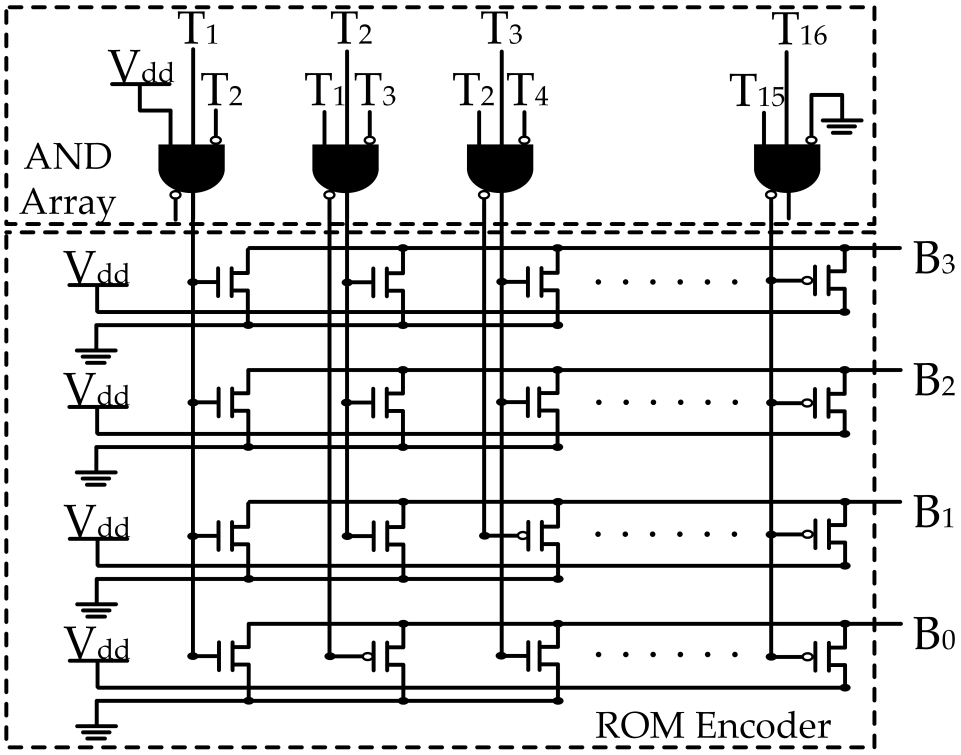


Figure 2.4: Circuit diagram of ROM thermometer to binary encoder [35] [37]

During this procedure, the output of binary encoder generation needs to scan from the LSB to the most significant bit (MSB). LSB will keep swinging until all significant bits are encoded until the higher significant bits are set.

2.3 Memristors

In 1971, Leon Chua related the fundamental circuit variables charge (q) and flux linkage (ϕ) with a mathematical description of a component. Because this relationship includes non-volatility in the adjustable resistance state, it was called ”Memristor”, short for ”memory resistor” [19, 38, 39]. The memristor was proposed as the fourth element in the charge and flux taxonomy [19, 39] and had a number of promising

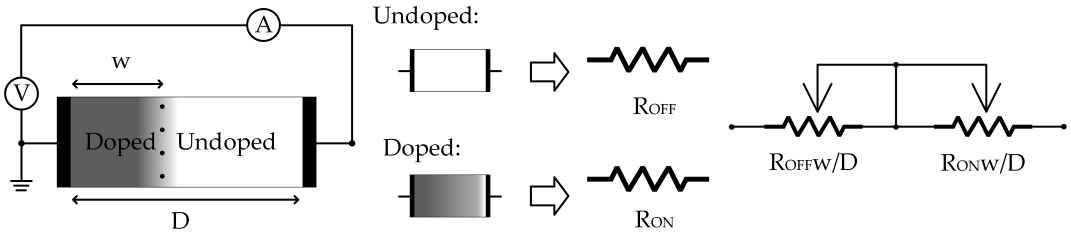


Figure 2.5: Details of memristor internal state [38].

characteristics. One of these is its potential in replacing semiconductor components in processing circuits. That is because, as a switchable device, a memristor can perform similar ON-OFF operations to a transistor, and this became more significant when practical memristor implementations appeared [38, 40]. The RS devices which come from emerging memory technologies are also known as resistive random access memory (RRAM) [41]. A memristor is an RRAM device, typically based on a metal-insulator-metal (MIM) structure. The proper voltage to the top electrode will generate conductive filament (CF) between the top and bottom electrodes. Thus the high density of CF makes the device in low resistance state (LRS). Figure. 2.5 shows in detail how memristor resistance state relates to doped region width (w) and device length (D). If the electric potential on the left side terminal of the doped region is higher than that on the right side terminal of the undoped region and over a threshold value, the doped region width will increase, and the memristor resistance will decrease, and vice versa [38,39].

Conversely, the rupture of CF by application of proper voltage to the bottom electrode will make the device in high resistance state (HRS) [42,43]. This kind of processing, called IMC, is a design for computing within the memory, thus eliminating the energy-intensive and time-consuming data movement. In this thesis, the design strategy applies the best memristor component with a transistor as individual functional cells.

As a nonvolatile component, memristor has been used in memory device design, which is now called "resistive memory" [44]. At the same time, the possibilities for performing arithmetic with memristors have also been explored, with multiplication being viewed as especially promising [45].

The multiplier solutions presented in this thesis are centred around the use of

memristors in novel ways. Fundamentally, the methods presented by using any resistive non-volatile memory. Memristors are chosen for this work because of their support for integration into normal CMOS circuits, the existence of memristor devices with suitable properties and the availability of reliable and trustworthy models for investigating the performance of implemented hardware.

2.3.1 Memristor Physical Models

Titanium-dioxide Thin-film Memristor

Inspired by Chua's theoretical work, HP lab presented the first Titanium Dioxide (TiO_2) thin-film memristor device. Strukov and colleagues built a physical model of a two-terminal electrical device that behaves like a perfect memristor [38]. In detail, the device state variable w specifies the distribution of dopants in the device. It is bounded between zero and D (maximum device length). R is the general resistance that depends on the device's internal state, which has the highest value R_{OFF} and the lowest value R_{ON}. The external bias $v(t)$ across the device will move the boundary between the high-dopant region and low-dopant region by causing the drifting of charged dopants and generate respective current $i(t)$. With average ion mobility μ_v, the simplest case of Ohmic electronic conduction and linear ionic drift in a uniform field can give us the following relations.

$$v(t) = \left(R_{ON} \frac{w(t)}{D} + R_{OFF} \left(1 - \frac{w(t)}{D} \right) \right) i(t) \tag{2.7}$$

$$\frac{dw(t)}{dt} = \mu_V \frac{R_{ON}}{D} i(t) \tag{2.8}$$

From Eq. (2.8), the formula for $w(t)$ is generated as:

$$w(t) = \mu_V \frac{R_{ON}}{D} q(t) \tag{2.9}$$

Then the memristance of this system can be derived by inserting Eq. (2.9) into Eq. (2.7) with simplification from $R_{ON} << R_{OFF}$,

$$M(q) = R_{OFF}\left(1 - \frac{\mu_V R_{ON}}{D^2}\right) q(t) \tag{2.10}$$

TiO_x devices have similar current-voltage (I-V) relations; meanwhile, I-V characteristic from a metal/oxide/metal cross-point device. This device applies the nanometer scale thick oxide film, which initially contained one layer of insulating TiO_2 and one layer of oxygen-poor TiO_{2-x}. This structure generates the boundary condition on the state variable of the device. A detailed model of TiO_2 memristor is presented in Fig. 2.6.

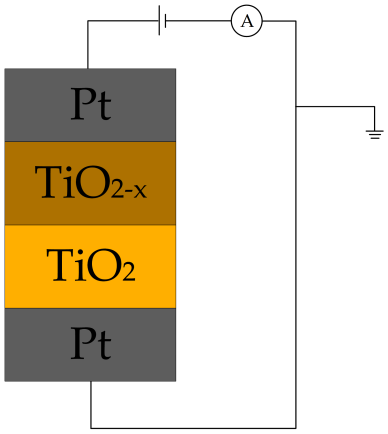


Figure 2.6: TiO_2 memristor device architecture [38].

As can be seen, the oxygen vacancies are drifting in the applied electric field as mobile +2-charged dopants. Also, they shift the actual boundary between TiO_2 and TiO_{2-x} layers. This shifting performs the switching characteristic on the state variable of device. Meanwhile, this model's ON/OFF memristance ratio ranged from 160 to 380. As will be discussed in later chapters, this type of digital-in/analogue-out multiplier does not represent Boolean 0 in the operands with true 0 values of physical parameters – the high resistive state (HRS) of a memristor cannot have a conductance of true 0 and the low resistive state (LHS) of a memristor cannot have a conductance of infinity. This means that $I_{i,j}$ cannot be 0 amps even when it represents a Boolean value of 0. Consequently, when multiple Boolean 0's are added together to produce an overall product P of 0, the actual value of I_{out} representing $P = 0$ is not 0 amps.

The maximal precision of such a multiplier is therefore limited by the ratio between

R_{MH} and R_{ML}, which is technology-dependent. This is because the value of I_{out} that represents $P = 0$ must be lower than the value of I_{out} that represents $P = 1$. Conservatively, this is true if I_{out} representing $P = 0$ is lower than the current $I_{i,j}$ representing a single bit value of 1. In other words, if the following inequality is true, the multiplier precision is not violated at a specific word length.

$$R_{MH} > P_{maxN} \times R_{ML}, \tag{2.11}$$

where P_{maxN} is the maximal value of the product for an $N \times N$-bit multiplier. For instance, for a four-bit multiplier $P_{max4} = 225$ and for a five-bit multiplier $P_{max5} = 969$. Thus, this can be usable in our low-precision multipliers.

However, memristor still suffers from low endurance (10^5 cycles, the satisfactory switch endurance should be larger than 10^6 cycles), high write energy (2 nJ, reported satisfactory operational energy is 0.375 pJ), and high latency (100 ns, the required fast switch speed is 5 ns). This sets a low bar for performance which memristors based on other materials have been shown to improve on [43, 46, 47].

Copper Doped: Zinc Oxide (Cu:ZnO) Thin-film Memristor

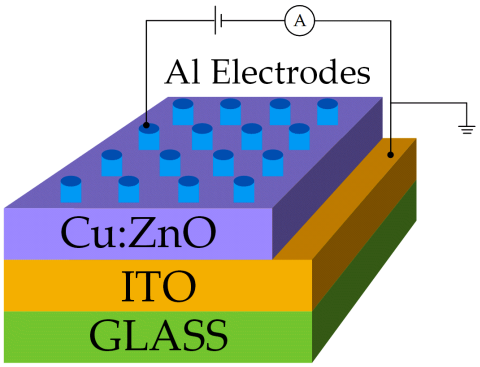


Figure 2.7: Schematic of the fabricated Cu:ZnO memristor architecture [46].

Cu:ZnO is an emerging material that possesses both ferroelectricity and oxygen vacancies, the key factors for realising meaningful memristors [46]. Suresh and colleagues presented their work in [48], where the fabricated, Cu:ZnO based, Set/Reset

devices exhibited low S/R voltages (+1.40/-1.2 V), high ON/OFF ratio (2×10^3), and high retention (up to 10^6 s period without degradation). The RS device based on this ferroelectric Cu:ZnO offers better performance when compared to the former lower temperature annealed Cu:ZnO devices. This character of Cu:ZnO memristor provides good temperature variation tolerance.

As can be seen, the schematic of Cu:ZnO memristor illustrated in Fig. 2.7 shows that aluminium is used as the top electrode while indium tin oxide (ITO) acts as the bottom electrode. A thin film of Cu:ZnO on ITO/glass substrate provides the characters of a RS. In detail, Zn^x, O_o^x, and $V_O(oxygen\ vacancies)$ are considered as internal defects during the formation of oxygen vacancies and ions in the ZnO lattices. The Zn_i and V_O defects in ZnO make ZnO show grown n-type behaviour. If a proper amount of Cu dopants is incorporated in the ZnO lattices to form an $Cu_{Zn} - V_O$ acceptor complex, Cu:ZnO will show p-type conductivity. The internal details of the respective RS states of Cu:ZnO memristor are presented in Fig. 2.8. Under different bias, O^{2-} and V_O^{2+} ions move in their respective direction in the device, which depends on the polarity of the applied voltage. This is the reason for switching between HRS and LRS.

The Cu:ZnO-based memristor technology is also suitable for the multipliers described in this thesis and will be compared with TiO_2-based devices. The earlier TiO_2 memristor device displayed clear and consistent memristive behaviour and stable logic TiO_2 memristor device performance [38, 49]. Its limited ON/OFF ratio fails to offer better performance for memristance variation tolerance in large-scale algorithm applications. Hence, we investigated the effects of memristor resistance variability. To this end, we selected the Cu:ZnO memristor device [50], which features a larger terminal resistance of more than 1000 and operates in a voltage range similar to our previous TiO_2 memristor device.

The Cu:ZnO device we chose exhibits a device-to-device (DD) variability of 59% for the high-resistance state (HRS) and 36% for the low-resistance state (LRS), while the cycle-to-cycle (CC) variability is 89% for the HRS and 51% for the LRS. Note that, although the CC variability is particularly high, it is impossible for RML to exceed RMH given that the baseline ratio between these two parameters is 1000 for the Cu:ZnO technology.

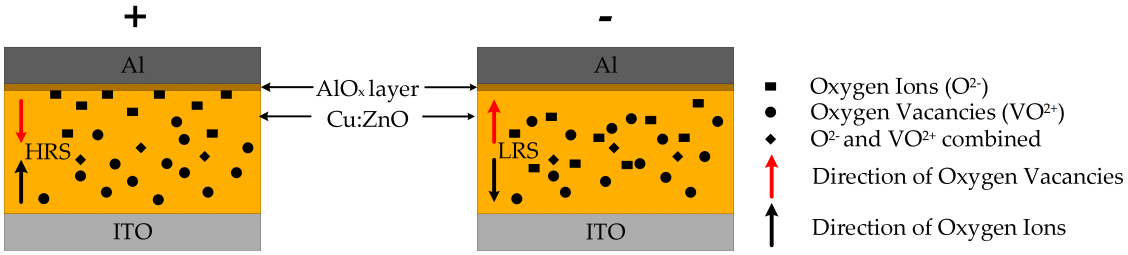


Figure 2.8: Schematic of the transport mechanism of the forming free Al/Cu:ZnO/ITO/glass device [46].

2.3.2 The Voltage ThrEshold Adaptive Memristor Model

In addition to the general mathematical model, analogue behavioural models (ABM) are developed for deeper research on memristor characteristics in circuits. The linear ion drift model has first developed from the basic memristive definition of memristor I-V relationship. This model uses the current-control method to adjust doped region width for changing memristor resistance [38]. The ideal assumption that the doped region width changes linearly is unrealistic and especially undesirable for logic circuits. With the assistance of window function, the relation between physical device size and resistance variation is also regulated. As a result, the nonlinear ion drift model was developed to present the complexity of fabricated memristive device state drift [51].

As early-stage models, both the linear ion drift and the nonlinear ion drift models offer low accuracy for modelling the oxide region and doped oxide region like two serially connected resistors. Aiming at building a more realistic model, a more accurate physical model is built by serially connecting an electron tunnel barrier with a resistor. This one is called the Simmons tunnel barrier model, which shows a higher level of accuracy among TiO_2 memristive devices without increasing model complexity [53, 54]. For balancing accuracy and complexity of the model, Kavatinsky simplifies physical behaviour and mathematical functions complexity in the Simmons tunnel barrier model, then the threshold adaptive memristor model (TEAM) is generated with a reasonable balance between accuracy and computational efficiency [55]. Since the existence of

Table 2.1: Voltage ThrEshold Adaptive Memristor Model Parameters

Parameter	Memristor	
	TiO_2 [52]	Cu:ZnO [48]
α_{OFF}	4	7
α_{ON}	4	5
V_{OFF} (V)	0.3	0.9
V_{ON} (V)	-1.5	-0.85
$R_{OFF}(\Omega)$	300k	152M
$R_{ON}(\Omega)$	1k	150k
k_{OFF} (m/s)	0.091	40
k_{ON} (m/s)	-216.2	-80
w_{OFF} (nm)	3	3
w_{ON} (nm)	0	0

threshold voltage is found from memristive devices, Kavtinsky updated ABM TEAM to voltage threshold adaptive memristor (VTEAM) [56, 57]. As a threshold-based voltage-driven model, VTEAM combines the advantage of the TEAM model with multiple freely chosen I-V characteristics that precisely estimates all reported physical device behaviours, such as linear ion drift [38], nonlinear ion drift [51] and the Simmons tunnel barrier [53]. At the same time, it exhibits superior computation efficiency especially for memory and logic applications [56–58].

This thesis utilises the VTEAM memristor model for design and analysis purposes, with the relevant parameters listed in Table. 2.1. Notably, k_{OFF}, k_{ON}, α_{OFF}, and α_{ON} are constants, while R_{OFF} and R_{ON} represent the terminal switching state resistances, and w_{OFF} and w_{ON} denote the undoped region length. Additionally, V_{OFF} and V_{ON} refer to the threshold voltages. A careful examination of these parameters reveals that both the TiO_2 and Cu:ZnO memristors possess a 1.8V region between two threshold voltages. The Cu:ZnO memristor exhibits a more balanced working zone that enables dual direction bias setting for multiplication cell writing operation. Furthermore, in the case of multiple

memristor design, the 500× larger terminal switching state resistance (under worst-case variation of 50% $R_{OFF(Cu:ZnO)}/R_{OFF(TiO_2)} \times \frac{1}{2}$)ensure that the memristor dominates the multiplication cell output current through the memristor resistance drop.

2.3.3 Transistor Memristor Cell Design

In functional circuit design, a major challenge memristors face is in array fabrication because of its requirement for high-quality metal thin film, which has high risks on current leakage between different functional units [59]. This requirement motivates mixing CMOS with memristor to mitigate the leakage issue [60]. Various types of transistor memristor combinations have been explored, such as one transistor two memristors (1T2M) [61], three transistors two memristors (3T2M) [62], eight transistors two memristors (8T2M) [63], etc. However, cell power efficiency still has room for improvement.

Memristor cell methods have already featured in complex logic calculations such as "material implication" (IMP) [64, 65] and majority inverter graph (MIG) [66]. These existing cases motivate the multiplication cell design presented in this thesis.

2.4 Memristor-based Multiplier Design for Energy Efficiency

Since computation and storage are physically separated in the predominant processing hardware architectures, the data traffic in a typical computing procedure cycle will start at importing data from the memory unit. Data will be transmitted to the processing unit (where computation takes place). Once the multiplication is completed, the data is sent back to the memory unit for storage. Fig 2.9(a) illustrates this conventional multiplication process. This data transfer between the processing unit and the memory unit can result in a fundamental bottleneck in computer performance, commonly referred to as the memory wall [67, 68]. One potential solution to overcome this bottleneck is to combine data loading and storage in the same block, as in the in-memory multiplication method depicted in Fig 2.9(b). The memristor's unique properties, including non-volatility and scalability, make memristor a promising candidate as the target memory component. Electronic NNs based on RS memory array or memristor have also been proposed

in [18,19,69].

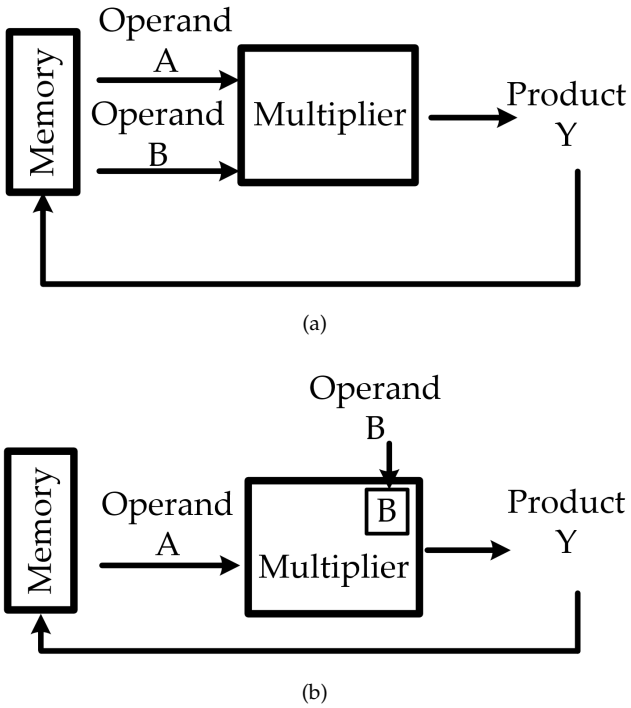


Figure 2.9: Multiplication type. In (a), typical type. In (b), in-memory type.

In [18], the crossbar array of hafnium oxide (HfO_2) memristor is used as a reconfigurable analogue processor for edge computing. Transistors are the most mature option for precisely programming individual memristor with stronger sneak-current tolerance in access devices [70]. A vector of voltage outputs from a sensor can be applied directly to the rows of a memristor crossbar. The conductance of the crossbar multiplication cells stores the values of the appropriate matrix elements. The currents that appear on the array columns in real-time represent the output vector of the multiplication. To read out the results in parallel, a trans-impedance amplifier (TIA) will convert the current signal from each column to a voltage signal. Moreover, the 1T1M crossbar with linear I-V memristors enables accurate analogue vector-matrix multiplication (VMM). During memristor programming, the gate voltage applied to the transistor is controlled to generate the respective compliance current. For each of the TiO_2 and Cu:ZnO technologies, the observed range between the two worst cases is then compared with

the specified (ideal) range from the memristor models and checked for compliance with $\frac{High\ Memoristor\ Conductance}{225} > (Low\ Memoristor\ Conductance)$. After programming, during inference, all transistors are turned ON to perform a one-step VMM [70]. It yields a good approximation to the scalar product of a vector component and matrix element. These generate adequate accuracy and high speed-energy efficiency for IoT and edge network (i.e., signal spectrum analysis, image compression, and convolutional filtering). Simultaneously, a crossbar multiplier is potentially an area-saving solution because the memristor crossbar can be built on top of the transistor-related layers using a back-end-of-line process [71]. Therefore, the area can be smaller than the traditional CMOS multiplier used.

2.5 Current-mode Arithmetic

Current-mode arithmetic circuits have shown their promising characteristics in improving energy efficiency [72]. In this mode, currents of varying amplitude in different circuit paths are driven by analogue bias voltages. Due to Ohmic elasticity of current paths this mode shows noticeable improvement in energy proportionality than the traditional voltage-mode digital circuits. Additionally, current-mode design generates high output charging speed per unit of time (slew rate) and simpler structure for arithmetic operations. For instance, directing a current path into a node or carrying a current path away from a node is equivalent to addition or subtraction. Moreover, adjusting the resistance of current generating cell enables low-complexity amplification, which is analogous to current multiplication or division. As such, when a network of current paths is generated it can be operated faster with lower energy consumption at significantly reduced circuit complexity [18, 19, 73].

2.6 Multiplying Digital-to-analogue Converters

An MDAC is a device which multiplies a digital (usually binary) numberD_b with an analogue signal s to generate an analogue product P, such that $P = s \times D_b$ [23]. It is most likely used to multiply a stream of variable digital numbers (input signal) to a

relatively constant analogue reference, or to multiply a constant digital number with a varying analogue input signal. In other words, the relatively stable or constant operand usually serves as a coefficient which is multiplied to a variable, i.e., $P = s \times D_b(t)$ or $P = D_b \times s(t)$ - a type of operation quite often found in signal processing and control applications, which feature prominently in IoT edge nodes. It also see applications in hardware neuromorphic computing serving as a synaptic node with the more stable operand as the weight and the varying operand as the input [23]. Requiring one of the operands to be analogue, which means that it has limited use in cases where both operands are the result of digital computation. And maintaining an analogue reference also require an energy overhead which could be objectionable for edge computing. The method can be applied to any resistive memory (RRAM) technology beyond memristors, so long as the crucial Ohm's law and KCL combination holds at cell and crossbar levels. With better resistive memory technologies and paying with more design effort and operating energy, it may be possible to scale the precision or resolution of multiplication up, but given the exponential nature of the 1TxM cell design, the method's significance for high-precision low-approximation arithmetic is limited.

2.7 Neural Networks

The NNs method predominates the existing AI systems. Modern NNs have developed into high complexity levels across different application domains compared with Rosenblatt's first neural automaton in 1957 [74]. Basically, NNs generate the weighted sum of all inputs in the training phase in multiple layers. Using activation functions, calculating weighted sums, and generating/adjusting the weights lead to heavy requirements of arithmetic circuits (i.e., MAC units) for modelling electronic neurons in hardware implementations. [75]. More inputs and added complexity of the problem will inevitably lead to a rapid increase of the number of MAC units in an NN [76]. Therefore, reducing the complexity of each MAC unit and improving its energy efficiency and speed are central design motivations.

In traditional computer architecture, data is typically stored in a separate memory unit and then transmitted to the processing unit for computation. Once the computation

is completed, the data is sent back to the memory unit to be stored. This process is often limited by the speed of data transfer between the processing unit and memory unit, a phenomenon commonly referred to as the ”memory wall” [67].

In-memory computing is a promising solution to the problem of the memory wall, where computation is performed directly on the memory. This approach reduces energy consumption and the time required for data movement, as the processor generates commands for calculations on the memory itself. By eliminating the need to transfer data between separate memory and processing units, in-memory computing can greatly improve computing performance. Concurrently, progress in memory architecture utilising resistive switching devices has facilitated the advancement of in-memory computing through their characteristic resistive switching properties. The ability to perform direct data processing within the memory module not only enhances energy efficiency but also reduces the required area for computation [67].

In Chapter 5, we present the design of a low-energy and low-latency MAC unit. This unit can be utilized as a standardized component for the construction of energy-efficient neural network implementations.

2.8 Summary

In this chapter, we have discussed the technology baseline and related work. And we also carried out appropriate literature survey.

Section 2.1 generally clarifies the core design requirement of high multiplication circuit. Besides the arithmetic circuit design, signal conversion circuit design is also a high significant part for latency shrinking. Respective high speed scheme has been reviewed in Section 2.2. Simultaneously, the core component used in the proposed multiplier design is introduced in Section 2.3. Besides the component, architecture applied in the proposed designs is also reviewed in Section 2.4. Meanwhile, several architectures applied for comparison with proposed work are reviewed in Section 2.5 and Section 2.6. Finally, target implementing application is reviewed in Section 2.7.

Chapter 3

Transistor-memristor Crossbar Multiplier with Current Amplifiers

AI and signal processing applications constitute the major driver of the IoT [12]. The dominant processing arithmetic used in these applications is multiplication. Additionally, in edge computing node applications, the results of these arithmetic operations must be presented in analogue form. However, the high stand-by latency and high power consumption caused by the complex logic chains and the additional carry propagation circuit used in conventional multipliers are major hindrances to their overall energy efficiency, particularly in the high density computing tasks of IoT and edge AI applications [12,77]. DACs are also associated with high overheads.

Over the years, researchers have investigated methods to reduce the energy cost and latency of multiplication operations. These methods have relied on adjustment of the carry chain length using ethier approximate [14, 78] or speculative circuits [79, 80] in CMOS logic based designs. However, the circuit's precision, latency, and power consumption are still limited by the lengths of the carry chains. These designs use voltage-mode logic boundaries that are defined by Landauer's limits by setting a set of fixed over-threshold voltage/frequency pairs. For low-power operation, the correct voltage/frequency pair is selected based on a combination of the carry propagation

length and the performance requirements [17].

As a result, the design of multiplying circuits with reduced energy and increased speed remains an ongoing challenge. This chapter presents a carry-free multiplier design using resistive elements that takes input digital signals and produces an analogue output in the form of a current signal designated I_{out}. This multiplier circuit consists of an array of memristor-transistor cells that can be selected (i.e., turned ON or OFF) using a combination of DC bias voltages based on the operand values (See Fig. 3.1). When a cell is selected, it contributes to the current in the array path, which is then amplified by CMs with various transistor gate sizes. The different current paths are connected to a node to accumulate the currents required to produce the multiplier output directly. This approach removes the requirement to have the latency-sensitive carry propagation stages that are typically seen in traditional multipliers. One essential feature of this multiplier is its autonomous survivability, i.e., when the power falls below the normal functional threshold, one of the operands retains its value at zero cost because of the nonvolatile properties of the memristors.

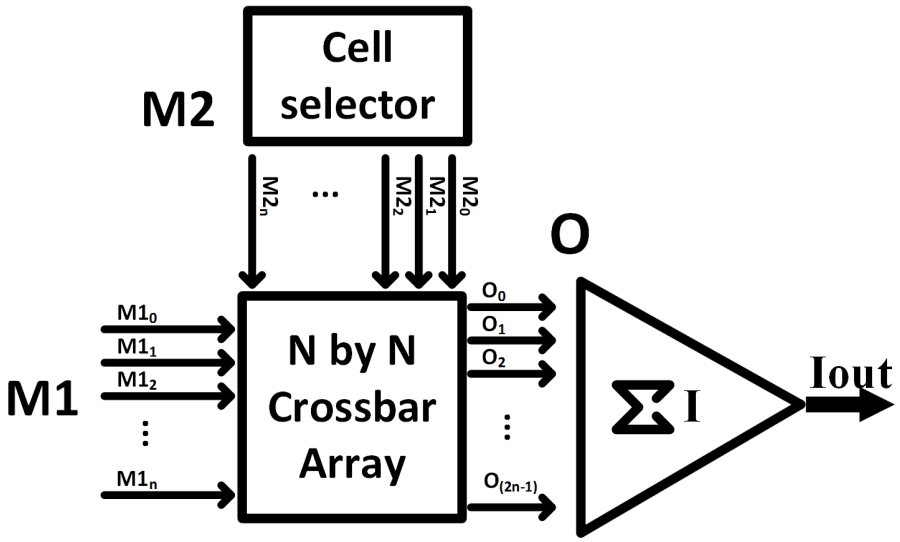


Figure 3.1: Block diagram of the proposed mixed-signal carry-free current-mode multiplier. This diagram shows the connections between the different blocks of the proposed multiplier. The n-bit signals M and N generate a (2n-1)-bit result via multiplication; here, $M1_0$ means the first bit of M1, $M1_1$ means the second bit of M1, and the regulation also fits M2 and O.

3.1 Single Transistor Single Memristor Multiplier

3.1.1 Algorithm for the Crossbar Multiplier

In a traditional ($N \times N$) binary multiplier, two unsigned integers can be multiplied using N^2 logic AND operations followed by up to $2N$ ADD operations. As an example, consider the multiplication of two 4-bit unsigned integers, where the multiplier is $M_1 : \{m_3 m_2 m_1 m_0\}$ and the multiplicand is $M_2 : \{n_3 n_2 n_1 n_0\}$. The multiplication of these two numbers is implemented using the long multiplication algorithm shown in Table 3.1.

Table 3.1: Binary Multiplication Algorithm with 4-bit Operands

				m_3	m_2	m_1	m_0		
			$\times$	n_3	n_2	n_1	n_0		
	0	0	0	m_3n_0	m_2n_0	m_1n_0	m_0n_0	$\leftarrow$	PP
	0	0	m_3n_1	m_2n_1	m_1n_1	m_0n_1	0	$\leftarrow$	PP
	0	m_3n_2	m_2n_2	m_1n_2	m_0n_2	0	0	$\leftarrow$	PP
	m_3n_3	m_2n_3	m_1n_3	m_0n_3	0	0	0	$\leftarrow$	PP
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	$\leftarrow$	FP

As shown, the N^2 logic AND operations produce partial product (PP) terms (i.e., the $m_i n_j$ bits), which can be generated rapidly in parallel. These terms are then added column-wise, with the columns having different numbers of PP terms. For the example

given here, the column-wise sums of the product terms can be expressed as follows:

$$\begin{aligned} P_0 &= m_0 n_0; \\ P_1 &= m_1 n_0 + m_0 n_1; \\ P_2 &= m_2 n_0 + m_1 n_1 + m_0 n_2; \\ P_3 &= m_3 n_0 + m_2 n_1 + m_1 n_2 + m_0 n_3; \\ P_4 &= m_3 n_1 + m_2 n_2 + m_1 n_3; \\ P_5 &= m_3 n_2 + m_2 n_3; \\ P_6 &= m_3 n_3. \end{aligned} \tag{3.1}$$

Equation (3.1) shows that when the number of PP bits in a column is two or more, carry propagation then becomes more likely, depending on the operand bit values. For example, if $m_1 = m_0 = n_1 = n_0 = 1$, P_1 is then expected to produce a carry into P_2. When both operands have all their bits set to 1, i.e., M_1=$\{1111\}$ and M_2=$\{1111\}$, the multiplier then experiences the largest carry propagation chain between the columns, starting from the LSB to the MSB in the multiplier output.

In traditional multipliers, the maximum delay between the longest PP addition (i.e., P_3 in the example shown in Table 3.1) and the carry propagation between the column-wise additions determine the critical path (i.e., the latency) and the energy consumption of the circuit. The latency can be reduced by applying a number of techniques including various carry save schemes with the last row of additon implemented via carry look-ahead (CLA) methods [81] or approximate equivalent methods [82]. However, full Boolean digital addition cannot avoid the carry processing and its associated overheads.

In a mixed-signal circuit that uses currents to encode the PP values, addition operations can be implemented by converging the current paths into a single node. When the length of the chain of add operands increases, more paths can be added or enabled without causing any significant changes to the circuit delay. This provides the key motivation to design a mixed-signal multiplier circuit using our proposed approach, which will be described next.

In our proposed multiplier, the column-wise terms (shown in Table 3.1) are expressed as non-Boolean values and programmed as current paths. In practice, this means that the current in a single wire can represent a wide range of values, with these values certainly

going beyond 0 and 1. When Eq. (3.1) is updated using this assumption, the values of the column-wise terms P_i, $i = [0, 6]$, can be expressed as follows:

$$\begin{aligned}
P_0 &= 2^0 \times (m_0 n_0)\,; \\
P_1 &= 2^1 \times (m_1 n_0 + m_0 n_1)\,; \\
P_2 &= 2^2 \times (m_2 n_0 + m_1 n_1 + m_0 n_2)\,; \\
P_3 &= 2^3 \times (m_3 n_0 + m_2 n_1 + m_1 n_2 + m_0 n_3)\,; \\
P_4 &= 2^4 \times (m_3 n_1 + m_2 n_2 + m_1 n_3)\,; \\
P_5 &= 2^5 \times (m_3 n_2 + m_2 n_3)\,; \\
P_6 &= 2^6 \times (m_3 n_3)\,.
\end{aligned} \tag{3.2}$$

Note that without the requirement for carry operations, there is no need for a signal P_7, and each individual P_i, where $i \in [0, 6]$, is not Boolean. The sum of all column-wise terms in Eq. (3.2) will then produce the multiplier output as follows:

$$M_1 \times M_2 = \sum P_c \quad (c = 0, 2 \ldots 2N - 2)\,, \tag{3.3}$$

where P_c is the sum of the products on the c_{th} column.

Because the summation of a number of currents does not need to be performed in Boolean space, the resulting current can be used to encode numbers much greater than 1. This eliminates the need to carry to the left.

3.1.2 Crossbar Multiplier Architecture

The multiplication algorithm given in Eqs. (3.2) and (3.3) can thus be simplified into three steps: 1) PP terms can be generated in parallel by switching the current paths ON or OFF; 2) each current path, as defined by using the column-wise terms in Eq. (3.2), is amplified in current mode, according to its position index i with the amplification coefficient 2^i; and 3) the final output shown in Eq. (3.3) can be generated by summing the currents from all paths. In the following, we provide a briefly outline of the design approach for these three steps.

The PP terms are generated by switching the current paths using the memristor-transistor cells, which are organised in a crossbar array as shown in Fig.3.2. The low-level circuit layout of the cell is shown in further detail in the pullout in Fig. 3.2.

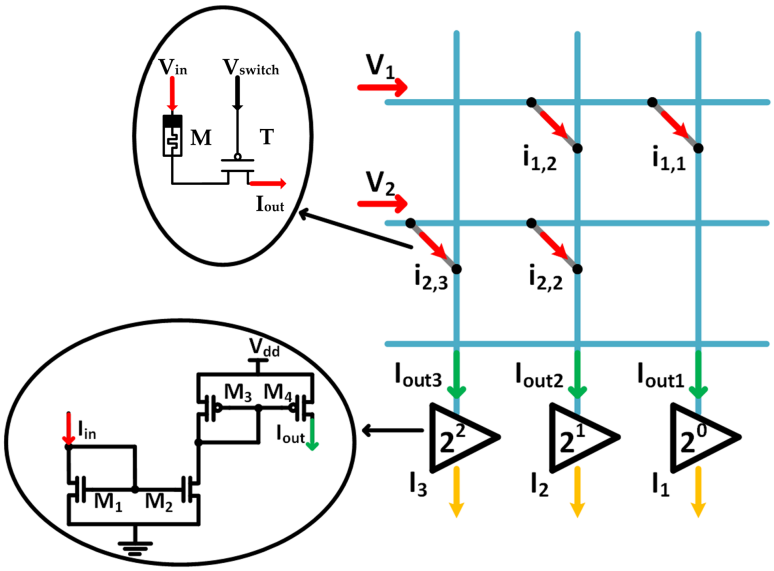


Figure 3.2: Multiplier product generation and accumulation circuits.

In the crossbar architecture, the row lines (RLs) and the column lines (CLs) are connected at the cross point through this cell. This arrangement allows the current paths to be switched ON or OFF based on the multiplier bit-wise operand values. One of the operands used is the combination of control signal V_{switch}, which switches all cell transistors on the same column, and the input signal V_{in}, which powers all cell memristors on the same row. Concurrent switching of the cells using V_{switch} and V_{in} produces a bit-wise AND-like operation at each corresponding cell for target current path conduction. The other operand is represented by the cell's passive memductance (i.e., the memristor conductance) G with the input voltage V_{in} and is used to generate the PP terms (current) in the multiplier.

In a current-mode switching arrangement, the current paths that define the PP terms are generated according to Ohm's law. Using this law, the currents in each pathway, which are denoted by $I_{k,i}, k, i \in [0, N-1]$ for an $N \times N$ bit multiplier, (where k is the row index that starts from 0 and ends at $N-1$ and i is the column index that has the same range), is defined as follows

$$I_{k,i} = V_{in_k} \times G_{k,i} \, , \tag{3.4}$$

where $G_{k,i}$ represents the memductance of the cell at the pathway between the k_{th} row and the i_{th} column. For convenience, we disregard the resistance value of the transistor during our reasoning process. However, this does not affect the generality of our analysis because this value is simply a constant offset term.

As shown previously in Eq. (3.2), the column-wise term P_i is then generated by amplifying I_{out_i} by a gain factor g_i, where I_{out_i} is the output current of the i_{th} column. As a result, P_i can be expressed as:

$$P_i = g_i \times I_{out_i} \, . \tag{3.5}$$

In the crossbar array, the column current I_{out_i} is the sum of the currents from the cells selected based on the multiplier row operand values, which are given by Kirchhoff's current law (KCL) as:

$$I_{out_i} = \sum_{\substack{k=0 \\ i=0}}^{N-1} a_{k,i} i_{k,i} \, , \tag{3.6}$$

where $a_{k,i}$ is the number of cells that contribute to the PP term, i.e., the current on the i_{th} column. The gain g_i follows the relationship above as follows:

$$g_i = 2^i \, . \tag{3.7}$$

In current-mode, the amplification of the output current is achieved by using suitably selected CM ratios. Using Eq. (3.7) the column-wise term P_i can be expressed as:

$$P_i = g_i \times I_{out_i} = 2^i \sum_{\substack{k=0 \\ i=0}}^{N-1} a_{k,i} V_{in_k} G_{k,i} \, . \tag{3.8}$$

The final product of the multiplication step is the accumulation (i.e., the sum) of all the column-wise terms as shown in Eq. (3.3). To enable completely carry-free accumulation of the current using the KCL, the column-wise terms after amplification are connected in parallel. As a result, the final product I can be written as:

$$
\begin{aligned}
I &= \sum_{i=0}^{N-1} P_i = \sum_{i=0}^{N-1} g_i \times I_{out_i} \\
&= \sum_{i=0}^{N-1} (2^i \sum_{\substack{k=0 \\ i=0}}^{N-1} a_{k,i} V_{in_k} G_{k,i}) \ .
\end{aligned}
\tag{3.9}
$$

716 To provide a detailed illustration, the following two examples are considered:

$$\text{Ex. 1}: \quad M_1 \times M_2 = 1110 \times 1111 = 11010010b \ (210d) \tag{3.10}$$

$$\text{Ex. 2}: \quad M_1 \times M_2 = 1101 \times 0110 = 01001110b \ (78d) \tag{3.11}$$

717 For the examples above, the respective cell numbers for each of the cases are presented in Table 3.2. Assuming that $G_{k,i} = m$ (i.e., the conductance of the memristor in the ON

Table 3.2: Cell Values and Path Currents in Eq. (3.11)Ex. 1 and Ex. 2

Case	i / $\alpha_{k,i}$ / k	1	2	3	4	5	6	7
Ex.1	1	/	/	/	1	1	1	0
	2	/	/	1	1	1	0	/
	3	/	1	1	1	0	/	/
	4	1	1	1	0	/	/	/
α_i(Final Product Contributor Cell Number)		1	2	3	3	2	1	0
Ex.2	1	/	/	/	0	0	0	0
	2	/	/	1	1	0	1	/
	3	/	0	0	0	0	/	/
	4	1	1	0	1	/	/	/
α_i (Final Product Contributor Cell Number)		1	1	1	2	0	1	0

718

719 state) and $V_{in_k} = n$ (i.e., the switching voltage on the cell row), the PP currents and the

corresponding transformations are given below for both examples. The $a_{k,i}$ values from Table 3.2 are used here to derive the output current I.

Ex. 1:

$$I = 0 \times 2^0 \times mn + 1 \times 2^1 \times mn + 2 \times 2^2 \times mn + 3 \times 2^3 \times mn + 3 \times 2^4 \times mn + 2 \times 2^5 \times mn + 1 \times 2^6 \times mn + 0 \times 2^7 \times mn = 210mn\ (Amp)$$

Ex. 2: $I = 0 \times 2^0 \times mn + 1 \times 2^1 \times mn + 1 \times 2^2 \times mn + 1 \times 2^3 \times mn + 2 \times 2^4 \times mn + 1 \times 2^5 \times mn + 0 \times 2^6 \times mn + 0 \times 2^7 \times mn = 78mn\ (Amp)$

As shown, the results above match the expected outcomes for the multiplication operations. In Section 3.1.5, the implementation details are presented, and later the experimental results will be studied and compared with those from the traditional multiplier circuits.

3.1.3 Single Transistor Single Memristor Cell

The building block for the crossbar array is a 1T1M cell, which is illustrated in Fig.3.3. The memristor values represent one set of operands, while the voltage signals in the RLs represent the other set of operands [18].

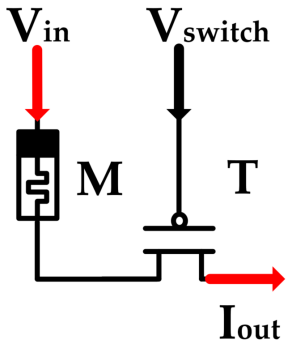


Figure 3.3: 1T1M cell. This building block for the crossbar array consists of a memristor and a transistor.

The 1T1M logic cell (LC) uses the memristor (W/L = 10 nm/10 nm [83]) as the memory unit, and the transistor (W/L = 1 μm/60 nm) as the switching unit. The

memristor is able to maintain its resistance state while using a below-threshold biasing power supply. When the memristor's biasing voltage exceeds its threshold, a set voltage (SV) biases the memristor into a low resistance state (LRS) or a reset voltage (RSV) biases it into a high resistance state (HRS). We designate the LRS to be a logic ″1″ and the HRS to be a logic ″0″ for the memristor working states. Fig. 3.4 depicts the responses of a standalone memristor to different writing biasing voltages using its logic state variations on the crossbar multiplier. The label 'nx' in Fig. 3.4 represents the amplification ratio at the output terminal. Figure 3.4(a) and (b)show that the writing speeds of the LCs with the different amplification ratios follow the same decreasing trend, i.e., $1\times$, $64\times$, $2\times$, $32\times$, $4\times$, $16\times$, and $8\times$. This occurs because columns with fewer LCs exhibit lower resistance than columns with more LCs. During the writing operation, a column with fewer LCs will receive a higher voltage when compared with the voltage for a column with more LCs. Consequently, the column with fewer LCs has a faster writing speed than the column with more LCs. Additionally, the amplifying circuit at the output terminal has the same effect on the columns. Specifically, columns with the same number of LCs will still exhibit different writing speeds; a column with a more extensive amplifying circuit will be slower during writing operations. The results in Fig. 3.4 (c)-(g) demonstrate that slight variations in the biasing voltage will cause slight changes in the LC writing speeds for all columns.

The LC operation involves three distinct processes, which are designated α, β, and γ. During process α, the tunable memristor resistance state is adjusted to be at a low level, which enables writing of a logic ″1″. Conversely, during process β, the tunable memristor resistance state is adjusted to be at a high level, which enables writing of a logic ″0″. In process γ, the memristor is used to perform reading and multiplication operations. The peak writing voltage value is given by write 1/write 0 = -3.5 V/3.5 V.

In the context of process γ, a voltage of 0.4 V is designated as the logic ″1″ in multiplier ″x″, while a voltage of 0 V is designated as the logic ″0″. The HRS and LRS of the memristor correspond to the logic ″1″ and logic ″0″, respectively. The current product generated from the voltage/resistance pair can also be used to represent the logic ″1″ and logic ″0″. Specifically, the logic ″1″ current can only be generated by applying the logic ″1″ voltage to the logic ″1″ resistance.

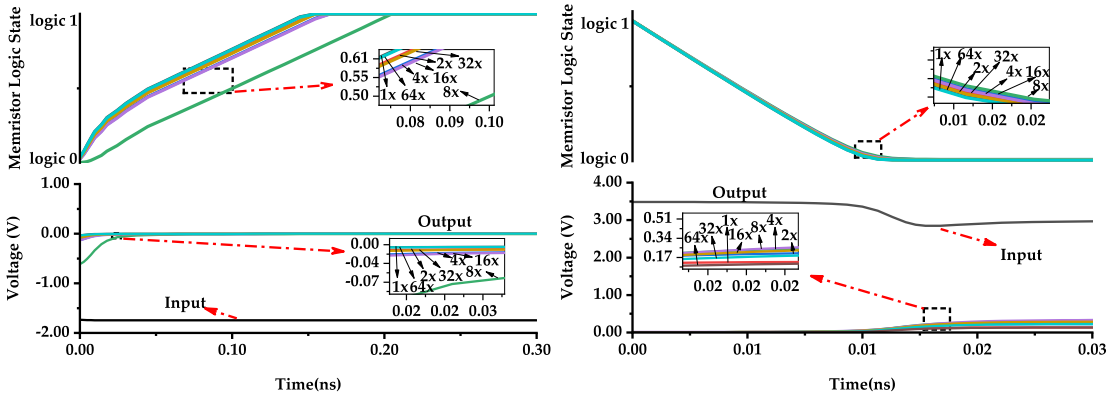


Figure 3.4: Responses of the memristor to writing biasing. In (a) and (b), under biasing by a DC voltage, the operations of writing logic 0 and logic 1, respectively, are shown.

Fig.3.6 shows an illustration of a 4-bit multiplier design using the proposed memristor-transistor transistor crossbar array. In the crossbar multiplier approach, voltage biasing is used to set all the operations; therefore, the voltage threshold memristor model is most appropriate for modelling of the operations of these multipliers. Simultaneously, stable and typical memristor behavior is also required in multiplier design. As a result, Kvatinsky's Voltage ThrEshold Adaptive Memristor (VTEAM) model and its associated physical parameters are used in this work.

Table 3.3: Voltage ThrEshold Adaptive Memristor Model Parameters taken from [52]

$alpha_{off}$	4	$alpha_{on}$	4
V_{off}(V)	0.3	V_{on}(V)	-1.5
R_{off}(Ohms)	300K	R_{on}(Ohms)	1K
k_{off}(m/s)	0.091	k_{on}(m/s)	-216.2
w_{off}(nm)	3	w_{on}(nm)	0

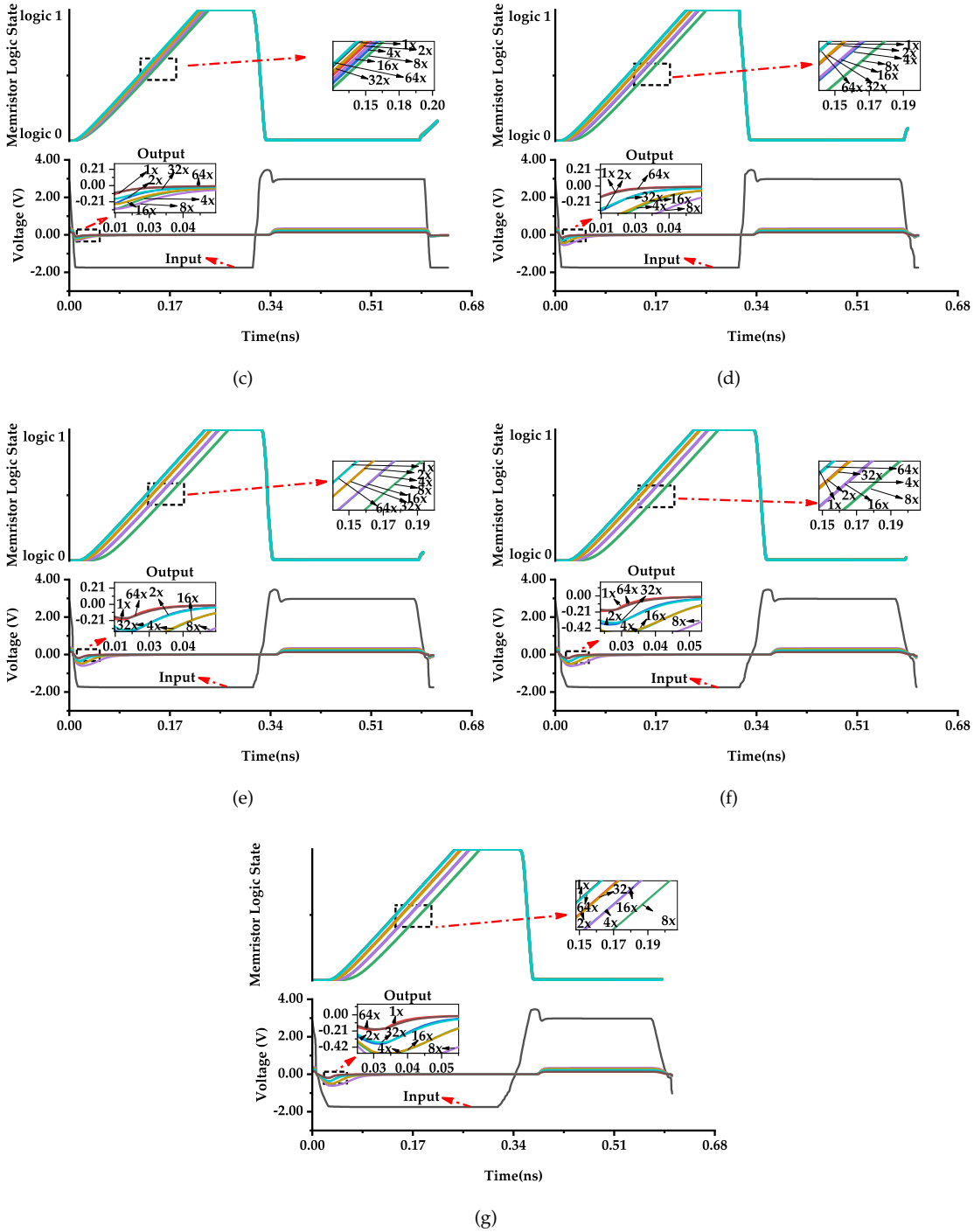


Figure 3.4: Responses of the memristor to writing biasing. In (c), (d), (e), (f), and (g), the biasing pulses have the same amplitude but differ in their rise/fall times, with voltage peak values of 3.5V/-3.5V. The pulse rise/fall time pairs are 10 ps/10 ps, 20 ps/20 ps, 30 ps/30 ps, 40 ps/40 ps, and 50 ps/50 ps, respectively.

3.1.4 Current Amplification

To improve energy proportionality, use of analogue current-mode arithmetic circuit designs has recently gained momentum [72]. These circuits operate using a dynamic range of currents (from μA to several mA)and provide considerably greater energy efficiency leverage than voltage-mode circuits, along with the added advantage of high slew rate and simpler circuitry. For example, when using CM networks, concurrent additions can be performed by directing the current paths into a single node, and subtractions can be performed by controlling the current paths away from a node. Because of their reduced circuit complexity, these networks can also offer faster operation with significantly reduced energy consumption [73, 84].

Before being input into the current accumulation (CMA) circuit, all the output currents simply show the numbers of 1T1M cells in their working state on each result line. Binary multiplication has different digit orders. Therefore, there should be a link between the RL output current and the binary number system's digit order. This pathway requires the use of groups of current amplifiers to provide the list of ratios according to Eq. (3.7). The corresponding CMA circuit also causes the j_{th} RL to generate the j_{th} digit of the result. In the proposed multiplier, the CMA circuit's unit structure is an

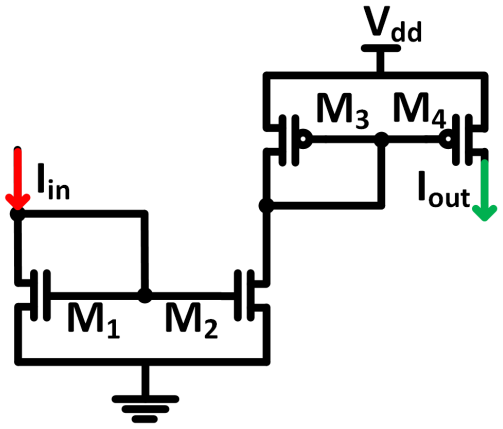


Figure 3.5: Multi-amplifier design for the current summer circuit. This design is built using an n-type CM that is series-connected to a p-type CM.

n-type CM that is coupled with a p-type CM as shown in Fig.3.5. The n-type CM takes in

the output current from the crossbar structure, and the p-type CM generates the output current. During the amplification procedure, the current is amplified twice using the n-type CM and the p-type CM. The multi-amplifier design can reach its target gain with a smaller overall size. Meanwhile, the gate voltage must be maintained at a reasonable level. In this way, the multi-amplifying design can avoid the problem where the need for a high amplification ratio will require an extremely large transistor to be included in the single CM. In other words, the transistor used in the proposed CMA circuit can be much smaller than a single-layer CM intended for use in the same task. The current leakage generated in each multiplication cell can also be amplified and this appears to present a new problem. However, this effect is at a negligibly low level when compared with the output current [73].

3.1.5 4-bit Crossbar Multiplier Implementation

We discuss the implementation details of the multiplier in the following.

In the multiplier circuit shown in Fig. 3.6, basic 1T1M cells are organised at each cross point (i.e., each node) via the mapping procedure. This design provides a combination of high-speed operation and accurate cell selection. Both the input and the control signals are applied in the form of a single bar source (SBS). Use of the SBS means that the source covers the power supplies of all 1T1M cells when they are connected to the same row bar, or it covers the control signals of all 1T1M cells when they are connected to the same column bar. For the same expression, the row bars that receive the input signals are called source lines (SLs), the column bars that receive the control signals are called gate lines (GLs), and the column bars that produce the output signals are called result lines. In our studies, we have used the VTEAM model [56] with the model parameters from [52] for the memristors used in the circuit. These parameters have been extracted from physical devices. This ensures that our design can be implemented in practice. The actual parameters are listed in Table 3.3.

The input voltages are Boolean and each voltage represents one of the operands, with input4 being the MSB and input1 being the LSB. The output currents represent the product values at each output bit positions, having been accumulated from the bit multiplications that occur in each bit position. Because the current values go beyond the

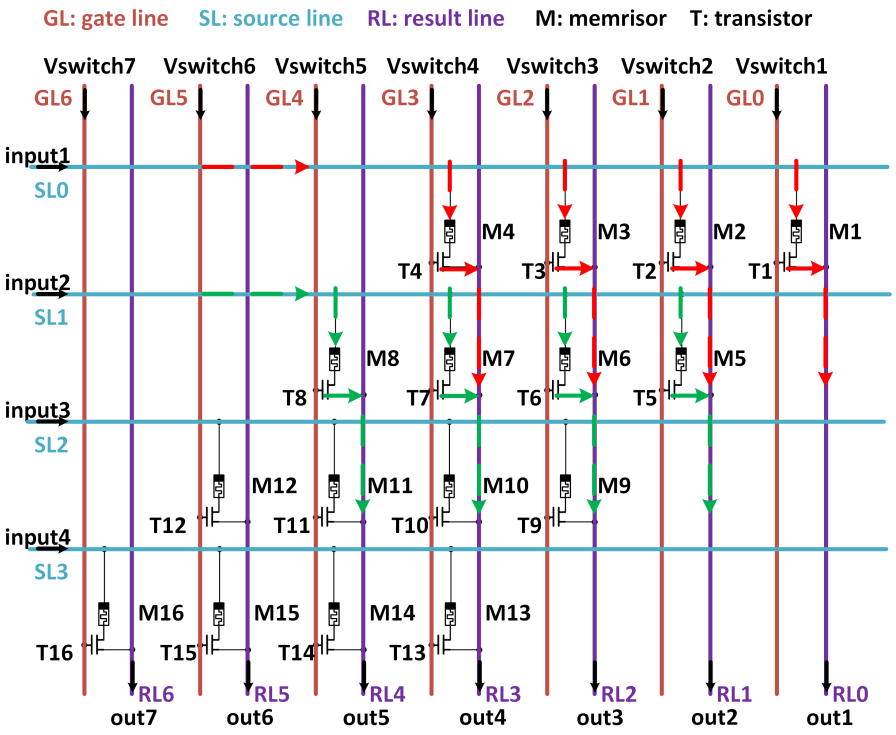


Figure 3.6: 4 by 4 1T1M crossbar circuit with three line settings, one RL, and two parallel CLs that are defined to give the circuit the ability to select any cell within the circuit.

Boolean values at each bit position, the multiplier only requires seven output columns rather than the 8 bits required for a digital multiplier with two 4-bit operands.

3.2 Simulation Results

The proposed design is based on UMC 65 nm circuit technology. The transistors are divided into two groups, designated LC and CM, as shown in Fig. 3.2. All LCs contain transistors of the same size; these transistors are 1000 nm width and 60 nm length. At the output terminal, the n-MOSFET and p-MOSFET CMs are connected in series to achieve high ratio output current amplification. Because the CMs work as amplifiers with individual gains, their transistor sizes differ as shown in Table 3.4.

In simulation experiments, a 4 by 4 crossbar multiplier is used to illustrate the multiplication process. The multiplication operation is executed between two 4-bit

Table 3.4: Transistor Sizes for the Current Mirrors

Group	n-MOSFET		p-MOSFET	
	M1 (nm)	M2 (nm)	M3 (nm)	M4 (nm)
1	1520/60	400/60	80/60	240/60
2	2720/60	1600/60	80/60	260/60
3	3840/60	2400/60	80/60	720/60
4	5440/60	3200/60	80/60	1680/60
5	4080/60	4800/60	80/60	1920/60
6	2720/60	4800/60	80/60	2680/60
7	1520/60	1840/60	80/60	5120/60

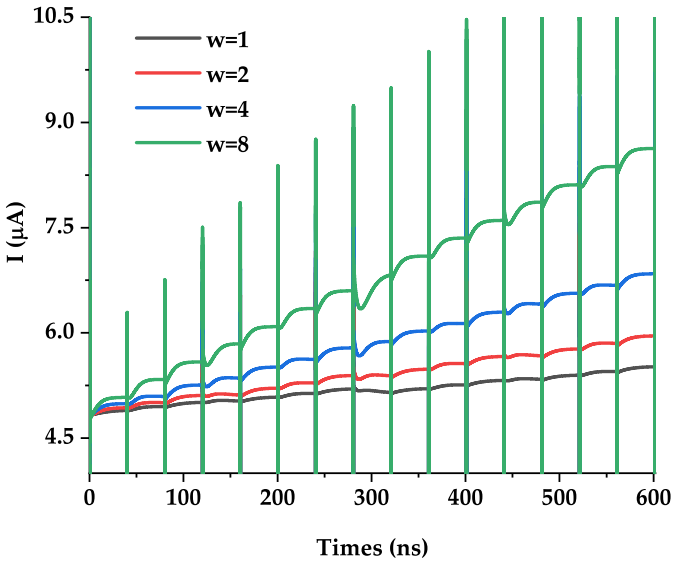
binary operands. One of these operands, which is denoted by the input variable "x" (voltage), ranges from 0 (0000) to 15 (1111), while the other operand, which is denoted by the weight or the reference "ω" (memristance), remains constant at 1 (0001), 2 (0010), 4 (0100) and 8 (1000). These values are chosen because they have only one bit high in binary representation, and any other number below 16 can be obtained by different combinations of them. The results of the multiplication operation, from both theory and the simulation, are presented in Fig. 3.7.

Figure 3.7(a) illustrates the output obtained from the proposed multiplier when operating in current mode. For example, the final step shown in Fig. 3.7(a) represents the outcome of multiplying x = 1111 by ω = 0001, 0010, 0100 and 1000. In the circuit, this means that:

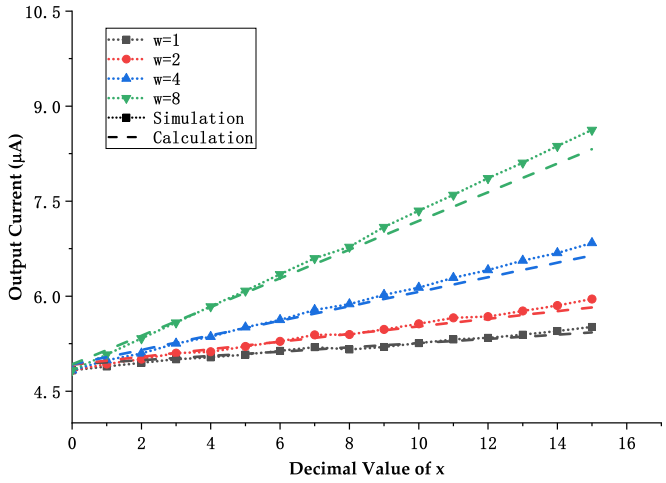
1. The input voltage series to the crossbar in Fig.3.6 is input1 = 0.4 V, input2 = 0.4 V, input3 = 0.4 V, and input4 = 0.4 V.

2. The switching voltage series to the crossbar in Fig.3.6 is Vswitch1 = 1.2 V, Vswitch2 = 1.2 V, Vswitch3 = 1.2 V, Vswitch4 = 1.2 V, Vswitch5 = 1.2 V, Vswitch6 = 1.2 V, and Vswitch7 = 1.2 V.

3. The memristors in selected LCs on the crossbar in Fig.3.6 are biased to the LRS, while all the remaining memristors are in the HRS. Starting from the right, for



(a) Multiplication outcome from the proposed multiplier and comparison with the expected results



(b) Calculation results comparison with simulation results for the 4-bit multiplication of w=1, w=2, w=4, and w=8

Figure 3.7: Multiplication performance for a 4-bit case.

a pattern of 0001, the memristors in the first LC are in the LRS. For a pattern of 0010, the memristors in the second LC are in the LRS. For the pattern of 0100, the memristors in the third LC are in the LRS. For the pattern of 1000, the memristors

in the fourth LC are in the LRS.

It is clearly shown that in the 4 by 4 crossbar multiplier in Fig. 3.6, the LSB refers to different items in different cases. For an input voltage series, the LSB is ”input1” in the circuit; for the final product, the LSB is ”out1” in the circuit; and for the memory, the LSBs are ”M1”, ”M5”, ”M9”, and ”M13” in the circuit. Using the same method, the MSBs for the input voltage series, the final product, and the memory in the circuit are ”input4”, ”out7”, and ”M4”, ”M8”, ”M12”, and ”M16”, respectively. The rising stairs characteristic means that the input ”x” binary value increases step-by-step from ”0000” to ”1111” with respect to the increasing input voltage series and generates specific currents to output the calculation result. The results graph presented in Fig.3.7(a) shows that the multiplication results increase with increasing input, as expected.

Figure 3.8 shows the timing diagram of the control signals of the memristor which demonstrates the ability to select a specific cell or multiple cells for reading (multiplication) and writing (operand setting) processes. Both tuning operations on the multipliers and the multiplicands are included.

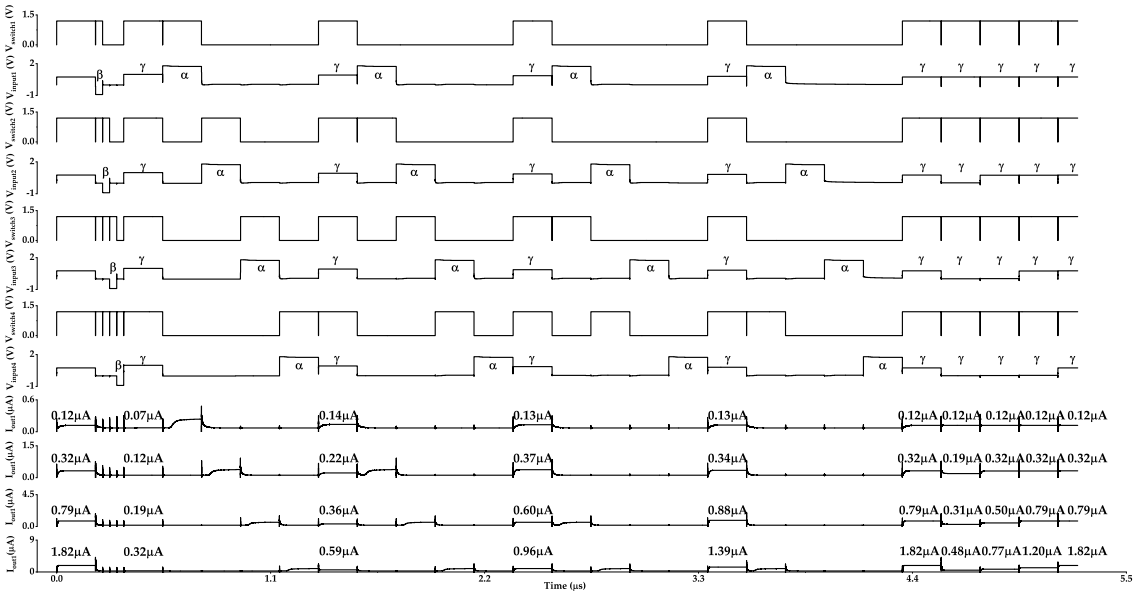


Figure 3.8: Writing and multiplying procedures of 1T1M crossbar multiplier.

The complete writing and multiplication procedures are shown in Fig. 3.8 and can be divided into three operational stages, as follows. All reading procedures were set to use a

0.2-μs pulse, the write logic 1 operation was set to use a 0.2-μs pulse, wand the write logic 0 operation was set to use a 36-ns pulse; additionally, all transistors were used to isolate the deselected LCs. The first stage starts from the state in where all 1T1M cells are in the LRS. The target result for each RL is confirmed during the period from 0 μs to 0.2 μs. The second stage runs from 0.2 μs to 4.544 μs. This stage shows the procedure used to modify the 1T1M cell state to change the output state. All operations consist of modifying all the 1T1M cells into the HRS, and then modifying all 1T1M cells on the same SL to return to the LRS until all the 1T1M cells are in the LRS. The results of each modification procedure are also monitored to show the changes in each RL output. The third stage runs over the period from 4.544 μs to 5.344 μs. This stage shows the procedure used to modify the input state to change the output state. All 1T1M cell states are kept in the LRS, and the input signals for each SL are then changed one-by-one from 0 V to 0.4 V. The third stage also provides the relationship between the current and the multiplication procedure. It is easy to see that both the 1T1M cell state modification process and the input variation procedure generated the same results. Therefore, the designed multiplier does indeed perform the multiplication operation. The step-by-step procedures shown in Fig. 3.8 can be described as follows:

1. 0 μs-0.200 μs: execute 1111 $\times$ 1111
2. 0.200 μs-0.344 μs: change multiplier 1111 to 0000
3. 0.344 μs-0.544 μs: execute 0000 $\times$ 1111
4. 0.544 μs-1.344 μs: change multiplier 0000 to 0001
5. 1.344 μs-1.544 μs: execute 0001 $\times$ 1111
6. 1.544 μs-2.344 μs: change multiplier 0001 to 0011
7. 2.344 μs-2.544 μs: execute 0011 $\times$ 1111
8. 2.544 μs-3.344 μs: change multiplier 0011 to 0111
9. 3.344 μs-3.544 μs: execute 0111 $\times$ 1111
10. 3.544 μs-4.344 μs: change multiplier 0111 to 1111

11. 4.344 μs-4.544 μs: execute 1111 $\times$ 1111

12. 4.544 μs-4.744 μs: change multiplicand 1111 to 0001 and execute 1111 $\times$ 0001

13. 4.744 μs-4.944 μs: change multiplicand 0001 to 0011 and execute 1111 $\times$ 0011

14. 4.944 μs-5.144 μs: change multiplicand 0011 to 0111 and execute 1111 $\times$ 0111

15. 5.144 μs-5.344 μs: change multiplicand 0111 to 1111 and execute 1111 $\times$ 1111

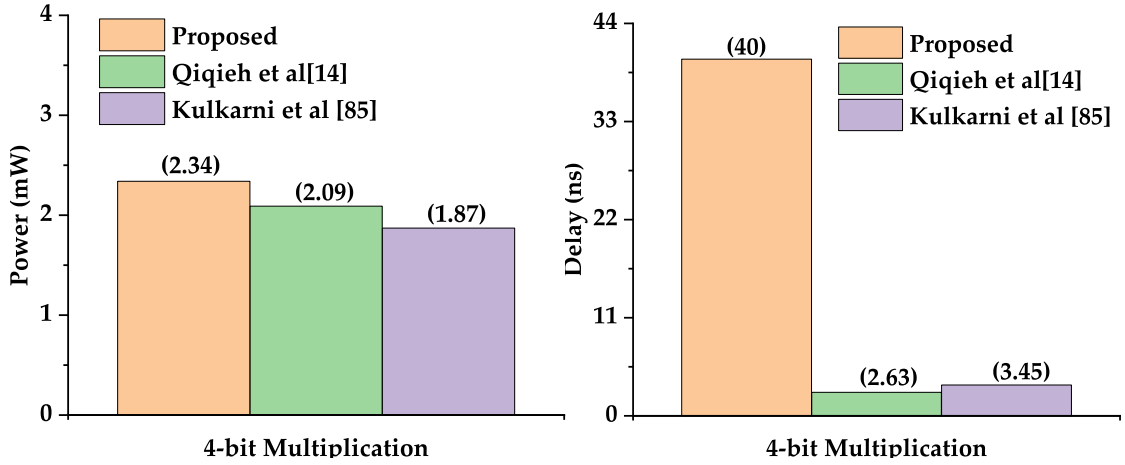


Figure 3.9: Comparative analyses of multiplier power, and delay. In (a), the power consumption of the proposed design is 2.45 mW, that of Qiqieh's approach [14] is 2.09 mW, and that of Kulkarni's approach [85] is 1.87 mW. In (b), the proposed multiplier shows a 40 ns delay, while Qiqieh's approach produces a delay of 2.673 ns, and Kulkarni's approach [85] shows a delay of 3.45 ns.

To validate our multiplier design via comparison, the proposed multiplier is evaluated against existing approximate designs, e.g., by remapping theresulting product to a lower significance by compressing PPs [14] or applying a low precision multiplier (i.e. a 2$\times$2 multiplier) as a building block for a larger multiplier [85]. For all validation experiments, the base parameter settings of the proposed multiplier are listed as follows: V_{in} = 0.4 V, V_{dd} = 1.2 V, R_{LRS} = 1kΩ, and R_{HRS} = 300k Ω. These parameters are applied

in the power model for energy consumption performance, and the values obtained from Qiqieh [14] and Kulkarni [85] are also used to perform a comparative analysis.

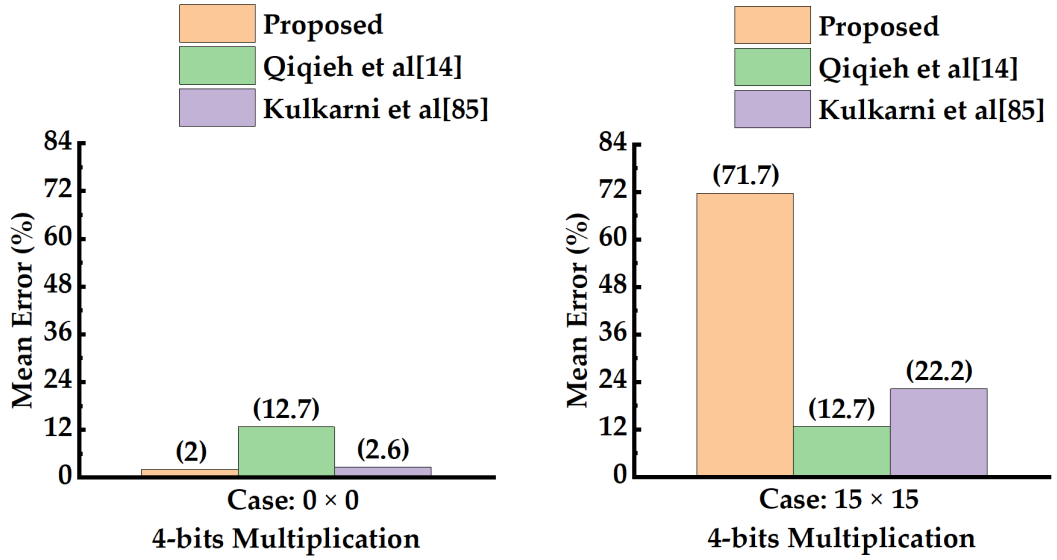


Figure 3.10: Comparative analysis of 4-bit multiplication accuracy. In (a), the low error level comparison results show that the proposed design has the lowest mean error (ME) at 2%, followed by that of Kulkarni at 2.6% with Qiqieh having the highest at 12.7%. In (b), the situation is reversed in the high error level comparison progress, with Qiqieh having the lowest ME at 12.7%, and Kulkarni still in the middle at 22.2%. The proposed multiplier shows the highest ME at 71%.

In Fig. 3.9 and Fig. 3.10, the 4-bit multiplication performances in terms of power consumption, delay, and mean error have been compared. The mean error (ME) shown in Fig. 3.10 is the product of the equation $ME = \frac{\sum(simulation\ current - prediction\ current)}{total\ number\ of\ multiplication\ group} \times 100\%$, where the simulation current and the prediction current are the corresponding result currents in a single multiplication group. Fig. 3.9 compares the power and delay of the proposed design with the works in [85] and [14] for single 4-bit multiplication. The proposed design consumes 20% more power than the work in [85] and and 11% more than the work in [14]. The proposed design also has 92% longer delay than the work in [85] and and 94% longer than the work in [14]. However, when the

memristance operand is constant, the proposed design has almost zero delay, which is 97% shorter than both the work in [85] and the work in [14]. Figure 3.10 illustrates both the minimum case and maximum case of mean error (ME) between the different approximate multiplier designs and that of the proposed work. The work in [14] compresses PP by adding a logic gate between the tree adders, which performs the approximation operation in the middle of the multiplication progress. The work in [85] applies a low accuracy multiplier for a large multiplier, where the approximation is performed in the middle of the multiplication progress. The proposed work eliminates the need for both carry propagation and an explicit DAC, because its approximation is performed at the end of the multiplication progress. In details, in the minimum case, the proposed design shows an 84.25% lower ME than the work in [14] and a 23.08% lower ME than the work in [85]. In the maximum case, the proposed design has an 82.29% higher ME than that of the work in [14] and 69.04% higher than that of the work in [85].

There are still some issues to be overcome in the early stages of the project. For example, several stairs were increased over the size of the next level. This is because the LRS of the LC causes a higher voltage drop than the HRS, which results in a lower current being generated for a logic ”1” and a higher current being generated for a logic ”0”. Additionally, errors in the output current can be amplified by the CM circuit, which is also affected by the terminal voltages. In the case of the logic ”1” current, a higher voltage drop leads to a greater reduction in the CM amplifier gain, while the logic ”0” current with the lower voltage drop realizes higher gain than it actually should. As a result, the current level of the LSB logic ”1” can be lower than the corresponding level of the MSB logic ”0”.

3.3 Summary

In this chapter, we have presented a mixed-signal digital input (DI)/analog output (AO) multiplier that uses current-mode principles to achieve carry-free computation. The resulting reduction in circuit complexity leads to significant improvements in both computational latency and power consumption. To evaluate the proposed approach, we compare the proposed multiplier’s performance with that of existing 4-bit approx-

imate multiplier designs in terms of energy consumption, delay, and accuracy. Our results demonstrate that the proposed crossbar array offers deterministic precision and consumes much less energy than the other designs, yielding power savings of up to 50%. This makes our proposed device particularly relevant for use in edge applications, where computation units are powered using limited energy sources with unpredictable or sporadic power supplies. Additionally, the use of memristors ensures the retention of the most stable operand in the face of power discontinuities.

Current amplification using CMs may seem intuitive, but it results in a significant CMOS overhead during multiplier design, along with several disadvantages. These disadvantages include requirements for transistors of different sizes, latencies, and energy penalties because of the switch-on and switch-off processes of large transistors. Additionally, the delay that arises from switching of large transistors presents a significant challenge in terms of reducing the multiplier's delay.

Chapter 4

Memristive Multiplier Design with In-cell Current Multiplication

The transistor-memristor crossbar multiplier scheme requires current amplification that corresponds to the bit locations of the current signals. Specifically, for the current at bit i, denoted as I_i, an amplification of 2^{i-1} is needed. As discussed in the previous chapter, the single transistor single memristor (1T1M) cell (shows in Fig. 4.1) can only generate I_i without amplification of 2^{i-1}. Therefore, current amplification in the 1T1M crossbar multiplier is achieved through the use of amplifiers based on CMs. In this chapter, current amplification is further optimised by generating the correct current value directly within each transistor-memristor cell by using multiple parallel memristors in a cell. Intuitively, if an amplification of 2^{i-1} is needed, a cell with 2^{i-1} parallel memristors can satisfy this need because of this **in-cell current amplification**.

By doing this, the CMA can be omitted removing the high-energy high-latency charging and discharging of potentially large capacitance and replacing them with in-cell resistive arithmetic according to Ohm's Law, which has zero theoretical latency. The method presented in this chapter is based on the single transistor multiple memristor (1TxM) cell structure, shown in Fig. 4.2. Developed from 1T1M cell by extending amount of memristor in parallel with ratio of 2^{i-1} in single cell, 1TxM cell structure can be used in

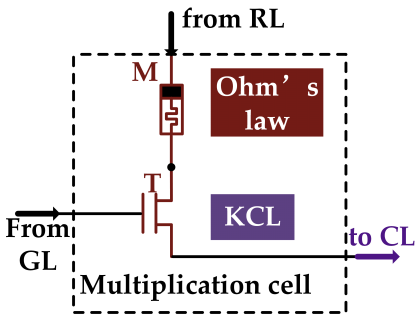


Figure 4.1: The structure of 1T1M cell with updated details. Transistor is in n-type, memristor applied Cu:ZnO thin film.

the same crossbar structure from the 1T1M-based multiplication scheme presented in the previous chapter, by replacing each 1T1M cell with a 1TxM cell where $x = 2^{i-1}, i \in [0, N]$. With this crossbar the CMA circuits can be entirely removed and the significance of each current path is directly set by x and already correct at the cells.

Individually, different current paths are then directed to the output node which accumulates the currents according to KCL, thus requiring no carry propagation. This allows for better performance and energy efficiency characteristics than conventional multipliers.

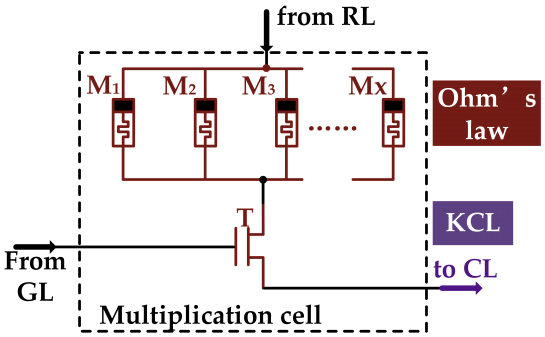


Figure 4.2: 1TxM cell structure. For a cell along the current path for bit i, $x = 2^{i-1}$.

In this chapter, the 1TxM design will be compared with the 1T1M multiplier from the previous chapter. For fair comparison, the 1T1M-based crossbar multiplier is redesigned with an updated cell structure, shown in Fig. 4.1. The difference between the 1T1M cell in

Chapter 3 (Fig. 3.3) and the one in this chapter is that the p-type transistor of the previous chapter is replaced by the n-type transistor here. The p-type transistor works better as a current source, and the n-type transistor works better as a current sink. When operating in switching mode, these characteristics can affect the circuit's output current. While this may not be significant for 1T1M cells, it is crucial for 1TxM cells, where n-type transistors perform better due to the elimination of the CMA below. To make a fair comparison, both p-type and n-type transistors were placed after the memristor, resulting in all cells having an n-type connection. Because of this cell update, the aspects and emphasise in the comparative studies, and for the ease of reading, some of the aspects of the 1T1M multiplier will be presented again in this chapter.

4.1 Number Representation and Encoding

This section further clarifies the method of representing numbers in the proposed multipliers.

As proposed in Chapter 3, in a conventional ($N \times N$) binary multiplier, two unsigned integers can be multiplied using N^2 logic AND operations, followed by up to 2N ADD operations. Meanwhile, a carry propagation procedure is required for the generation of each midterm product.

Multiplication is different in a crossbar multiplier, which directly implements the long multiplication algorithm and whose structure can be seen in Fig. 4.3. The 1T1M cell locates at each intersection of the crossbar, connecting a column (CL) to a row (RL). Such a cell has its position indexed with both row k and column i according to its location on the crossbar network and is called $C_{k,i}$. $C_{k,i}$ connects the kth RL and the ith CL. It gets voltage V_{Ck}, and its memductance is $G_{k,i}$. This cell implements the single-bit multiplication between the operands V_{Ck} and $G_{k,i}$, with the product being $I_{Ck,i}$, according to Ohm's Law as described in Table 4.1. The particular indexing system used for the memductance is relevant to implementing the multiplication algorithm in Table 3.1, where the same operand bit m_k features in diagonally placed cells from upper right to lower left in the addition part of the algorithm. These cells thus implement the N^2 logic single-bit AND (bit-wise multiplication) operations required for the first step of

Table 4.1: Single Transistor Multiple Memristors Cell Operations

Data Representation			
	Signal	Logic 0	Logic 1
Operand 1	V_C	$V_{C_{low}}$	$V_{C_{high}}$
Operand 2	G	G_{low}	G_{high}
Product	I_C	$I_{C_{low}}$	$I_{C_{high}}$
Memristor Operations			
Writing	$V_C > V_{th}$		
Reading	$V_C < V_{th}$		
Multiplication	$I_C = V_C \times G_M$ (Ohm's Law)		
Truth Table	V_C	G_M	I_C
	$V_{C_{low}}$ (0)	G_{low}(0)	$I_{C_{low}}$ (0)
	$V_{C_{low}}$ (0)	G_{high} (1)	$I_{C_{low}}$ (0)
	$V_{C_{high}}$ (1)	G_{low} (0)	$I_{C_{low}}$ (0)
	$V_{C_{high}}$ (1)	G_{high} (1)	$I_{C_{high}}$ (1)

multiplication, in parallel across all intersections of the crossbar.

For performing writing operation of cell $C_{k,i}$, the transistor in this cell should be in the ON state. As the same gate biasing voltage is applied to cells along the same column, when writing cell $C_{k,i}$, all other cells along column i should have their transistors in the OFF state to maintain their memductance state. In this way, any single memductance $G_{k,i}$ can be set to its target logic state corresponding to the correct operand value. At the same time, multiple gate biasing control allows the same writing operation for memristors on the same row.

The single-bit data operations of a 1T1M cell is summarised in Table 4.1 when it is used to perform single-bit multiplication. The cell voltage V_C is used to represent one operand and the memductance G is used to represent the other, whilst the cell current I_C represents the product of the two operands according to Ohm's Law.

Note that the reading mode is when the multiplication result is read out, and therefore

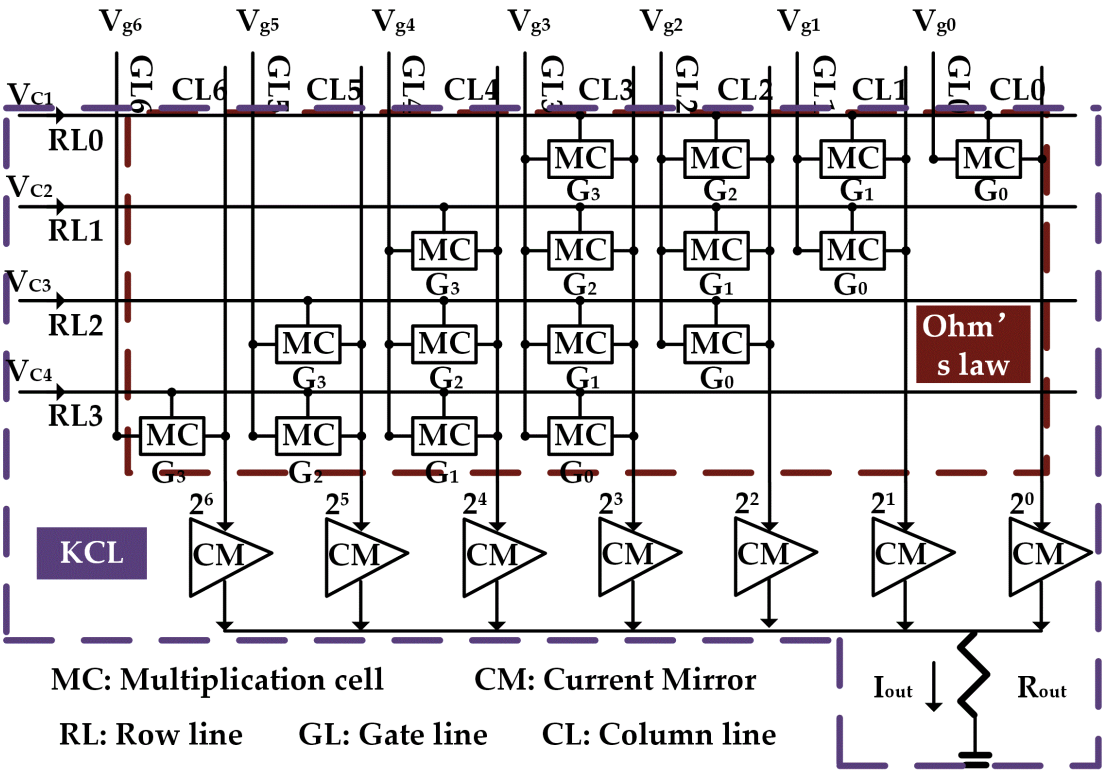


Figure 4.3: The architecture of 1T1M crossbar multiplier. The current amplification is implemented with CM. Each MC is a 1T1M cell described in Fig. 4.1, and a CM amplifier has one n-type CM and one p-type CM series connected.

$V_{C_{high}}$ must be lower than V_{th}. For single-bit Boolean multiplication, there needs to be enough separation between $I_{C_{high}}$ and the highest possible value of $I_{C_{low}}$ to ensure logical correctness. This can be realised by having $V_{C_{low}}$ = GND = (0 V) and $G_{high} >> G_{low}$. In this work we have up to $G_{high} = 1000 \times G_{low}$, a realistic margin of difference [48]. This, as demonstrated later in this chapter, is more than enough for a multiplication precision of 4 bits.

Since the result of the multiplication I_{out} is accumulated according to KCL, and because of the logic "0" and "1" definitions for currents given in Table 4.1, there exists the possibility that the accumulation of multiple $V_{C_{high}} \times G_{low}$ currents, which is the highest possible cell current value representing logic "0", pollute I_{out} enough to affect the accuracy of the result.

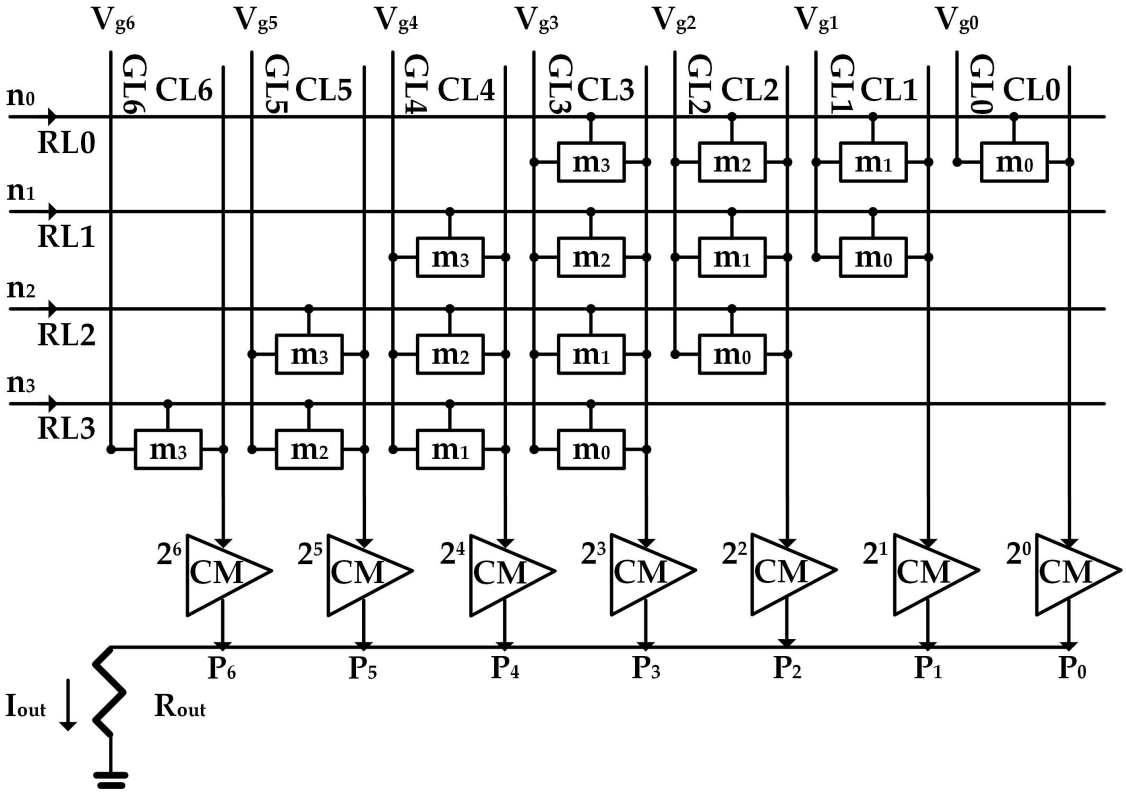


Figure 4.4: The mapping of numbers onto the crossbar structure with multiplication operands (M_1, M_2) and final product (P).

Fig. 4.4 illustrates how the long-multiplication algorithm maps onto the 1T1M

crossbar multiplier. The operand $M_1 : \{m_3\ m_2\ m_1\ m_0\}$ is represented by memductances $w_1 : \{G_3\ G_2\ G_1\ G_0\}$, and $M_2 : \{n_3\ n_2\ n_1\ n_0\}$ is represented by row voltages $w_2 : \{V_{in3}\ V_{in2}\ V_{in1}\ V_{in0}\}$ in Fig. 4.3. Single-bit multiplications at the cells are the result of Ohm's Law during reading mode, as described in Table 4.1. The output current of a single cell is the partial product which can be denoted as $I_{k,i}$, where k and i are the row and column indices, respectively. The operation is described by the following equation:

$$I_{k,i} = V_{ini} \times G_{k,i}, \tag{4.1}$$

where $k = 0, ..., 3$ and $i = 0, ..., 6$.

From Fig. 4.4, it can be seen that the worst case for such potential inaccuracy happens when the operands are $M_1 = \{1\ 1\ 1\ 1\}$ and $M_2 = \{0\ 0\ 0\ 0\}$, a case preliminarily explored in Chapter 3. Here we analyse it in detail. This multiplication results in all cells having the same current $I_{k,i} = V_{C_{high}} \times G_{low}$. With the following relations, $P_0 = 2^0 \times I_{0,0}$, $P_1 = 2^1 \times (I_{0,1} + I_{1,0})$, $P_2 = 2^2 \times (I_{0,2} + I_{1,1} + I_{2,0})$, $P_3 = 2^3 \times (I_{0,3} + I_{1,2} + I_{2,1} + I_{3,0})$, $P_4 = 2^4 \times (I_{1,3} + I_{2,2} + I_{3,1})$, $P_5 = 2^5 \times (I_{2,3} + I_{3,2})$, and $P_6 = 2^6 \times I_{3,3}$.

Respectively, we can get $P_0 = 2^0 V_{in0}\ G_0$, $P_1 = 2^1\ (V_{in0}\ G_1 + V_{in1}\ G_0)$, $P_2 = 2^2\ (V_{in0} G_2 + V_{in1} G_1 + V_{in2} G_0)$, $P_3 = 2^3\ (V_{in0}\ G_3 + V_{in1}\ G_2 + V_{in2}\ G_1 + V_{in3}\ G_0)$, $P_4 = 2^4\ (V_{in1}\ G_3 + V_{in2}\ G_2 + V_{in3}\ G_1)$, $P_5 = 2^5\ (V_{in2}\ G_3 + V_{in3}\ G_2)$, and $P_6 = 2^6 \times V_{in3} \times G_3$. Finally, all the partial product P_j current values are added up to generate I_{out}, which in this case can be presented as Eq. (4.2)

$$I_{out} = \sum_{j=0}^{6} P_j, \tag{4.2}$$

which encodes the result of the multiplication (overall product). The total number of digits for P_j shown in Fig. 4.4 is 1 bit less than that for the regular long multiplication algorithm in Table 3.1. This is because carries are not propagated to the left in the 1T1M crossbar mixed-signal multiplier.

$$\begin{aligned} I_{out} = &(2^0 + 2^1 \times 2 + 2^2 \times 3 \\ &+ 2^3 \times 4 + 2^4 \times 3 + 2^5 \times 2 \\ &+ 2^6) \times V_{C_{high}} \times G_{low} \\ &= 225 \times V_{C_{high}} \times G_{low}, \end{aligned} \tag{4.3}$$

The I_{out} value in Eq. (4.3) is supposed to encode logic "0". According to Table 4.1, logic "1" current at a single cell is

$$I_{C_{high}} = V_{C_{high}} \times G_{high} \, . \tag{4.4}$$

Combining Eq. (4.3) and Eq. (4.4), in order to avoid a bit error at the least significant bit, whose value is a single I_C, the following must be true

$$G_{high} > 225 \times G_{low} \, . \tag{4.5}$$

We choose up to $G_{high} = 1000 \times G_{low}$, which provides a substantial error margin for the 4-bit 1T1M crossbar multiplier. On the other hand, for a 5-bit multiplier with the same architecture, the minimum requirement for accuracy at the LSB is $G_{high} > 969 \times G_{low}$. Our chosen gap between G_{high} and G_{low} will be much less safe from accuracy problems at that level of precision. In other words, the upper limit of bit-width for such a multiplier depends on the chosen memristor technology and the multipliers are suitable mostly for low-precision applications.

4.2 Single Transistor Multiple Memristors Multiplier

4.2.1 Baseline Design

The 1T1M crossbar multiplier employs three main types of components, transistors serving as switches, memristors serving as adjustable conductance values, and CMs serving as bit significance weighting manager (providing the 2^i coefficients). CMs with high amplification ratios require radically disproportional sizing of their constituent transistors, as shown in Table 3.4. With more types of components involved in an analogue system, managing the effects of parametric variations becomes more complicated. In addition, the CMs used in the 1T1M crossbar multiplier all have different amplifications and sizes, which usually necessitates careful per-component design.

The 1TxM crossbar multiplier design seeks to reduce system design complexity by eliminating the need for CMs, thus reducing the types of used components. By moving the functionality of bit significance weighting from the different amplifications of CMs

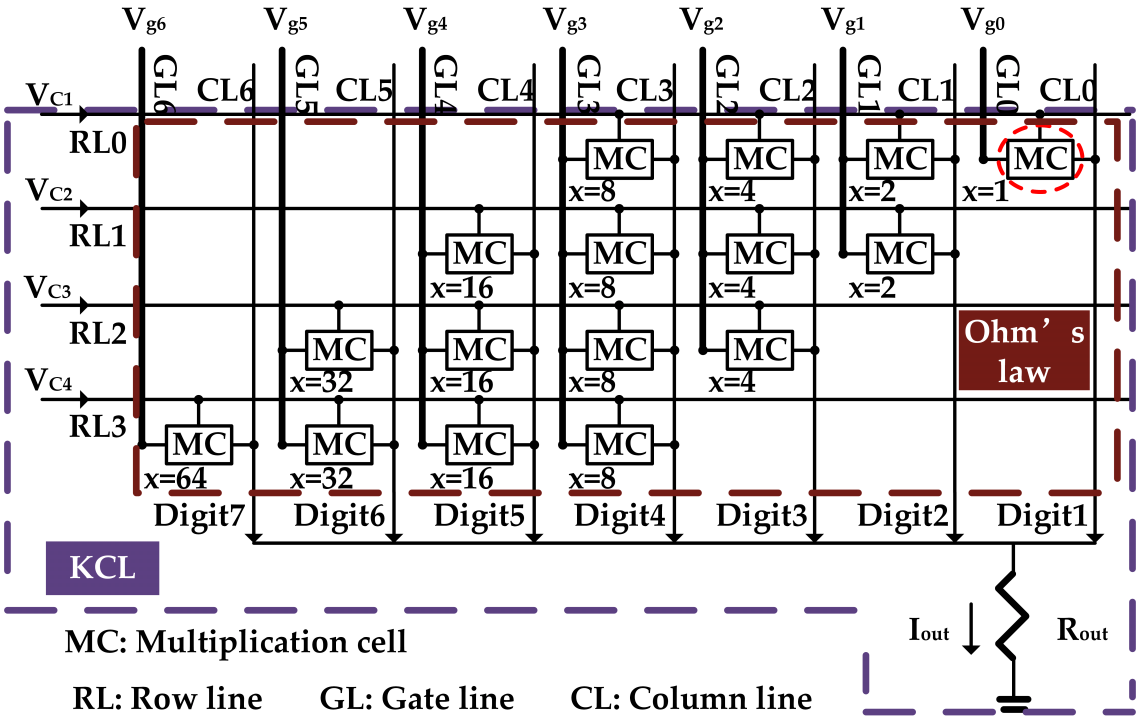


Figure 4.5: The Architecture of 1TxM crossbar multiplier. The number of parallel memristors x in a 1TxM cell is determined by its column location.

to the number of memristors in each cell, it also eliminates the need for components of different specifications. This is implemented by constructing each 1TxM cell, which connects a row with a column in the crossbar, with a single transistor switch controlling x memristors in parallel, as shown in Fig. 4.2. With such a cell, the bit significance weighting can be managed through the following equation $x_i = 2^i$, where x_i is the number of memristors in each cell in the ith column, $i \in [0, 6]$ for the 4-bit 1TxM multiplier. By setting x_i values this way, the column-specific CMs in Fig. 4.5 are functionally replaced by the number of memristors in the 1TxM cells (in Fig. 4.2). Note that the numerical significance of every memristor is exactly the same across the entire multiplier and parametric variations in any memristor have exactly the same degree of effect on the overall product, no matter where the memristor is located. This simplifies variation modelling and analysis as well as variations-aware design.

For this 1TxM multiplier, Eq. (4.1) no longer describes the cell current but instead

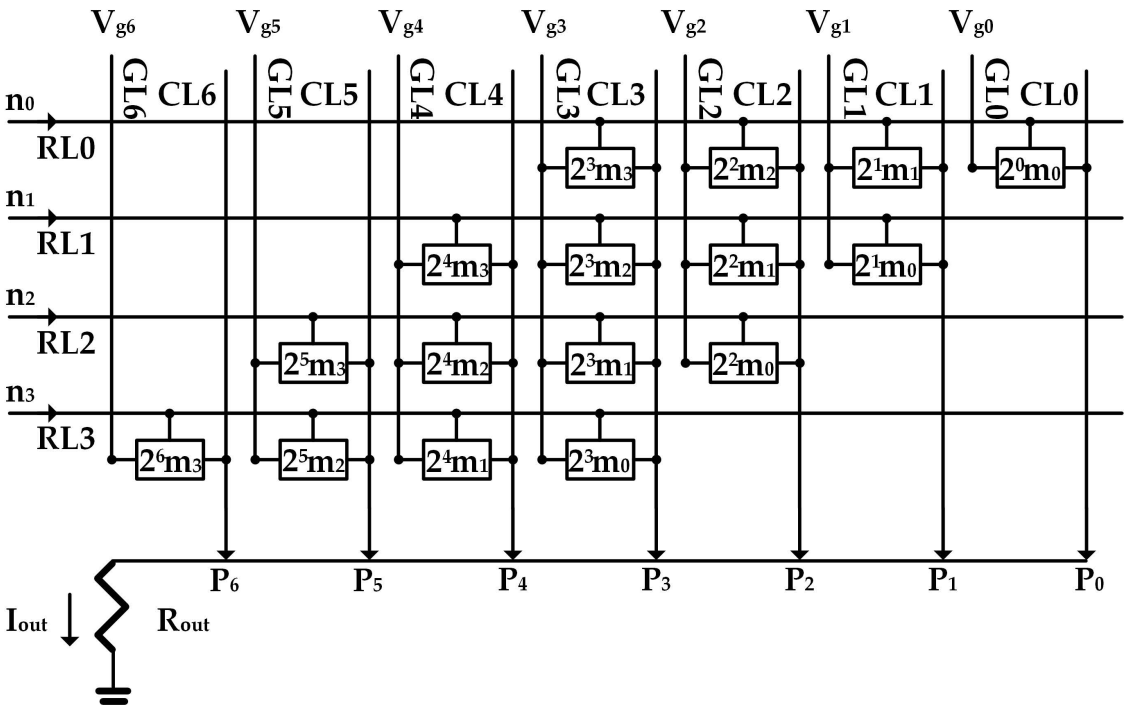


Figure 4.6: The 1TxM crossbar mapping with multiplication operands (M_1, M_2) and final product (P).

describes the current flowing through any one memristor within the cell located at intersection $\{k, i\}$ on the crossbar. With this revised understanding, $P_0 = 2^0 \times I_{0,0}$ in Eq. (4.2) remain the same. In other words, the 1TxM multiplier in Fig. 4.5 functions exactly the same as the 1T1M multiplier in Fig. 4.3. Fig. 4.6 clarifies this point when compared to Fig. 4.4. The multiplier precision analysis also remains the same.

A 1TxM multiplier using the same TiO_2 memristor technology whose characteristics are shown in Table 2.1 VTEAM MODEL parameters has been investigated. Memristor writing voltages and biasing times need to be adjusted. Writing is slower but reading is faster than the 1T1M cell.

4.2.2 Technology Improvements

So far in the analysis we have assumed that the transistor in a transistor-memristor cell does not make a contribution to the Ohm's Law single-bit multiplication. In other words,

these transistors are assumed to be ideal switches with zero resistance in the ON state and infinitely-large resistance in the OFF state. These assumptions are unrealistic and it is possible for the resistance value of the transistor to affect both the writing and reading modes of a transistor-memristor cell.

During the reading mode, for computational correctness, Eq. (4.1) needs to be as close to reality as possible. This requires the following to be true:

$$\begin{aligned} R_{T_{ON}} &<< \frac{1}{G}, \ \ and \\ R_{out} &<< \frac{1}{225 \times G}, \end{aligned} \tag{4.6}$$

where G is the memductance of single memristor, $R_{T_{ON}}$ is the source-to-drain resistance of the cell transistor in its ON state, and R_{out} is the output resistance of the multiplier. When these inequalities are true, the voltage-current relationship depends on the memristors, not the transistors or the output resistor. This means that Eq. (4.1) is approximately true and the design is appropriate.

Unlike for the case of using CMs to control the bit significance weighting, where CM size tuning is obligatory, there is no logical/functional requirement to size transistors in the 1TxM multiplier according to where they are located. For component standardisation, we take advantage of this fact and do not employ transistors of different sizes in our 1TxM multiplier design. This means that when there are a comparative large number of memristors in parallel in a cell, the transistor's resistance becomes more significant and affect the accuracy of the cell's multiplication. To offset this, the transistor size is determined by the worst-case scenario, i.e., appropriate for cell $\{3, 6\}$ located at the far left edge of the crossbar. This in turn leads to using comparatively large transistors across the multiplier, with negative implications on speed, current, leakage and general energy consumption.

Instead of TiO_2 memristors, using Cu:ZnO memristors addresses many of these concerns. For instance, Cu:ZnO memristors have much higher resistance in reading mode compared with TiO_2 memristors (smaller G_{low} and G_{high}). This allows the use of comparatively smaller switch transistors with higher resistances. The differences in the other parameters also lead to large improvements in writing speed and some improvements in reading speed. The speed improvements can be observed by comparing

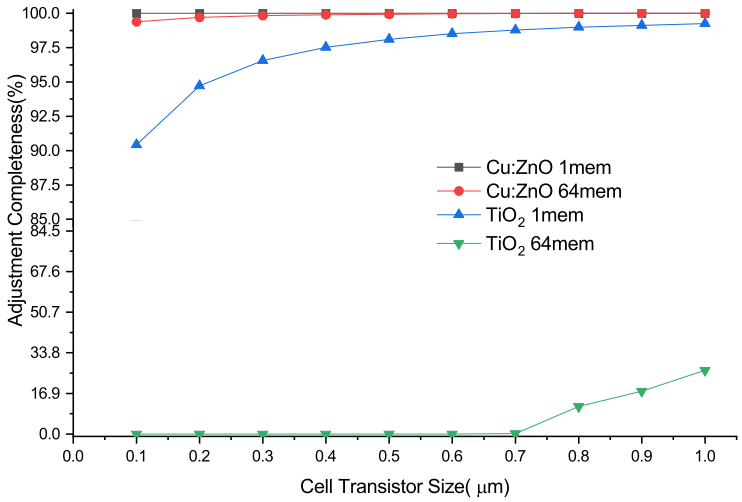
Fig. 4.8(c) to Fig. 4.8(d) and Fig. 4.9(c) to Fig. 4.9(d).

With the parallel memristors in cells driven through a single transistor of fixed size, the writing speed of 1TxM cannot compete with that of 1T1M. With Cu:ZnO technology the reading speed is improved significantly. This means that these multipliers are well used in cases where the operands are not operationally symmetrical. In other words, these target cases have one operand that does not change frequently, which can be represented by memductances, and another operand that changes frequently, which is the best represented by voltages. This asymmetry, together with the asymmetry in operand non-volatility, exactly matches applications in IoT edge nodes and NN neurons.

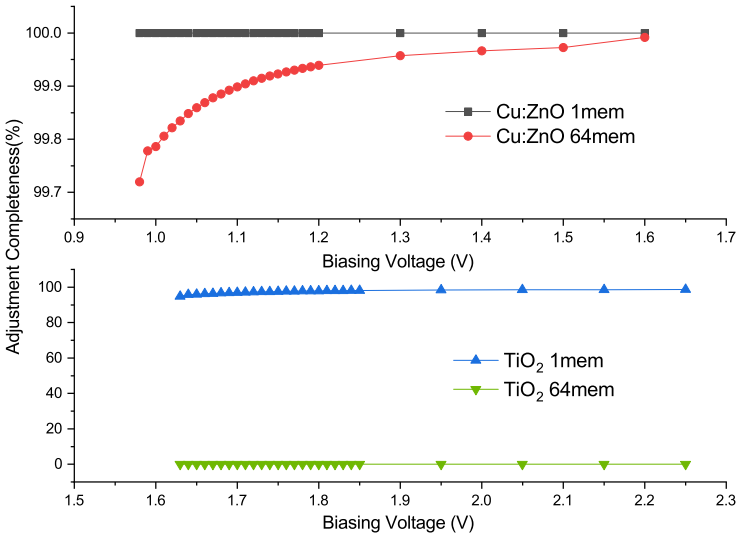
4.3 Simulation Results

The results of simulation studies concerning writing mode are shown in Fig. 4.7(a) and Fig. 4.7(b). In these experiments, the cells are set in writing mode and the memristors have their G values start in the maximum of their respective ranges, corresponding to the R_{ON} values found in Table 2.1. The writing action attempts to adjust these values to the minimum of their respective ranges, corresponding to the R_{OFF} values found in Table 2.1. The experiments are run for long enough time when G stabilises to a value G_{end} which is checked to find how much of the range between R_{ON} and R_{OFF} has been completed in this writing action. This is called adjustment completeness and is shown in percentage points in Fig. 4.7. The ideal result should be 100%, but it can be seen that with the TiO_2 technology, adjustment completion is very low with the largest cell size in the 4-bit 1TxM multiplier (i.e., the 1T64M cell at intersection $\{3, 6\}$).

In general, the larger the transistor size, the higher adjustment completeness can be achieved for the same biasing voltage, and the higher the biasing voltage, the higher adjustment completeness can be achieved for the same transistor size. These trends follow intuition. It is worth noting that TiO_2 technology is inferior to Cu:ZnO in most writing cases, except for writing 0 in 1T1M and 2x and 64x configurations, where TiO_2 is slightly better. This is because the non-ideal conductance variation of memristor in these cell leads the voltage drop on them also varied in the same way. Conversely, the non-ideal varied terminal voltage increases the non-ideal conductance variation of memrsitor.



(a)



(b)

Figure 4.7: The behaviour of the 1TxM cell. In (a), biasing voltages are set as V_{TiO_2} = 1.85 V and $V_{Cu:ZnO}$ = 1.2 V, and length of transistor in cell is also fixed at 60 nm. In (b), transistor size is fixed at Width/Length = 500 nm/60 nm. For TiO_2 model and Cu:ZnO model, the difference between biasing voltage and threshold voltage are the same.

1178 It is also evident that writing 0 in general takes less time than writing 1 for the 1TxM
1179 multipliers which do not have CM delays. These points can be seen from Fig. 4.8 and

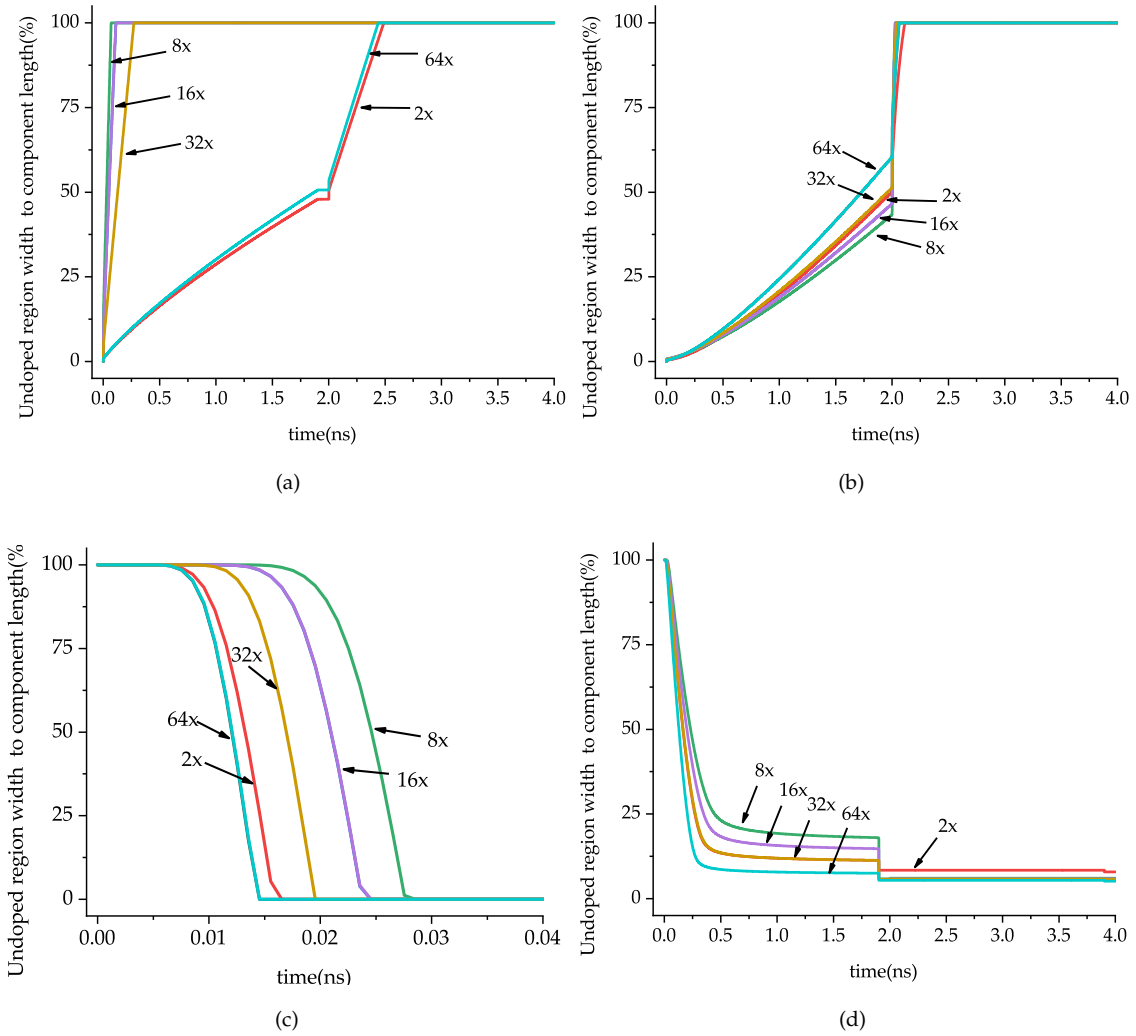


Figure 4.8: The comparison of 1T1M crossbar writing operation. The writing has been presented in (a) to (d), and the amplification ratios are marked with number and "x". (a) presents Cu:ZnO memristor writes 0, (b) presents TiO_2 writes 0, (c) presents Cu:ZnO memristor writes 1, and (d) presents TiO_2 writes 1.

Fig. 4.9.

Since the switching performance of component depends on the worst case scenario, the results of writing operation of 1T1M crossbar in Fig. 4.8 shows that, in writing 0 operation, both Cu:ZnO memristor and TiO_2 memristor have a delay around 2.4 ns.

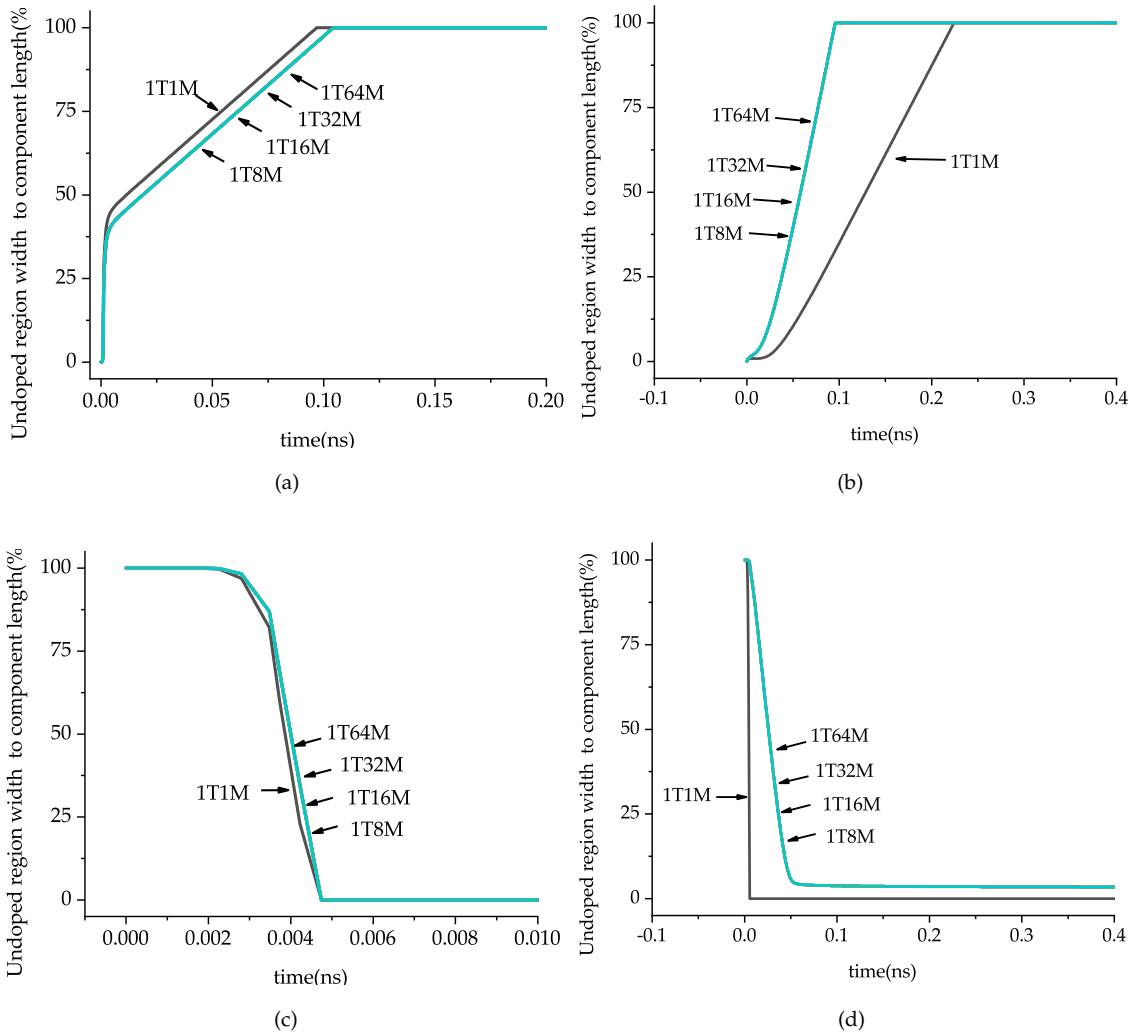


Figure 4.9: The comparison of 1TxM crossbar writing operation. The writing has been presented in (a) to (d). (a) presents Cu:ZnO memristor writes 0, and (b) presents TiO_2 writes 0, (c) presents Cu:ZnO memristor writes 1, and (d) presents TiO_2 writes 1.

In writing 1 operation, Cu:ZnO memristor has a 27ps delay, while TiO_2 memristor has a 1.8 ns delay, which is 67× longer than the Cu:ZnO memristor. Moreover, Cu:ZnO memristor writes memristor to the almost the maximum of device length, while TiO_2 memristor can't achieve the same level of writing at even 1000× more time cost under the same over-threshold biasing potential difference. The results of writing operation of

1TxM crossbar in Fig. 4.9 illustrates that Cu:ZnO memristor performs 2× faster writing 0 operation and almost 1000× faster writing 1 operation than TiO_2 memristor under the same difference between biasing voltages and threshold voltages.

4.3.1 Cell Performance on Crossbar

In this section, we concentrate on the system's capability of maintaining computational correctness with a decent accuracy margin during reading mode. For this, we compare the range of $I_{k,i}$, i.e., the ratio between the high (logic "1") and low (logic "0") current values for memristors in 1TxM cells sited within complete crossbar multipliers, when actual multiplications are being carried out. Four groups of binary multiplication are executed. In decimal values, they are 15×15, 10×5, 5×10, and 15×0. From the observations we find the worst-case scenarios, i.e., when the ratios between logic "1" and logic "0" memductance values reduce the most by the theoretical ratios from the memristor models in Table 2.1.

Note that these experiments are about the reading (computation) mode under the assumption that the correct operand values have been written into the memristors, i.e., any preceding writing operations are correct.

The worst-case data is obtained at the particular memristor with the minimum $I_{k,i_{high}}$ (the lowest observed current value representing logic "1") found across the entire space of all four experiments, and the memristor with the maximum $I_{k,i_{low}}$ (the highest observed current value representing logic "0") found across the same data space. These two worst cases do not involve the same memristor or happen during the same multiplication, but they constitute the worst-case ratio. For each of the TiO_2 and Cu:ZnO technologies, the observed range between the two worst cases is then compared with the specified (ideal) range from the memristor models, as well as checked for compliance with Eq.(4.5).

Different transistor and memristor size combinations are tried and the best case (showing the best/worst-case memristor range) for either technology is selected for comparison.

In the case of TiO_2, the ideal range of memductance adjustment from Table 2.1 is $\frac{R_{OFF}}{R_{ON}} = 300$. The worst case $\min\{I_{k,i_{high}}\}$ loses 2% from the top of the range and the worst

case $\max\{I_{k,i_{low}}\}$ loses 0.5% from the bottom of the range. The observed range is 293, reduced from 300 by 2.5%.

Investigating the case for Cu:ZnO in the same way, we find the observed range to be 997, reduced from 1000 by 0.26%.

From these observations it can be seen that, with appropriate transistor and memristor sizing, it is possible to limit accuracy margin reductions from the ideal cases during implementation. The accuracy requirement for a ratio of 225 between the lowest logic "1" current and the highest logic "0" current for a 4-bit multiplier given in Eq. (4.5) can still be satisfied. It is worth noting that with the best design implementations, TiO_2 loses more accuracy margin than Cu:ZnO, which has a much larger margin to begin with.

4.3.2 Case Experimental Study: 4-bit Multiplier

The multipliers presented in this chapter are studied in more detail through analogue simulations in Cadence and compared with relevant existing work. The main reference work featured in these comparisons come from a body of research reported in [23], [86–88] and [89]. The entirely novel nature of proposed pure digital-in, pure analogue-out multiplication scheme, to the best of author's knowledge, has no competing designs solving the exact same problem. Therefore, the most reasonable comparisons performed here are with low-power, low precision MDAC implementations which is mixed digital- and analogue-in, pure analogue-out [23], and with memristor-based full digital multipliers [86, 88, 89]. Since this chapter presents the first work on full DI/AO multipliers, these citations represent the closest related methods available for comparison.

For comparison fairness, we re-implement the existing work and our multipliers using the same technology (65 nm UMC) in 4-bit resolution, and compare the results obtained from simulations in the same environment (Cadence Virtuoso) under the same operating conditions. Our re-implementations of existing work tend to perform better than reported in their original papers, because we include optimisations such as transistor size explorations with the best results selected to feature in the comparison. For the memristor technologies, the VTEAM model used is the same across the entire comparison.

The results reported here are obtained by using the multipliers on a number of different combinations of operand values from 0×0 up to 15×15. The entire set of input data values across the 4-bit range [0, 15] is explored to ensure the numerical correctness of the compared circuits.

4.3.3 Results and Comparisons

In order to compare total multiplication speed, the source of delays from all multipliers in the comparison are analysed.

In one multiplication cycle, our 1T1M and 1TxM crossbar multipliers complete two phases of work: writing, during which the operand values are copied to the memductance values in the transistor-memristor cells, and reading, during which cell, column and full system currents are generated to produce the result of the multiplication. The multiplication operand for memristor on crossbar is in the form of a diagonal vector. Meanwhile GL signals can control the transistors in cells on an entire column simultaneously but not multiple columns at a time because the memductance values along the same row are not always the same. This means that the writing operation is normally processed one diagonal at a time. For an $N \times N$ multiplier, there are N steps in the writing procedure. The reading or multiplication procedure costs only a single step. Consequently, both the 1T1M and 1TxM crossbar multipliers cost $N + 1$ steps per multiplication. For the 1TxM crossbar multiplier, the delay of the last step, the reading phase, between input ready and output current stable, is shown as zero in Cadence because of the resistive Ohm's Law and KCL. For the 1T1M crossbar multiplier, additional delays are incurred from the CM circuits during the reading phase. According to [86], the existing multipliers in the comparison all require more discrete steps for each complete round of multiplication. The smallest number of steps ($2N$) is required by the MAD gate version of Shift-and-Add multiplier.

The theoretical number of discrete steps needed is regarded as a main technology-independent criterion by the authors of [23] and [86], but equally important is the time required to complete a multiplication. Our experiments include full-multiplier execution runs whose latency values are recorded. For each multiplier, both the writing and reading delays are reported in two types. The writing delays are overwriting an existing

0 with 1 and overwriting an existing 1 with 0 in each cell. The reading delays are fast and slow cases depending on the input data values. These delays are reported in Table 4.2, in addition to the required number of steps.

Note that we do not compare with multiplier methods based on reducing the partial product additions using tree structures, such as Wallace and Dadda trees. This is because these methods are not relevant for low-precision multipliers and our methods are not relevant for high-precision multipliers. Therefore, to make a fair DI/AO comparison with the work in [86], we have included a high-performance DAC [90]. Assuming the conversion is completed in one clock cycle, the delay and power of DAC are 0.625 ns and 40 mW, respectively.

It can be seen from the results in Fig. 4.8, and Fig. 4.9 that the CM circuits incur significant additional delays, which strengthens the case for eliminating them by moving from 1T1M to 1TxM cells.

For MDAC operation, writing operation is irrelevant as either the reference or the incoming data is assumed to be constant. As a result, if our proposed multipliers are used in MDAC mode, the writing of memristances happens only once when setting the reference or incoming constant data, and this delay is shared across many multiplication cycles and per-cycle writing delay becomes negligible. This is why writing delay is not included for the MDAC in [23] in the comparison.

Our 1TxM multiplier using Cu:ZnO technology is 100—300$\times$ faster than the memrister-based digital multipliers in [86] in multiplier mode, and faster than the low-power MDAC in [23] in MDAC mode because of the latter's reading delays.

The next metric studied and compared is the numbers of transistors and memristors required by each multiplier design. This is hardware complexity by component count. These metrics are reported in Table 4.3. As can be seen, the proposed 1TxM approach uses the smallest number of transistors and the greatest number of memristors in the 4-bit case. Compared to the 1T1M cell, adding parallel memristors does not increase writing latency per cell. However, the elimination of the CMs reduces the full-multiplier writing latency significantly.

Peak power dissipation is studied next. The recorded power typically fluctuates during each multiplication round, and here the maximum power value recorded during

Table 4.2: Multiplier Operation Steps and Delay per Multiplication

Multiplier	Steps	4-bit Writing		4-bit Reading		The Worst Case
		Write Logic ″1″	Write Logic ″0″	Fast Case 15×0	Slow Case 15×15	Total Delay
1T1M (TiO_2) (This Work)	$N+1$ (CM Delay Exists)	1.87 ns	2.2 ns	36 ns	36 ns	38.2 ns
1TxM (TiO_2) (This Work)	$N+1$	0.05 ns	0.23 ns	**0**	**0**	0.23 ns
1TxM (Cu:ZnO) (This Work)	$N+1$	**4.7 ps**	**0.1 ns**	**0**	**0**	**0.1ns**
Shift-and-Add (IMPLY Logic) [86]	N^2+N	14.9 ps	9.75 ns	0.67 ns	2.33 ns	12.13 ns
Shift-and-Add (MAD) [86]	$2N$	31.0 ns	29.0 ns	0.68 ns	1.18 ns	32.23 ns
MDAC [23]	**1**	N/A	N/A	10.3 ps	0.816 ns	0.816 ns

each of the writing and reading phases are reported. For this comparison, the best-performing multiplier designs from [23], [86] and [89] are compared with the best-performing multiplier design presented in this chapter, the 1TxM multiplier based on Cu:ZnO memristors. The results are given in Table 4.4. As expected, our best multiplier returns competitive power figures when operating in multiplier mode. When operating in MDAC mode, the writing power dissipation is negligible because a single write is shared by many multiplication cycles.

Our best multiplier is worse in peak power than the IMPLY multiplier in [86] for

Table 4.3: Circuit Complexity of Memristor Based Multipliers

Multiplier	Memristor	Transistor	4-bit Multiplier Complexity
1T1M (TiO_2) (This Work)	N^2	$N^2 + 4(2^N - 1)$	16 Memristors, 76 MOSFETs
1TxM (Cu:ZnO) (This Work)	$(2^N - 1)^2$	N^2	225 Memristors, **16 MOSFETs**
Shift-and-Add (IMPLY Logic) [86]	$7N + 1$	$15N - 1$	29 Memristors, 59 MOSFETs
Shift-and-Add (MAD Logic) [86, 88]	$5N$	$17N + 2$	20 Memristors, 70 MOSFETs
MDAC [23]	0	$2N + 13$	**0 Memristors**, 21 MOSFETs
Array [65]	$7N^2 - 8N + 9$	$132N + 6$	89 Memristors, 534 MOSFETs
IMPLY Semi-Serial Adder [65]	$2N^2 + N + 2$	$2N^2 + \frac{5N}{2} + 3(N \geq 2)$	38 Memristors, 45 MOSFETs

writing 0, but because the writing 0 delay is only slightly more than 1% of that required by the IMPLY multiplier (See Table 4.2), the energy consumed for writing 0 is much smaller. Because our 1TxM multiplier with Cu:ZnO memristors has negligible reading delay, reading power only matters in the sense that it should not peak too high for the sake of safety and longevity. In this case the peak reading powers stay competitive with the compared designs.

The last metric compared is energy consumption per multiplication, with the results reported in Table 4.5. A "multiplication" here refers to an entire cycle including the writing and reading phases and the energy figures are obtained through integrating power over time across the entire operation. For the multipliers based on memristor

technology, the memristors start with digital 0 in the initial state before writing. This arbitrary choice of initial state does not favour any method, but does result in some cases of zero energy being recorded as nothing happens (product directly available) in some of the multipliers in those cases. Our 1TxM multiplier with Cu:ZnO memristors return the best-in-comparison figures in all experiments, with orders of magnitude improvements over the compared designs.

Table 4.4: Multiplier Peak Power per Phase

Multiplier	4-bit Writing Power		4-bit Reading Power	
	Write Logic "1"	Write Logic "0"	15×0	15×15
1TxM (Cu:ZnO) (This Work)	**8.40 μW**	270 μW	**0.67 μW**	655 μW
Shift-and-Add (IMPLY) [86]	656 μW	**98.1 μW**	40.1 mW	44.1 mW
Shift-and-Add (MAD) [86]	732 μW	1.52 W	40.6 mW	40.4 mW
MDAC [23]	N/A	N/A	98.4 μW	**489 μW**

Table 4.5: Energy per Multiplication Corner Cases

Multiplier	Energy Consumption					
	0×0	0×15	15×0	7×8	8×7	15×15
1TxM (Cu:ZnO) (This Work)	**0**	**0.158 aJ**	**0**	**0.118 aJ**	**0.039 aJ**	**0.158 aJ**
Shift-and-Add (IMPLY) [86]	25 pJ	107.72 pJ	26.59 pJ	108.18 pJ	76.71 pJ	127.8 pJ
Shift-and-Add (MAD) [86]	25.04 pJ	413.3 pJ	25.11 pJ	891.9 pJ	3.15 nJ	413.4 pJ
MDAC [23]	1.407 fJ	196.4 aJ	30.96 fJ	105.3 fJ	110.4 fJ	214.3 fJ

In principle, the reference items from [86] and similar work are memristor multipliers which are based on conventional CMOS digital multiplier principles including oblig-

atory carry-passing and/or sequential operations with more steps. Implementation-wise they also require a substantial amount of switching logic compared to the number of memristors used, leading to worse area, power and speed. In comparison, by representing data in three different physical quantities, our designs leverage laws of physics such as Ohm's Law and KCL for naturally parallel operations across resistive elements, with much-reduced memristor writing operations and virtually delay-free reading, saving delay, power and energy costs.

4.4 Summary

In this chapter, novel multiplier designs that make use of transistor-memristor cells for bit-wise multiplication are presented. Working in mixed-signal mode, these designs remove the need for carry-to-the-left operations in conventional digital multipliers, and directly provide an analogue output. The elimination of carry propagation and DAC circuits, whilst maintaining digital input interfaces, is important in edge computing because this allows the majority of the computation to remain digital, with its associated advantages, but produces the required analogue output directly. The designs take advantage of the substantial margin of memristance differences between the ON and OFF states of a memristor. High and low analogue current values with large separation conveniently represent logic ”1” and ”0” and provide sufficient accuracy for analogue-out multipliers.

The multiplication is performed by mapping one of the operands to memductance values. With non-volatile memristors as the core in-memory compute units, the multipliers benefit from intrinsic data retention in a number of scenarios. These include when an input variable is multiplied by a constant coefficient, or when a variable number is multiplied by a relatively constant reference, or when a fixed number is multiplied by a variable reference, which are frequently seen in control, signal processing, AI and MDAC applications.

By using multiple memristors in parallel in each cell, we relocate the bit significance weighting function from CMs to the number of memristors in a cell. It is clear to see that TiO_2 memristor decreases nearly 90% writing 0 time cost and 97% writing 1 time cost

in one multiplication. In other words, the same proportions of energy consumption are also saved from the multiplication. Also, this allows the proposed 1TxM based multiplier with Cu:ZnO to outperform recently reported designs in terms of hardware complexity, performance and energy while staying competitive on peak power. Moreover, the Cu:ZnO memristor itself also performs better energy efficiency and speed in crossbar multiplier than TiO_2 memristor, and most existing work is based on the TiO_2 memristor.

However, these advantages come at the cost of limits in the memductance adjustment range, which affect the large scale implementations beyond a 4-bit multiplier, which is nonetheless sufficient for many micro-edge applications [91]. In our future work, the input/output characteristics of our 1TxM multiplier will be modelled as a perceptron to design a new machine learning accelerator. Other resistive memory units can also be investigated in similar memristor architectures.

Chapter 5

Memristive Multiply-accumulate Unit for Neural Networks

AI applications implemented using NNs require extensive arithmetic capabilities through MAC units. Their designs based on traditional voltage-mode circuits feature complex logic chains in both the multiplication as well as accumulation operations. Additionally, as the data loading and storage operations are performed using a separate memory block (i.e., using Von Neumann architecture), each data movement incurs further on-chip communication bottlenecks. The effect of these manifests in terms of high latency and power consumption for MACs in hardware acceleration. MACs for NNs require both digital inputs and digital outputs. This is a major difference between the work reported in this chapter and that of the previous chapters.

Many modern applications, such as neuromorphic, signal processing and control, require the multiplier output in an analogue form with digital input interfaces [18]. This is conventionally satisfied by attaching a digital-to-analogue conversion (DAC) device to the output of a digital circuit [20]. However, DAC circuits add to the energy and performance costs that depend on the precision of the digital multipliers.

Multiplication with mixed-signal arithmetic circuits is a potential alternative [22] and has a successful academic and commercial history. An example is the multiplying DAC

(MDAC) circuit, which multiplies a digital number by a usually analogue reference signal to produce an analogue output [23]. Digital-in analogue-out, where both operands are digital but the product is analogue, has remained under-explored. This chapter addresses this problem. Table 1.1 lists different types of multipliers (including MDACs) by the digital and analogue nature of their input and output signals.

This chapter presents a novel MAC unit based on a single-bit multiplication cell (SBMC) consisting of a number of memristors and transistor switches which is called multiple transistors multiple memristors (yTxM), a development from the 1TxM cell structure presented in Chapter 4. Apart from this difference, the crossbar multiplier structure remains unchanged from the previous chapters.

The input voltage and memristor self-conductance represent the multiplication operands, and the current represents the product, according to Ohm's Law. In the arithmetic design from Chapter 4, the multiplication operands are expressed in the mixed form (voltage, conductance, and current) for achieving better calculating speed and energy efficiency. Single-bit multiplication comes naturally with Ohm's law, with voltage input representing one operand, conductance the other, and current the product. In addition, KCL takes care of the addition operations. With KCL, addition and subtraction are equivalent to joining multiple current paths into a node and removing current paths from a node. Most of these design aspects of the crossbar multiplier are maintained in this chapter, up to the analogue product represented by the current sum, with the only extension being the yTxM cell structure, which provides more fine-tuning capabilities for matching with the additional circuits required to convert the analogue product to digital form.

When converting the intermediate current product conventional voltage-encoded multi-bit digital format, targeting multi-MAC applications such as NNs, there is a built-in bit-precision reduction that makes the output the same bit resolution as the inputs. This is unlike typical digital multipliers, which have double the number of bits in their products compared with the operands. This helps keep the precision of multi-layer NNs constant and energy and latency under control.

Our MAC unit consists of the memristor-transistor crossbar multiplier and mixed-signal flash ADC shown in Fig. 5.1. This chapter introduces the main parts of this MAC

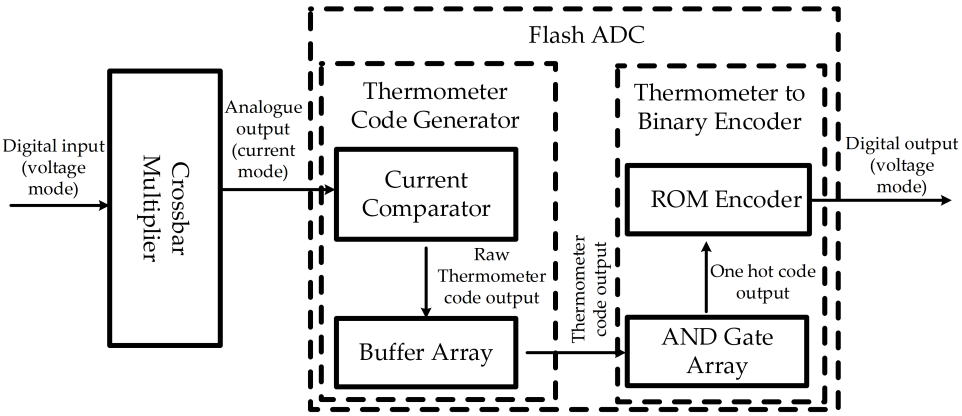


Figure 5.1: The structure of MAC units.

unit.

For this design, the computation latency consists of memory writing and result encoding operations, with the Ohm's Law and KCL operations contributing negligible delay. This is because the crossbar structure eliminates the need to deal with carries. When compared with other memristor-based multipliers in UMC 65-nm technology, the proposed work shows an order of magnitude improvement in latency in 4-bit implementations. In addition, the energy consumption per multiplication cycle of the proposed work is shown to improve by up to 92%. To investigate the usefulness of this MAC design in machine learning applications, its input and output relationship has been characterised to represent a 4-bit input perceptron which is then replicated to demonstrate multi-layer perceptrons (MLPs) to classify the well-known dataset of hand-written digits, modified national institute of standards and technology (MNIST). This case study implements a hyper-parameter search to find configurations of the MLP that lead to high accuracy for this classification problem.

5.1 Multiple-transistor Multiple-memristor Multiplier

5.1.1 Resistive Multiple Memristors Multiplication Cell

Taking advantage of memristor resistivity, the resistive xM cell can perform amplification by adjusting the cell R_M for the target operand. The most straightforward method is to keep single-memristor resistances the same across the multiplier, but build 1TxM cells with different numbers (x values) of parallel memristors corresponding to their bit significances. For example, we use 1M for bit 0, 2M for bit 1, 4M for bit 2, 8M for bit 3, etc. In this way, the cells perform the required current amplification, removing the need for CMs. When applied to the crossbar architecture, both 1M and xM cells help reduce the energy cost and latency. Additionally, the space cost of multipliers based on these cells can also be lower [18].

This type of mixed-signal multiplier is DI/AO. Because transistor switching only happens when setting the memristor values and connecting the input voltages, the only delay is associated with making the operands (multiplier and multiplicand) ready. After that, the multiplication operation itself only involves resistive Ohm's Law which together with KCL can be regarded as instantaneous. This means that the product is immediately obtained once the operands are ready. This compares to regular digital schemes that have to go through multi-stage addition and carry-handling operations once the bit products appear.

Another advantage of such transistor-memristor crossbar multipliers is that one of the operands is represented by memductance $G_M = \frac{1}{R_M}$, which is non-volatile. This is a good match for applications such as NNs and reference-based arithmetic where one of the operands tends to be relatively stable and requires only sporadic change. For multi-stage operations such as NNs, a digital-in/digital-out (DI/DO) MAC unit is required. If this type of mixed-signal multiplier is to be used, additional circuits are needed to generate the appropriate digital output from the intermediate current that encodes the product.

Fig. 5.2 represents the SBMC. The serial connection of multiple memristors (xM) and multiple transistors (yT) generates the basic multiplication cell in the proposed

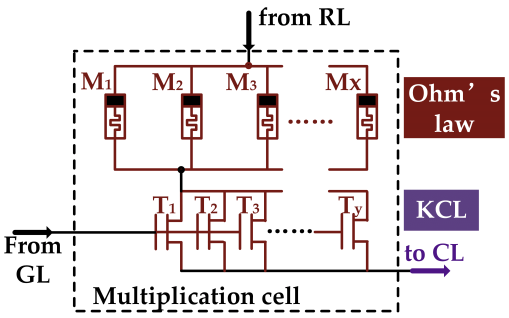


Figure 5.2: The structure of yTxM multiplication cell.

multiplier.

A memristor can be set in two interchangeable states: a high conductance state (HCS) and a low conductance state (LCS). These two states are used to represent the value on one of the two single-bit operands (inputs). When providing/preparing the value of this operand, the cell works in writing mode, with the input voltage used to write either a HCS or an LCS into the memristor. After this operand is set, the cell can work in reading mode, which is the multiplication operation. In reading mode, the input voltage takes the value of the other operand and is in either of the two states: the high voltage state (HVS) or low voltage state (LVS). The cell current then forms the output (product) of the single-bit multiplication according to Ohm's law, and is also in Boolean format with high and low states. The transistors additionally serves the purpose of turning the cell off (not-writing and not-reading, but holding the operand encoded in the memductance state).

Therefore, the operation of the multiplication cell can be easily used to encode Boolean logic: HCS and HVS represent logic "1", while LCS and LVS represent logic "0". Similarly, the output current also has high and low states that can encode logic "1" and logic "0". In this way, a memristor-transistor cell can perform single bit multiplication (same as logic AND).

5.1.2 Crossbar Multiplier Current Identification

SBMCs (shown in Fig. 5.2) are then composed into a multi-bit multiplier using a crossbar structure, with KCL taking charge of the partial product addition step. A 4-bit case is shown in Fig. 4.5, with the updated part being that the 1TxM MC in Fig. 4.2 is replaced by yTxM MC in this chapter.

In this updated architecture, all SBMCs are included in the Ohm's law zone (enclosed in brown dashed lines). All wires and nodes through which currents flow belong to the KCL zone, enclosed in purple dashed lines. In the KCL zone, nodes "Digit1" to "Digit7" represent partial products whereas the current through the load resistor R_{out} is the final product. Note that, unlike the common long-multiplication algorithm, there is no attempt to find horizontal partial products and no attempt to pass carries horizontally. All partial products are generated vertically. Carries can be avoided because the vertical partial products and the final product are encoded in currents with higher upper limits to their values than that which encodes a single logic "1". In other words, the currents at the Digit1 to Digit7 nodes and I_{out} can take values that are multiples of the high current state across a single memristor, which encodes logic "1" at the lowest level of detail. For instance Digit2's current can be up to four times this single-memristor logic "1" and the maximum value of the partial product at Digit2 is therefore 4 (because each MC_2 can generate twice the maximum current compared with MC_1), instead of 2 in the case of a typical digital multiplier at this bit position.

Because the multiplication is performed by fixed voltage values for 0 and 1 from the voltage operand, the output currents of cells in each column corresponding to logic "1" at these cells need to be set according to the column's digit significance. Avoiding CM amplifiers, this can be implemented using x memristors in parallel with the appropriate x value. The relationship between x and the digit significance N follows Eq. (5.1):

$$x = 2^{N-1} \tag{5.1}$$

Let us use the 4-bit multiplier in Fig. 4.5 as an example, assuming that the cell transistors are ideal switches, with V_{MH} and V_{ML} as the high voltage and low voltage operand inputs, and R_{MH} and R_{ML} as the high and low cell resistance (memristor

resistance) operand inputs. The possible output current states in each cell are shown in Fig. 5.3 as I_1, I_2, I_3, and I_4. Because the logic "1" state is defined by V_{MH} and R_{ML}, I_4 is the output current representing logic "1", whereas the other three current states I_1, I_2 and I_3 all represent logic "0" because at least one of their input operands encodes 0. Given the cell structure, none of I_1, I_2 and I_3 can be true 0 A. This is because R_{MH} cannot be true infinity and to maintain the commutative property of multiplication, true 0 V should not be used in the voltage input operand either. Because of KCL, a potentially large number of relatively small I_1, I_2 and I_3 values accumulated with the sum still required to represent a product value of 0. In other words, a single I_4 needs to be greater in value than the sum of a large number of I_1, I_2 and I_3 values to differentiate 0 and 1 at the final product.

The final result I_{out} matrix shown in Fig. 5.3 illustrates this issue in detail by enumerating all possible I_{out} values across all possible combinations of input operand values. This current map assumes that the operand encoded in the voltage is called the multiplier and the other operand encoded in the memristor resistance is called the multiplicand, without losing generality. Each operand is 4 bits wide and takes values from 0 to 15. When the multiplier increases from 0 to 15 we move from left to right along the i axis, $0 \leq i \leq 15$, and when the multiplicand increases from 0 to 15 we move from top to bottom along the j axis, $0 \leq j \leq 15$. At each position (i, j) in the matrix, $I_{i,j}$ encodes the product of multiplying (multiplier $= i$) by (multiplicand $= j$). To simplify the presentation, we use four coefficients a, b, c and d to differentiate all the output currents and define $I_{i,j}$ as $I_{i,j} = aI_1 + bI_2 + cI_3 + dI_4$. This means that moving down in the matrix, a decreases and c increases, with b and d held constant, and move right in the matrix, b decreases and d increases, with a and c held constant. Because $15 \times 15 = 225$, $a + b + c + d = 225$. The four corner cases of the matrix are therefore $I_{out} = 225I_1$, $I_{out} = 225I_2$, $I_{out} = 225I_3$, indicating final product values of $0 = 0 \times 0 = 0 \times 15 = 15 \times 0$, and $I_{out} = 225I_4$ which indicates a final product value of $225 = 15 \times 15$.

For the 4-bit crossbar multiplier shown in Fig. 5.3, the coefficients a, b, c and d are related to the operand values i and j according to Eq. (5.2) – Eq. (5.5).

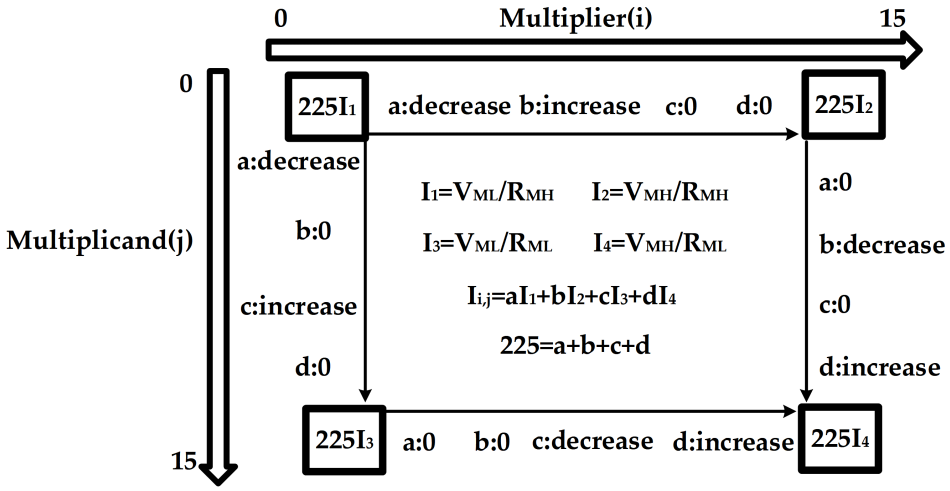


Figure 5.3: The mapping of all multiplication output current.

$$a = i * j - (2^4 - 1)i - (2^4 - 1)j + (2^4 - 1)^2 \tag{5.2}$$

$$b = (2^4 - 1)i - i * j \tag{5.3}$$

$$c = (2^4 - 1)j - i * j \tag{5.4}$$

$$d = i * j \tag{5.5}$$

For a general $N \times N$-bit multiplier, the equations above are replaced by Eq. (5.6) – Eq. (5.9), where $0 \leq i \leq (2^N - 1)$ and $0 \leq j \leq (2^N - 1)$.

$$a = i * j - (2^N - 1)i - (2^N - 1)j + (2^N - 1)^2 \tag{5.6}$$

$$b = (2^N - 1)i - i * j \tag{5.7}$$

$$c = (2^N - 1)j - i * j \tag{5.8}$$

$$d = i * j \tag{5.9}$$

From these, the output current for position (i, j) in the result current matrix can be found according to Eq. (5.10).

$$I_{i,j} = [i*j - (2^N-1)i - (2^N-1)j + (2^N-1)^2]I_1 + [(2^N-1)i - i*j]I_2 + [(2^N-1)j - i*j]I_3 + (i*j)I_4 \quad (5.10)$$

Assuming a base voltage $V_0 \neq 0$ and base resistance $R_0 \neq 0$, we can relate the high and low memristor voltages and resistances to these base values as in Eq. (5.11):

$$V_{MH} = \alpha V_0 \; V_{ML} = \beta V_0 \; R_{MH} = \gamma R_0 \; R_{ML} = \lambda R_0 \quad (\alpha > \beta > 0, \; \gamma > \lambda > 0) \quad (5.11)$$

Then, the base current $I_0 = V_0/R_0$ can be substituted into I_1 – I_4, resulting in Eq. (5.12) – Eq. (5.15).

$$I_1 = \frac{V_{ML}}{R_{MH}} = \frac{\beta}{\gamma} I_0 \quad (5.12)$$

$$I_2 = \frac{V_{MH}}{R_{MH}} = \frac{\alpha}{\gamma} I_0 \quad (5.13)$$

$$I_3 = \frac{V_{ML}}{R_{ML}} = \frac{\beta}{\lambda} I_0 \quad (5.14)$$

$$I_4 = \frac{V_{MH}}{R_{ML}} = \frac{\alpha}{\lambda} I_0 \quad (5.15)$$

Substituting Eq. (5.12) – Eq. (5.15) into Eq. (5.10) and simplifying the result, we obtain Eq. (5.16).

$$I_{i,j} = \frac{i*j(\alpha-\beta)(\gamma-\lambda)}{\gamma\lambda} I_0 + (2^N-1)\left[(2^N-1)\frac{\beta}{\gamma} + \left(\frac{\alpha}{\gamma}i + \frac{\beta}{\lambda}j\right) - \frac{\beta}{\gamma}(i+j)\right] I_0 \quad (5.16)$$

It is evident that the multiplication is commutative if $\frac{\alpha}{\beta} = \frac{\gamma}{\lambda}$. In practice, this is ensured by adjusting the parameters of the cell components to make the contributions of both operands symmetrical and linear.

5.2 Analogue-to-digital Conversion

After the analogue output $I_{i,j}$ is generated, its value needs to be represented as a 4-bit (or N-bit in the general case) digital value either as a memristor resistance or voltage

encoding for the entire MAC unit to function in a multi-MAC NN using copies of the same MAC hardware. Because the memristor resistance values are written in by digital voltage signals, we do not lose generality if a 4-bit MAC outputs a 4-bit voltage encoded product (4 Boolean voltage signals).

We implement this functionality using a flash ADC, designed from components adapted from [35, 37]. The choice of using thermometer code as an intermediate step comes from the desire to make this MAC approximate in the sense of generating a 4-bit product from input operands that themselves are also 4 bits in width. This ADC consists of a single-action multiple-current comparator, buffer array and a read only memory (ROM) encoder. The following subsection describes this part of the system in detail.

5.2.1 Thermometer Code Generating Current Comparator

Table 5.1: Thermometer Code Generator Transistor Size

Component	Size	Component	Size	Component	Size	Component	Size
M_{in}	3.2 μm	M_{ref}	1.6 μm	P_1	100 nm	P_{11}	80 nm
M_{out}	1.6 μm			P_2	140 nm	P_{12}	100 nm
M_1	100 nm	M_9	715 nm	P_3	80 nm	P_{13}	100 nm
M_2	110 nm	M_{10}	785 nm	P_4	80 nm	P_{14}	100 nm
M_3	310 nm	M_{11}	850 nm	P_5	80 nm	P_{15}	100 nm
M_4	365 nm	M_{12}	965 nm	P_6	80 nm	P_{16}	100 nm
M_5	440 nm	M_{13}	1 μm	P_7	80 nm		
M_6	510 nm	M_{14}	1.11 μm	P_8	80 nm		
M_7	580 nm	M_{15}	1.19 μm	P_9	80 nm		
M_8	650 nm	M_{16}	1.27 μm	P_{10}	80 nm		

Fig. 2.3 represents the current comparator. Given that the digital output is expected to be in 4 bits, the comparator is set to 16-value thermometer code output. The input current is mirrored by a p-type CM that generates a row of pull up current sources; similarly, the reference current is mirrored by an n-type CM that generates a row of pull down current

sinks. By adjusting the size of M_1 to M_N, the reference current can be set to different levels. If a current source has a higher value than the corresponding current sink, the voltage at the junction point is pulled up to Vdd; otherwise, the junction point voltage is pulled down to ground. Therefore, the comparator will generate a thermometer code in the buffer array.

To make this design work for our 4-bit crossbar mixed-signal multiplier, the transistor sizes need to be tuned to fit the multiplier current output characteristics. Details of the MOS transistor size choices are listed in Table 5.1.

5.2.2 Thermometer Code to Binary Encoder

The thermometer code is an intermediary format that, after serving the purpose of fast comparison and product precision adjustment, has to be converted into a voltage binary code for MAC output. The structure of the thermometer to binary encoder is shown in Fig. 2.4. This encoder consists of an AND gate array and a ROM encoder. For a 4-bit digital output, the 16-value thermometer code is first converted by the AND array to a 16-digit one-hot code, which is then fed to the ROM encoder to generate a 4-bit binary output.

The complete MAC unit therefore accepts as inputs a multiplier in the form of 4-bit binary voltage signals and a multiplicand in the form of 4-bit binary memductance values, and generates a product in the form of 4-bit binary voltage signals. This voltage-encoded 4-bit binary number can then be used directly as the multiplier for another MAC of the same configuration, or used to write the multiplicand for it. This means that the digital-to-digital MAC can be instantiated multiple times to form an NN or other machines that require a number of distinct MAC units of the same type working together.

5.3 Neural Network Implementation

This section presents a case study to validate the proposed MAC unit. In this section, a machine learning algorithm (MLA) NN is created using copies of our MAC unit servicing as perceptrons. The machine learning problem solved with this NN is the classification of the MNIST data set.

As our MAC unit supports only 4-bit inputs (integers), we need to apply a quantization technique to preserve the high accuracy while using such low-precision numbers. Two state of the art techniques exist for this: post-training quantization (PTQ) and QAT. The weights used in the PTQ will be quantified to the target bit-width after the floating-point based training. This is a simple technique, yet not suitable for <8-bit resolution applications because of the increase in quantization error [92]. Alternatively, the QAT technique injects the quantization error during training. This allows the lower-resolution NN to learn and improve its weights appropriately. Previously, 98% accuracy for MNIST classification using 4-bit NN with the QAT technique has been shown in [93]. Therefore, this technique will be applied in our NN training.

The most challenging issue in our NN training is that the output of our MAC unit contains variations because of its analogue nature. To overcome this issue, we will use the same idea as QAT; the variations will be included in our training so that the NN can learn these variations and adjust its accuracy accordingly. In summary, this section contributes the QAT technique analysis to inject the MAC unit variations, the demonstrates NN training for MNIST classification and compares the accuracy of our NN trained MAC unit with the basic 4-bit QAT NN. Note that, for ease of computation analysis, our NN consists of fully-connected layers only. We are considering extra software library development to include the proposed MAC unit in the convolution layers as future work.

5.3.1 Quantization-aware Training Analysis

Fundamentally, fully-connected NN computation contains dot-product operations between weight matrices and input vectors. Eq. (5.17) means that the resulting matrix element r_3 at row i and column k is obtained from the sum of the products between the pairs of the weight matrix elements r_1 at row i and the input vector elements r_2 at column k. In general, these variables are presented precisely in floating-point format.

$$r_3^{(i,k)} = \sum_{j=1}^{N} r_1^{(i,j)} r_2^{(j,k)} \tag{5.17}$$

To compute the above equation using integer-arithmetic hardware, we need to

quantify these real numbers. Following [94], any real numbers can be quantified, resulting in positive quantified-values q in integers minus the zero-point Z and scaled by the scale factors S as shown in (5.18). In addition, the range of q is between 0 and 2^{n-1}, where n is the number of bits. Therefore, in this work q is in the range [0, 15] (4-bit unsigned integer).

$$r = S(q - Z) \tag{5.18}$$

Replacing the weights r_1 and inputs r_2 in (5.17) by Eq. (5.18) yields Eq. (5.19) which can be re-written as Eq. (5.20):

$$r_3^{(i,k)} = \sum_{j=1}^{N} S_1 \left(q_1^{(i,j)} - Z_1 \right) S_2 \left(q_2^{(j,k)} - Z_2 \right) \tag{5.19}$$

$$r_3^{(i,k)} = S_1 S_2 \left(N Z_1 Z_2 - Z_1 \sum_{j=1}^{N} q_2^{(j,k)} - Z_2 \sum_{j=1}^{N} q_1^{(i,j)} + \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} \right) \tag{5.20}$$

In Eq. (5.20) there is no dot-product operation on floating-point numbers; this happens only in term $\sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)}$ where both operands are integers, and therefore our multiplier is applicable to this operation.

Another issue is that our MAC unit is centred around an analogue product. It therefore contains a non-ideal effect where its multiplication results deviate from the expected values as shown in Table 5.2. In Eq. (5.21), we add $\sum_{j=1}^{N} C^{\left(q_1^{(i,j)}, q_2^{(j,k)}\right)}$ to sum up the variation from every multiplication. The value of C can be found at column $q_1^{(i,j)}$ and row $q_2^{(j,k)}$ of Table 5.2. This allows the NN to learn and adjust its weights according to our multiplier's numerical characteristics.

$$r_3^{(i,k)} = S_1 S_2 \left(N Z_1 Z_2 - Z_1 \sum_{j=1}^{N} q_2^{(j,k)} - Z_2 \sum_{j=1}^{N} q_1^{(i,j)} + \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} - \sum_{j=1}^{N} C^{\left(q_1^{(i,j)}, q_2^{(j,k)}\right)} \right) \tag{5.21}$$

From Eq. (5.21), we can separate the loss term from the main bracket by multiplying the scale factors S_1 and S_2 as expressed in Eq. (5.22). It can be seen that the large term remains the same as in Eq. (5.20). We can thus conclude that the variation in our MAC unit can be simulated by subtracting the product of both scale factors and the sum of the MAC unit's errors from the basic dot-product's result. Eq. (5.23) will be added to our

Table 5.2: Multiplication Errors of the Proposed Multiply Accumulate Unit

Result		Multiplier															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Multiplicand	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	-1	-1	-1	-1	-1	-2	-2	-1	-1	-1	-2	-2	-2	-2	-2
	2	0	-1	-2	-2	-1	-2	-2	-2	-3	-3	-3	-4	-3	-3	-3	-4
	3	0	-2	-2	-2	-2	-2	-3	-3	-3	-3	-3	-4	-4	-4	-4	-4
	4	0	-2	-2	-2	-3	-3	-3	-3	-4	-4	-3	-4	-4	-5	-4	-4
	5	0	-2	-2	-2	-3	-3	-3	-4	-3	-4	-4	-4	-4	-5	-4	-4
	6	0	-2	-2	-3	-2	-3	-4	-3	-4	-3	-4	-4	-4	-4	-4	-4
	7	0	-3	-2	-3	-3	-3	-3	-4	-3	-4	-3	-4	-3	-4	-3	-4
	8	0	-2	-2	-2	-3	-3	-3	-3	-4	-3	-4	-3	-4	-3	-4	-4
	9	0	-2	-3	-2	-3	-3	-3	-3	-3	-4	-3	-3	-3	-3	-4	-3
	10	0	-2	-3	-3	-2	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
	11	0	-2	-3	-3	-2	-2	-3	-3	-2	-2	-3	-3	-2	-2	-2	-2
	12	0	-2	-2	-3	-3	-2	-2	-2	-3	-2	-2	-2	-2	-2	-2	-1
	13	0	-2	-2	-2	-3	-3	-2	-2	-2	-2	-1	-1	-2	-2	-1	-1
	14	0	-2	-2	-2	-2	-2	-2	-1	-2	-2	-2	-2	-1	-1	-1	0
	15	0	-2	-2	-2	-2	-2	-2	-2	-1	-1	-1	-1	0	0	0	0

training graph as explained in the next section.

$$r_3^{(i,k)} = S_1 S_2 \left(N Z_1 Z_2 - Z_1 \sum_{j=1}^{N} q_2^{(j,k)} - Z_2 \sum_{j=1}^{N} q_1^{(i,j)} + \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} \right) - S_1 S_2 \sum_{j=1}^{N} C^{\left(q_1^{(i,j)}, q_2^{(j,k)}\right)} \tag{5.22}$$

$$r_3^{(i,k)} = \sum_{j=1}^{N} r_1^{(i,j)} r_2^{(j,k)} - S_1 S_2 \sum_{j=1}^{N} C^{\left(q_1^{(i,j)}, q_2^{(j,k)}\right)} \tag{5.23}$$

5.4 Simulation Results

5.4.1 Multiple Transistors Multiple Memristors Multiplication Cell Performance

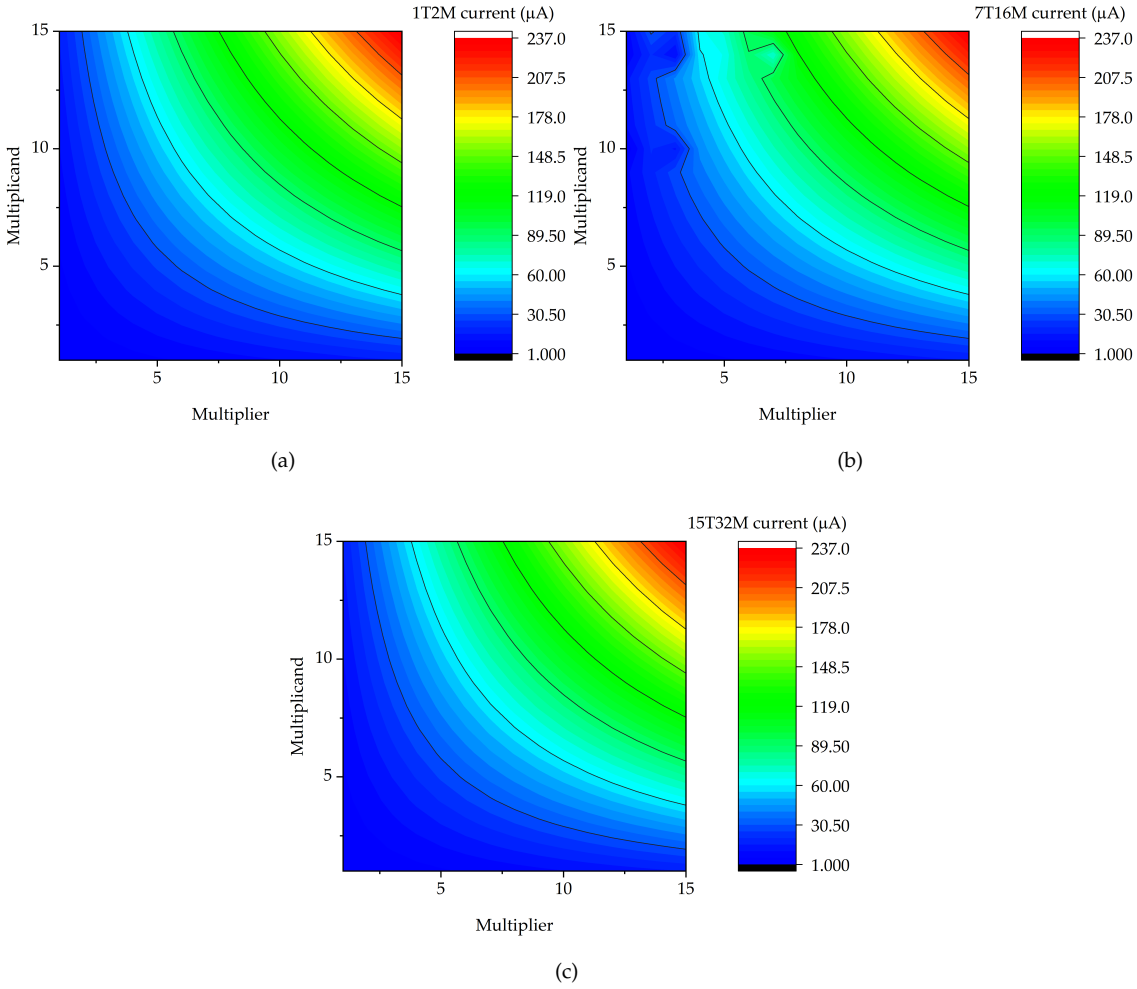


Figure 5.4: The yTxM MC output current mapping in all 4 by 4 multiplications.

The structure of our multiplication cell is presented in Fig. 5.2, the parallel-connected memristors and transistors are marked in brown to indicate them operating under Ohm's law. Similarly, the cell output current path to CL is marked in purple to indicate the

operation of KCL. Because the multiplication cell works as a conductive component on the crossbar, both the memristors and transistors contribute to the cell conductance. It is therefore important to ensure that the memristor dominates the cell conductance, because we use the transistors as (ideal) switches. In other words, even in ON state, the low resistance state transistor still contributes current to multiplication cell. In order to eliminate the transistor effect, memductance should be much larger than the ON state transistor conductance. Additionally, the OFF state transistor conductance should be small enough to isolate a selected cell from the rest of the crossbar so that it can be in holding mode while other cells are written. With the memristor count for each cell determined by the digit significance, the transistor count and size need adjustments to balance that. Therefore, our proposed 4-bit crossbar multiplier uses cells with fixed ratios for the memristor count and transistor count.

In Fig. 5.4 and Fig. 5.5, comparisons between the crossbar with the respective yTxM cell shown. The 4-bit crossbar multiplier generates the same levels of I_{out} with different count transistor-memristor cells, and the product values are symmetric between multiplicand and multiplier indicating commutative multiplication. The 1T2M cell stands out in the error rate comparison. The maximum error rate for the crossbar multiplier is 0.58% with the 1T2M cell, 0.72% with the 7T16M cell, and 0.86% with the 15T32M cell.

Therefore, apart from the LSB using a 1T1M cell, all the multiplication cells in this 4-bit multiplier follow the memristor-transistor ratio for the 1T2M; i.e., two memristors for each transistor in a cell.

5.4.2 Crossbar Multiplier Performance

The 4-bit crossbar multiplier shown in Fig. 4.5 has two operations in each multiplication, writing (operand preparation) and reading (multiplying) operation. When multiplication starts with a new multiplicand, all multiplication cells will be clear to LCS by each RL; then the multiplicand is written by each GL column. Finally, the reading (multiplier) voltages are applied on all RLs. Meanwhile, all cell transistors are switched ON. The multiplication result can be obtained from the ADC out terminal (see Fig. 2.4). When multiplying with an existing multiplicand, the writing step is omitted and the

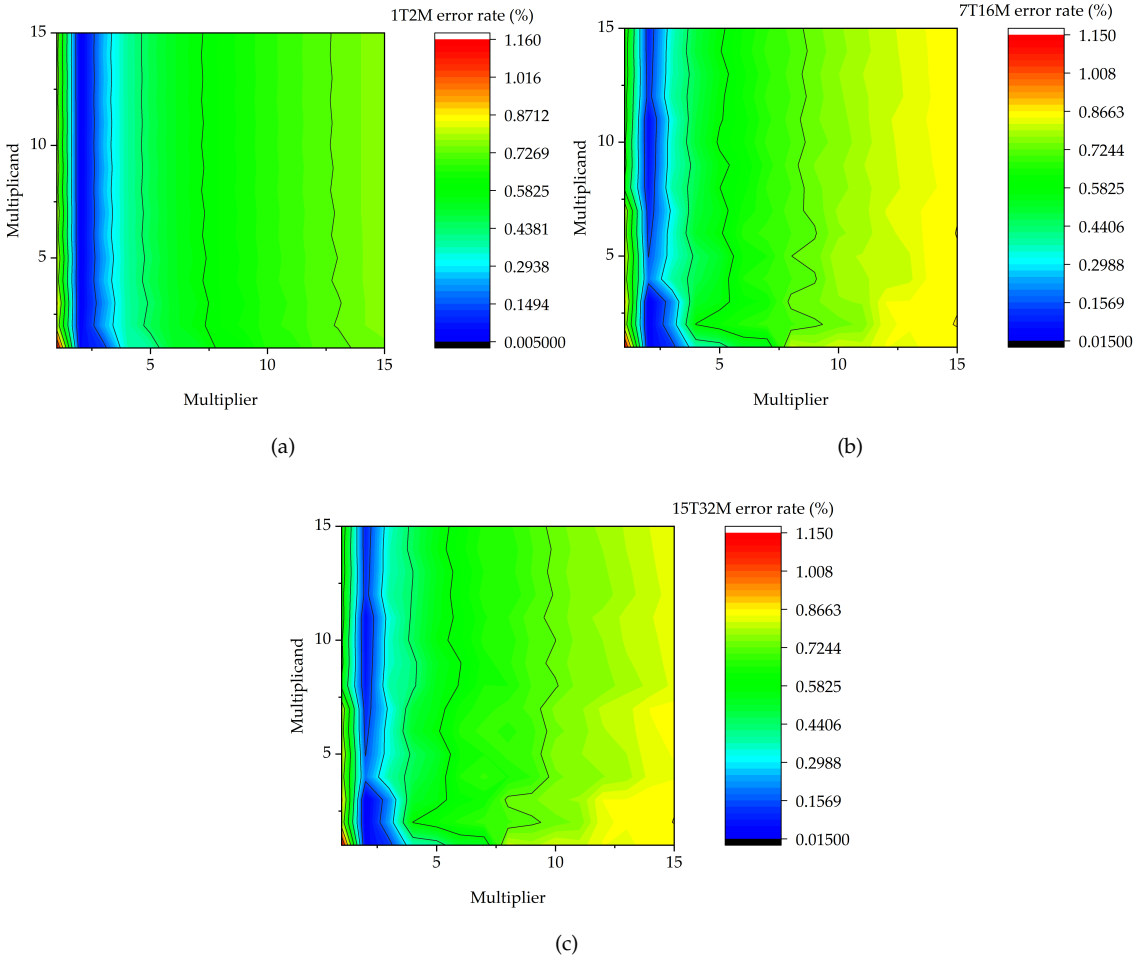


Figure 5.5: The yTxM MC output current error rate mapping in all 4 by 4 multiplications.

reading step starts directly. That is why this multiplier is well suited for asymmetrical multiplication applications such as multiplying variables to coefficient/reference values found in applications such as monitoring and control and certain operations in neural networks where one of the operands (i.e., the multiplicand) does not change too often.

ADC transistor design parameters are presented in Table 5.1 and writing operation setting parameters are presented in Table 5.3. To reduce latency, the writing operations are parallelised on a per-row basis. To match the values of high and low memductance, the reading (multiplier) voltage has values of 0.42 V as logic ”0” and 0.7 V as logic ”1”. The total delay in each multiplication is 2 ns, which is almost entirely ADC delay. Three

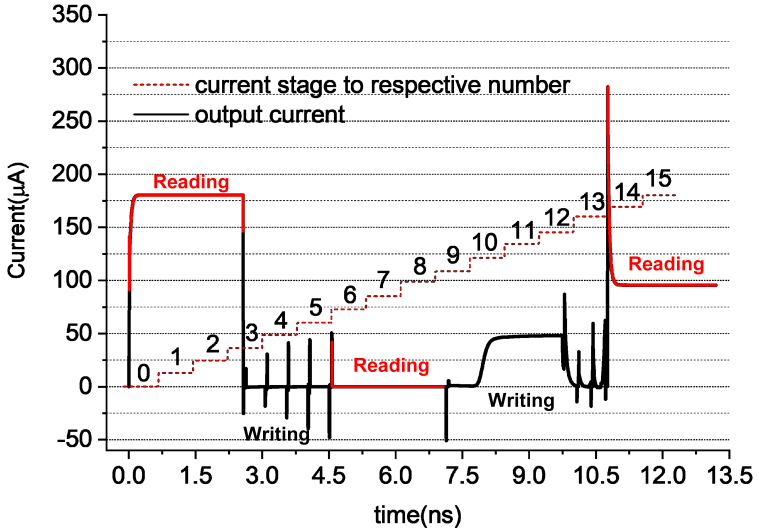


Figure 5.6: The output current details of three multiplication cases. The red dash steps are the threshold for each digital output. 0-2.97 ns is 15×15, 4.57 ns-7.13 ns is 0×0, and 10.77 ns-13.2 ns is 9×6.

Table 5.3: Multiplier Operation Design Details

Area	Time (ns)		Voltage (V)	
Entire Crossbar	Write 1	Write 0	Write 1	Write 0
	0.43	16.9	1.8	-2
Single Row	Write 1	Write 0	Write 1	Write 0
	0.275	0.43	1.8	-2
Single Cell	Write 1	Write 0	Write 1	Write 0
	0.261	/	1.8	/

multiplications, 15×15, 15×0 and 9×6 are tested on the 4-bit multiplier. The results are presented in Figs. 5.6 and 5.7.

The red dashed steps in Fig. 5.6 are the thresholds for the current comparator, which translates currents to thermometer code. For instance, $I_{out} = 100$ μA translates to the thermometer code value of 8, and 9×6 results in $I_{out} \approx 90$ μA, which translates to a

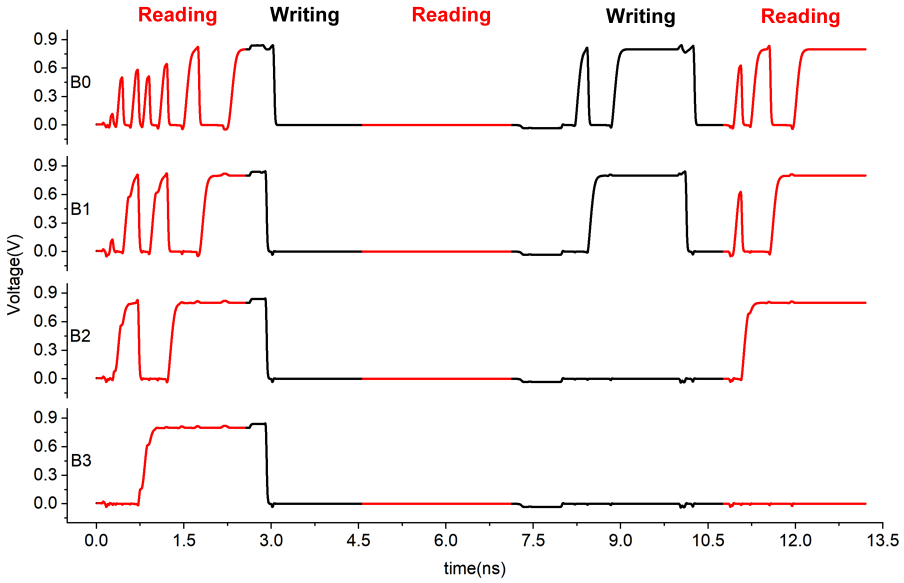


Figure 5.7: The binary pulse output details of three multiplication cases. 0-2.97 ns is 15×15, 4.57 ns-7.13 ns is 0×0, and 10.77 ns-13.2 ns is 9×6.

thermometer code of 7. The output bit voltages are recorded in Fig. 5.7. Here B3 is the MSB and B0 the LSB. It can be seen that the ADC delay is data-dependent and the more bits that are 1 the longer the delay. This is because the less significant bits are settled after the more significant bits, and before then they have swings. The output value of 1111, corresponding to 15×15, takes just less than 2 ns to become stable, which is the worst-case delay for the MAC. In comparison, 0×0 incurs almost no delay.

Value-wise, 15×15 results in 1111 (the largest number possible out of 4 bits), 15×0 results in 0000 and 9×6 results in 0111. These values work well for a 4-bit digital-in and 4-bit digital-out MAC unit.

5.4.3 Energy Efficiency

Our study is mainly based on the worst-case delay assumptions. The worst-case multiplication cycle includes 4 row-writing 0 (reset) operations with a 1.72-ns delay, 4 row-writing 1 (set) operations with a 1.1-ns delay, and one entire crossbar reading (multiply+ADC) operation with a 2-ns delay. The average power is 290 μW. The average

energy consumption per multiplication cycle for the 4-bit 1T2M crossbar multiplier is 1.39 pJ over a 4.82-ns period.

The worst-case energy per multiplication cycle happens with 15×15 because it has the longest delay and the highest I_{out} value (187.3 μA) among all multiplication cases. This worst-case cycle has an energy consumption of 3.91 pJ. The most optimal scenario occurs when computing 0×0, which requires a minimal energy input of approximately 0.01 pJ. This outcome is attributed to the parameter settings, particularly the uniform transistor size on the crossbar of width/length = 500 nm/60 nm. This low energy consumption can be attributed to the insignificant time taken during writing by the crossbar and Analogue-to-Digital Converter (ADC), as well as the low current and voltage values involved in the single multiplication sample. On the other hand, the worst-case reading scenario occurs when computing 15×15, which consumes 0.84 pJ over a period of 2.97 ns.

In Fig. 5.8, the best-case and the worst-case energy consumption figures for our multiplier are compared with state-of-the-art memristor multipliers. The figure shows that the proposed MAC saves 83.7% and 74.1% of energy per multiplication cycle more than the MAD Shift-and-Add multiplier and the optimised MAD Shift-and-Add multiplier and 82.6% per multiplication energy cost than MDAC in their respective worst cases. In the best case, the comparative energy savings can reach up to almost 99%. Even the average energy consumption of the proposed MAC unit, at 1.39 pJ, is significantly lower than the best-case figures achieved by the competition.

5.4.4 Neural Network Training and Results

To demonstrate the application of the proposed MAC unit in our NN training, we constructed three fully-connected layers for MNIST classification as illustrated in Fig. 5.9(a). The numbers of neurons in the input/hidden/output layers were 800/500/10. The forward-pass calculation for each layer follows the graph in Fig. 5.9(b). Regarding the QAT concept, the inputs and weights of each layer were quantified and dis-quantified based on Eq. (5.18) to simulate the quantization error. Note that this procedure is known as fake quantization in the literature [93]. In addition, the resolution q was set to 4 bits, which is consistent with the input resolution of our MAC unit.

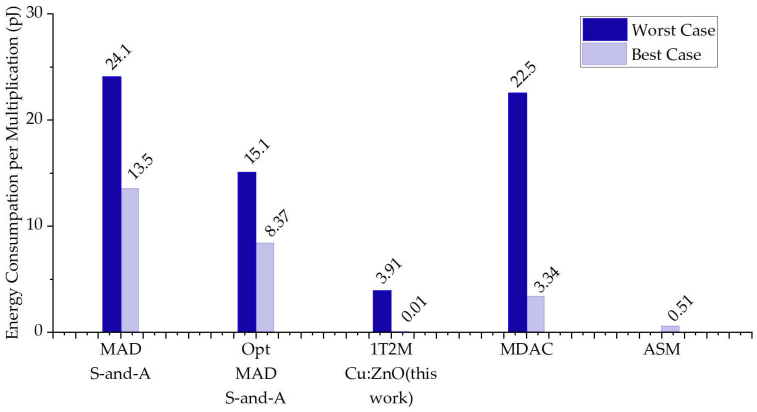


Figure 5.8: The comparison of energy consumption per multiplication with MAD Shift-and-Add multiplier, optimised MAD Shift-and-Add multiplier [86], MDAC [23], and alphabet set multiplier (ASM) [95]. This work consumes the least energy in both worst case and best case. When compared the memristive multiplier [86], the proposed design saves 83.7% and 74.1% energy in the worst case, and saves up to over 99% energy in the best case. When compared with MDAC [23], proposed design still has 82.6% energy cost reduction in worst case and up to over 99% energy saving in the best case. When compared with alphabet set multiplier, the proposed design has 98% energy efficiency advantage in the best case.

Subsequently, we performed the dot-product operation between the inputs and weights, followed by the addition of biases. To ensure output stability, we incorporated a MAC block that subtracted the dot-product results by our MAC's output variations, as elaborated in Section 5.3.1. The resulting values from the MAC block underwent rectified linear unit (ReLU) activation function, and then underwent another round of fake quantization of the activation. The output of this layer was then used as the input for the following layer.

The NN configurations presented in Table 5.4 were implemented using the PyTorch library [96]. The first NN, which served as the baseline, was a 4-bit QAT NN obtained from [93] without any convolution layers. Stochastic gradient descent was employed for the backward pass, while the fake quantization blocks were handled using the straight through estimator. The key parameters were batch size of 64, learning rate of 0.01, and

Table 5.4: Modified National Institute of Standards and Technology (MNIST) Classification Accuracy Comparison

NN Configuration	Training Acc. (%)	Testing Acc. (%)
4-bit QAT NN (Baseline)	93	94
4-bit QAT NN W/O MAC Variation Training	93	30
4-bit QAT NN With MAC Variation Training	89	93
3-bit QAT NN With Precise Multiplier	90	92
2-bit QAT NN With Precise Multiplier	84	86

momentum of 0.5.

To analyse the impact of the MAC unit's output variations, a second NN was trained following the aforementioned procedure, with variations only being injected during the testing phase. Finally, variations were included in both the training and testing phases to evaluate the accuracy improvement. The baseline model produced an accuracy of 94%, indicating only a 4% decline in accuracy compared to the convolutional NN implementation in [93]. This suggests that a pure fully-connected layer is adequate for MNIST classification. Nonetheless, the accuracy drops substantially to 30% in the 4-bit scenario when the MAC unit's impact on the NN training is not simulated, underscoring the importance of MAC unit simulation in the training phase.

After training, the accuracy of the 4-bit scenario was restored to 93% when the NN was trained with the MAC unit's output variations. Moreover, we compared the performance of the proposed work in the NN configuration with lower-precision multipliers. The 2-bit precise scenario had an accuracy of 86%, while the 3-bit precise scenario had an accuracy of 92%. This suggests that the proposed MAC unit is suitable for NN applications, and that variation injection is required during NN training to maintain accuracy. The designed approximate 4-bit multiplier outperformed the 2-bit and 3-bit precise multipliers in terms of accuracy.

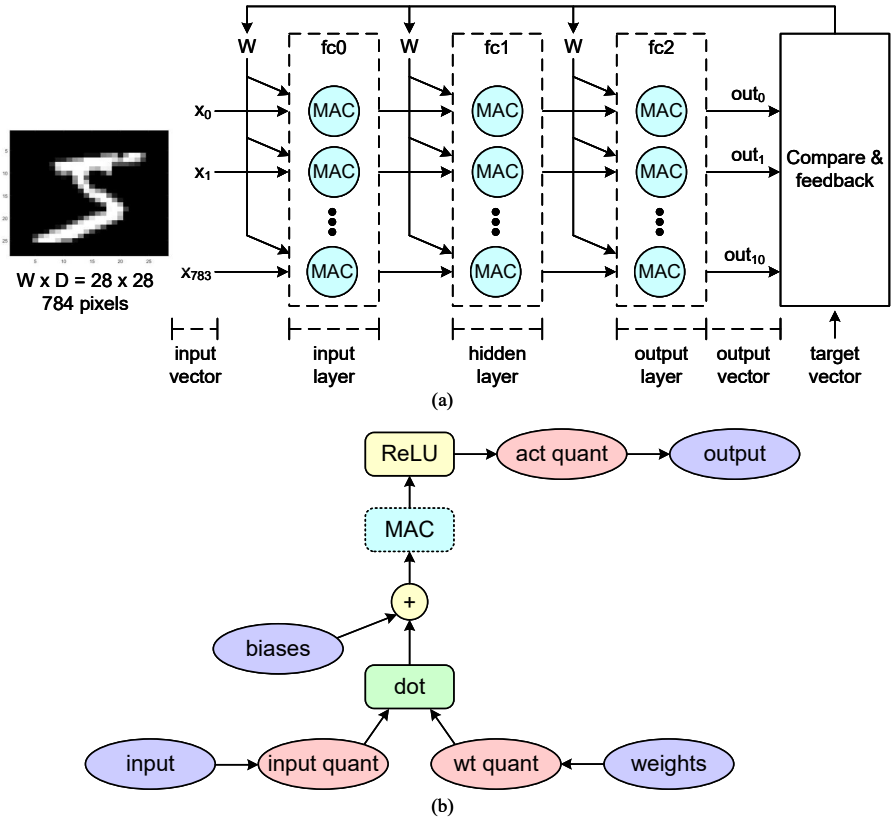


Figure 5.9: The NN structure and training graph. (a) presents NN structure to demonstrate MNIST classification using the proposed MAC unit. It consists of three fully-connected layers, each of which (input/hidden/output) contains 800/500/10 neurons. The traditional MAC unit will be replaced by the proposed one. (b) presents the training graph of the NN. We added the MAC block (highlighted in blue) where the output of the dot-product will be subtracted by the non-ideal effect of our MAC unit following Eq. (5.23) and the multiplication errors in Table 5.2. This allows the NN to learn the loss regarding the proposed MAC unit.

5.4.5 Effects of Technology Parametric Variations

However, device parametric variation in multiplication cell may lead to additional and substantial analogue output error. Devices may have different properties or technology parametric variations. For our MAC, we consider faster/slower operating speeds of transistors and higher/lower R_{MH} and R_{ML} values of memristors. Therefore,

the multiple-component cell design in this work risks large accuracy drops resulting from such variations. Both the transistor variation and memristor variation have been investigated to show the relation between variation and NN accuracy of MNIST classification.

The variability transistor models are investigated first. The fabricated transistor's performance are studied for the Fast-Fast (FF), Typical-Typical (TT), and Slow-Slow (SS) corners. analogue simulations of the MAC corresponding with these corners are used to generate modified MAC input to output error maps in the same style as Table 5.2. Then respective NN simulation using the method given in Section 5.4.4 generates the accuracy results reported in Table 5.5.

In this study, we investigate the impact of memristor resistance variability. According to the findings presented in reference [50], our selected technology (Cu:ZnO) demonstrates a device-to-device (DD) variability of 59% for the high resistance state (HRS) and 36% for the low resistance state (LRS). Furthermore, the cycle-to-cycle (CC) variability is particularly significant, with the HRS exhibiting 89% variability and the LRS exhibiting 51% variability. It is worth noting that, despite the considerable CC variability, the resistance of the low resistance state (LRS) cannot exceed that of the high resistance state (HRS), as the baseline ratio between these two parameters is fixed at 1000 for the Cu:ZnO technology. Even in the worst-case scenario, the Cu:ZnO technology's OFF state resistance is only 640 times greater than the ON state resistance, which is considerably better than technologies with smaller ON/OFF ratios. In fact, under CC variation, the ON/OFF ratio remains at 227, which is adequate to fulfil the precision requirements of a 4-bit multiplier.

Similar to the case of transistor variation investigations, our simulation investigations include analogue simulations of one MAC unit with all possible corner cases of expected variability in the memristors. The result of these simulations is put into digital models in the form of input value to output value correspondence error maps in the form of Table 5.2. These corner case models are then used in NN training exercises on the MNIST dataset, using exactly the same method described in Section 5.4.4. The accuracy results are reported in Table 5.5. In addition, distribution of the sum current of each column has also been put in Table 5.5, since the multiplication cell current includes transistor's

Table 5.5: QAT NN with MAC Component Variation Training.

Transistor		**Training acc. (%)**		**Testing acc. (%)**	
Slow-Slow		96		96	
Typical-Typical		96		96	
Fast-Fast		90		**85**	
Memristor		**Training acc. (%)**		**Testing acc. (%)**	
		Average	Worst	Average	Worst
DD		95	**86**	94	**79**
CC		95	95	95	94

Distribution of Column Sum Current								
Slow-Slow		CL6(MSB)	CL5	CL4	CL3	CL2	CL1	CL0(LSB)
Relative Standard	i_{off}	13.3	13.3	13.3	13.3	13.3	13.3	13.3
Deviation (%)	i_{on}	7.65	7.65	7.65	7.65	7.65	7.65	7.65
Typical-Typical		CL6(MSB)	CL5	CL4	CL3	CL2	CL1	CL0(LSB)
Relative Standard	i_{off}	12	12	12	12	12	12	12
Deviation (%)	i_{on}	7.53	7.53	7.53	7.53	7.53	7.53	7.53
Fast-Fast		CL6(MSB)	CL5	CL4	CL3	CL2	CL1	CL0(LSB)
Relative Standard	i_{off}	0.667	0.667	0.667	0.667	0.667	0.667	0.660
Deviation (%)	i_{on}	5.88	5.88	5.88	5.87	5.87	5.87	5.46

contribution, different transistor models are listed as name for the relative standard deviation data group. This relation proves the fixed proportional I-V relation derived from Eq. (5.16)

In presenting these results we focus on investigating how the worst-case scenarios of memristor variability may affect the NN application and compare with the average case. The worst case happens when R_{MH} takes the lowest possible value coinciding with R_{ML} taking the highest possible value. This maximally reduces the margin between these two values and hence reduce the precision of the multiplier part of the MAC, as discussed in

Section 2.3.1.

The reported average case results are the average values obtained from all different corner cases and do not correspond with any one particular set of parameter value. It is noteworthy that some of the accuracy numbers reported in Table 5.5 are actually better than those reported in the last row of Table 5.4. This is because in many cases, the technology parametric variation corner cases have smaller errors in their input-output relation error maps than the non-variation case of Table 5.5. This is a result of effective cancellations between the two kinds of errors. The true global worst case results, however, do happen with worst-case memristor parametric variation combinations.

As can be seen from the results, in all experiments both training and testing always successfully complete, but in the highlighted cases the accuracy does not achieve better than 90%. Even the global worst case of 79% accuracy should be tolerable for low-power edge AI applications. It is also noteworthy that NN operations seem to be especially resistant to the CC type of parametric variability. This is likely because NN operations usually include a substantial number of cycles during which CC variability in the MACs is moderated by a kind of low-pass filtering process.

5.5 Summary

This chapter presents a MAC unit based on the crossbar multiplier. Using memristor-transistor SBMCs with mixed-signal design, this crossbar multiplier saves the time required for carry propagation, and reduces the circuit complexity by avoiding long logic chains. Multiplying by passive current generation across resistive elements only, the multiplication step itself can be regarded as instantaneous according to Ohm's law and KCL. Using a mixed-mode, flash A2D conversion step, latency is kept under control for the ultimate DI/DO unit by employing single-action thermometer code generation. This means that the worst-case delay depends only on writing memristor values and converting thermometer code to binary code. This latency management means that the MAC unit has a relatively low working latency of 5.36 ns, the worst latency scenario includes reset (4 row write 0 operations), fully write (4 row write 1 operations), and read (1 read operation).

At the same time, the energy efficiency is also improved over conventional digital multipliers using memristors by eliminating the need for costly carry-to-the-left operations.

The proposed MAC unit also has the same precision for both input and output, which means that it can be used to compose multi-MAC structures such as NNs without worrying about bit-conversion when fitting the outputs of one layer to the inputs of another layer. The approximation happens in the thermometer code generation step where it leads to reductions in circuit size and complexity in subsequent circuitry without sacrificing precision unnecessarily.

To validate this MAC unit, it is used as the basic perceptron in the creation of an NN of multiple neurons and layers, and the resulting NN is used to classify the MNIST dataset. The low precision and multiplication errors attributed to the analogue product from the crossbar multiplier are shown to be compensatable through an extended use of QAT. With such compensation techniques, the proposed case study NN achieves comparable learning accuracy to the same NN based on fully-digital QAT MAC units of the same bit width. In doing this, this chapter additionally demonstrates the potential for extending QAT to compensate for any characterisable imprecision beyond quantization effects in the perceptron unit. The effects of parametric variability for both transistors and memristors are also investigated demonstrating the usability of this type of MAC units. These have shown promising results and further development of this demonstrates that this MAC design approach opens up future research opportunities in low-energy, low-latency, edge AI applications.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

6.1.1 Contributions

In this research, memristor-transistor cell-based design solutions have been proposed to improve energy efficiency in IoT devices for higher-level standard requirements.

For the algorithm circuit design, we present novel multiplier designs that use transistor-memristor cells for bit-wise multiplication. By working in a mixed-signal mode, these designs remove the need for carry-to-the-left operations in conventional digital multipliers and provide an analogue output. It is important to eliminate carry propagation and DAC circuits while maintaining edge computing digital input interfaces. Because this allows the majority of the computation to remain digital, with its associated advantages, but produces the required analogue output directly. The substantial margin of memristance differences between the ON and OFF states of a memristor provides this design several advantages. The major advantage is sufficient accuracy for analogue-out multipliers, also the ability to represent logic ”1” and ”0” with large separation between high and low analogue current values.

The multiplication is performed by mapping one of the operands to memductance values. The multipliers benefit from intrinsic data retention in several scenarios with

non-volatile memristors as the core, in-memory compute units. These scenarios include when an input variable is multiplied by a constant coefficient, a variable number multiplied by a relatively constant reference, or a fixed number multiplied by a variable reference. These use cases are frequently seen in control, signal processing, AI, and MDAC applications.

Using multiple memristors in parallel in each cell, we relocate the bit significance weighting function from current mirrors to the number of memristors in a cell. This allows the proposed multiplier, which is based on a single transistor multiple Cu:ZnO memristor (1TxM), to outperform recently reported designs in hardware complexity, performance, and energy while staying competitive on peak power. However, these advantages come at the cost of limits in the memductance adjustment range, which affect the large-scale implementations beyond a 4-bit multiplier, which is nonetheless sufficient for many micro-edge applications [91].

All our multiplication circuit implementation are based on 4-bit cases. The upscale of the circuit could be realised through algorithm adjustment with the same level performance [97, 98].

Also, a MAC unit based on a crossbar multiplier is presented. Using memristor-transistor SBMCs with a mixed-signal design, this crossbar multiplier removes the need for carry propagation. It also reduces circuit complexity by avoiding long logic chains. Multiplying by passive current generation across resistive elements only, the multiplication step can be regarded as instantaneous according to Ohm's law and KCL. By using a mixed-mode, flash ADC conversion step, latency is kept under control for the ultimate DI/DO unit through single-action thermometer code generation. The worst-case delay depends only on writing memristor values and converting thermometer code to binary code. This latency management means that the MAC unit has a relatively high working frequency of (20.7 MHz).

The proposed MAC unit also has the same precision for input and output. It can be used to compose multi-MAC structures such as NNs without worrying about bit-conversion when fitting outputs of one layer to the inputs of another layer. The approximation happens in the thermometer code generation step, leading to circuit size and complexity reductions in subsequent circuitry without sacrificing precision

unnecessarily.

To validate this MAC unit, a basic perceptron based on it is used in the creation of an NN of substantial size, and the resulting NN is used to classify the MNIST dataset.

6.1.2 Limitations of the Research

The numbers of memristors and transistors in a single cell are in reality limited by such problems as leakage. At some point this would overwhelm any additional bit resolution increase predicted by theory. The number of memristors per cell increases exponentially with the number of multiplier bits, leading to practical difficulties if the precision needs to be scaled up.

Memristors have better characteristics in some ways and worse characteristics in other ways compared with other forms of RRAM technologies. This work does not demonstrate whether memristors are the best RRAM technology of choice for these types of multiplier designs. It only shows that it is possible to realise working designs using Cu:ZnO memristors. The design approach and crossbar structure should be applicable to cells based on any type of RRAM - this remains unexplored.

The design's low multiplication energy consumption claim is based on the energy consumed between points in time regarded as the start and end of multiplication. The end of multiplication is defined as when a usable product first appears. In reality, this output needs to be maintained for some time for the user to make effective use of it and this further holding time is not included in the energy estimate. This is because the user of the output of the multiplier is outside the scope of the thesis and the required holding time is therefore unknown.

One of the real-world problems encountered during this work is that memristor and transistor widths need to be carefully selected because the unintended tuning (UT) caused by over threshold voltage, when the memristance is too high, or the transistors are too wide (with low channel resistances).

6.2 Future Work

This thesis opens up possibilities for future research in energy-efficient high-performance arithmetic circuit design. Below we discuss several possible work for next step:

1. **Design Up-scaling –** Although one crucial problem with these multiplier solutions is that the multiplier bit-width is limited by the ratio between the ON and OFF currents in the RRAM technology, it is possible to scale up the number of bits of a multiplication by using multiple copies of low-bit multipliers. Whether this is a realistic proposition for the multipliers presented in this thesis remains unexplored. The principles of the design approach should be applicable for any RRAM technology which is an opportunity for future work, as and when good SPICE-level models of current and future proposed RRAM technologies appear.

2. **Practical Implementation –** Whether the DI/AO multipliers can be used effectively in real-world edge computing applications remains unexplored. Opportunities exist in exploring the use of such multipliers in a wide range of potential application systems, especially for computing at the edge.

3. **Robust Test –** More sophisticated NNs and learning automata, and larger and more complex data sets have not been explored with the MAC unit presented in this thesis. Given the promising results achieved so far, further explorations in such uses of multipliers and MACs designed using the methods presented in this thesis have good potential of yielding good results.

Bibliography

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[2] N. Zheng and P. Mazumder, *Learning in Energy-Efficient Neuromorphic Computing: Algorithm and Architecture Co-Design*. John Wiley & Sons, 2019.

[3] S. Choi, J. Yang, and G. Wang, "Emerging memristive artificial synapses and neurons for energy-efficient neuromorphic computing," *Advanced Materials*, vol. 32, no. 51, p. 2004659, 2020.

[4] D. Fujiki, X. Wang, A. Subramaniyan, and R. Das, "In-/near-memory computing," *Synthesis Lectures on Computer Architecture*, vol. 16, no. 2, pp. 1–140, 2021.

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

[6] P. Kumar, K. Zhu, X. Gao, S.-D. Wang, M. Lanza, and C. S. Thakur, "Hybrid architecture based on two-dimensional memristor crossbar array and cmos integrated circuit for edge computing," *npj 2D Materials and Applications*, vol. 6, no. 1, pp. 1–10, 2022.

[7] J.-M. Hung, C.-J. Jhang, P.-C. Wu, Y.-C. Chiu, and M.-F. Chang, "Challenges and trends of nonvolatile in-memory-computation circuits for ai edge devices," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 1, pp. 171–183, 2021.

[8] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: A review," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 1, pp. 4–23, 2019.

[9] S. Zhang, K. Huang, and H. Shen, "A robust 8-bit non-volatile computing-in-memory core for low-power parallel mac operations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 6, pp. 1867–1880, 2020.

[10] I. Surekcigil Pesch, E. Bestelink, O. de Sagazan, A. Mehonic, and R. A. Sporea, "Multimodal transistors as relu activation functions in physical neural network classifiers," *Scientific Reports*, vol. 12, no. 1, pp. 1–7, 2022.

[11] S. Madakam, V. Lake, V. Lake, V. Lake *et al.*, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[12] R. Shafik, A. Yakovlev, and S. Das, "Real-power computing," *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1445–1461, 2018.

[13] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 688–698.

[14] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, S. Das, and A. Yakovlev, "Significance-driven logic compression for energy-efficient multiplier design," *IEEE J. Emerging and Selected Topics in Circuits and Systems*, 2018.

[15] A. Cilardo, D. De Caro, N. Petra, F. Caserta, N. Mazzocca, E. Napoli, and A. G. M. Strollo, "High speed speculative multipliers based on speculative carry-save tree," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 61, no. 12, pp. 3426–3435, 2014.

[16] R. A. Shafik, S. Yang, A. Das, L. A. Maeda-Nunez, G. V. Merrett, and B. M. Al-Hashimi, "Learning transfer-based adaptive energy minimization in embedded systems," *IEEE Trans. on Comp.-Aided Des. of Integ. Circuits and Systems (TCAD)*, vol. 35, no. 6, pp. 877–890, 2016.

[17] A. Yakovlev, "Enabling Survival Instincts in Electronic Systems: An Energy Perspective," in *Transforming Reconfigurable Systems*. Imperial College Press, Apr 2015, pp. 237–263.

[18] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature electronics*, vol. 1, no. 1, pp. 52–59, 2018.

[19] C.-X. Xue, Y.-C. Chiu, T.-W. Liu, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, S.-Y. Wei, C.-Y. Lee *et al.*, "A cmos-integrated compute-in-memory macro based on resistive random-access memory for ai edge devices," *Nature Electronics*, vol. 4, no. 1, pp. 81–90, 2021.

[20] B. Razavi, *Basic Principles of Digital to Analog Conversion*. Wiley-IEEE Press, 1995, pp. 45–78.

[21] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[22] A. Raj, D. R. Bhaskar, and P. Kumar, "Novel architecture of four quadrant analog multiplier/divider circuit employing single CFOA," *Analog Integrated Circuits and Signal Processing*, vol. 108, no. 3, pp. 689–701, Sep 2021.

[23] Z. Gafsi, N. Hassen, M. Mhiri, and K. Besbes, "A new efficient-silicon area mdac synapse," *American Journal of Applied Sciences*, vol. 4, no. 6, pp. 378–385, 2007.

[24] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Advanced Materials*, vol. 30, no. 9, p. 1705914, 2018.

[25] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms," in *2015 Int. Joint Conf. Neural Networks (IJCNN)*. IEEE, Jul 2015, pp. 1–7.

[26] S. Mileiko, T. Bunnam, F. Xia, R. Shafik, A. Yakovlev, and S. Das, "Neural network design for energy-autonomous artificial intelligence applications using temporal encoding," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, no. 2164, p. 20190166, Feb 2020.

[27] D. R. Gandhi and N. N. Shah, "Comparative analysis for hardware circuit architecture of wallace tree multiplier," in *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*. IEEE, 2013, pp. 1–6.

[28] W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," *IEEE transactions on computers*, vol. 49, no. 7, pp. 692–701, 2000.

[29] D. E. Knuth, "The average time for carry propagation," in *Indagationes Mathematicae (Proceedings)*, vol. 81, no. 1. North-Holland, 1978, pp. 238–242.

[30] O. J. Bedrij, "Carry-select adder," *IRE Transactions on Electronic Computers*, no. 3, pp. 340–346, 1962.

[31] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on electronic Computers*, no. 1, pp. 14–17, 1964.

[32] M. V. Wilkes, D. J. Wheeler, and S. Gill, *The Preparation of Programs for an Electronic Digital Computer: With special reference to the EDSAC and the Use of a Library of Subroutines*. Addison-Wesley Press, 1951.

[33] K. Uyttenhove and M. S. Steyaert, "Speed-power-accuracy tradeoff in high-speed cmos adcs," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 4, pp. 280–287, 2002.

[34] P. E. Allen and D. R. Holberg, *CMOS analog circuit design*. Elsevier, 2011.

[35] V. Vinayaka, S. P. Namboodiri, S. Abdalla, B. Kerstetter, F. Mata-carlos, D. Senda, J. Skelly, A. Roy, and R. J. Baker, "Monolithic 8x8 sipm with 4-bit current-mode flash adc with tunable dynamic range," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019, pp. 57–62.

[36] D. Freitas and K. Current, "Cmos current comparator circuit," *Electronics letters*, vol. 19, no. 17, pp. 695–697, 1983.

[37] V. H. Bui, S. Beak, S. Choi, J. Seon, and T. T. Jeong, "Thermometer-to-binary encoder with bubble error correction (bec) circuit for flash analog-to-digital converter (fadc)," in *International Conference on Communications and Electronics 2010*. IEEE, 2010, pp. 102–106.

[38] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

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

[40] A. G. Radwan, M. A. Zidan, and K. N. Salama, "HP Memristor mathematical model for periodic signals and DC," in *IEEE Int. Midwest Sym. on Circuits & Systems*, 2012.

[41] W. S. Lew, G. J. Lim, and P. A. Dananjaya, *Emerging Non-volatile Memory Technologies: Physics, Engineering, and Applications*. Springer, 2021.

[42] S. Hong, O. Auciello, and D. Wouters, *Emerging non-volatile memories*. Springer, 2014.

[43] K. Roy, I. Chakraborty, M. Ali, A. Ankit, and A. Agrawal, "In-memory computing in emerging memory technologies for machine learning: an overview," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[44] Y. Ho, G. M. Huang, and P. Li, "Nonvolatile memristor memory: Device characteristics and design implications," in *2009 Int. Conf. Computer-Aided Design (ICCAD'09), November 2-5, 2009, San Jose, CA, USA*, 2009.

[45] T. Reid, "Memristor multiplication," *Nature Nanotechnology*, pp. 1–1, 2009.

[46] P. K. R. Boppidi, P. M. P. Raj, S. Challagulla, S. R. Gollu, S. Roy, S. Banerjee, and S. Kundu, "Unveiling the dual role of chemically synthesized copper doped zinc oxide for resistive switching applications," *Journal of Applied Physics*, vol. 124, no. 21, p. 214901, 2018.

[47] H. Lee, P. Chen, T. Wu, Y. Chen, C. Wang, P. Tzeng, C. Lin, F. Chen, C. Lien, and M.-J. Tsai, "Low power and high speed bipolar switching with a thin reactive ti buffer

layer in robust hfo2 based rram," in *2008 IEEE International Electron Devices Meeting*. IEEE, 2008, pp. 1–4.

[48] B. Suresh, P. K. R. Boppidi, B. P. Rao, S. Banerjee, and S. Kundu, "Realizing spike-timing dependent plasticity learning rule in pt/cu: Zno/nb: Sto memristors for implementing single spike based denoising autoencoder," *J. Micromechanics and Microengineering*, vol. 29, no. 8, p. 085006, 2019.

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

[50] A. Siddik, P. K. Haldar, P. Garu, S. Bhattacharjee, U. Das, A. Barman, A. Roy, and P. K. Sarkar, "Enhancement of data storage capability in a bilayer oxide-based memristor for wearable electronic applications," *Journal of Physics D: Applied Physics*, vol. 53, no. 29, p. 295103, 2020.

[51] E. Lehtonen and M. Laiho, "CNN using memristors for neighborhood connections," in *2010 12th Int. Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, 2010, pp. 1–4.

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

[53] R. Berdan, A. Khiat, C. Papavassiliou, and T. Prodromakis, "Qualitative SPICE modeling accounting for volatile dynamics of TiO_2 memristors," in *2014 IEEE Int. Sym. Circuits & Systems (ISCAS)*, 2014, pp. 2033–2036.

[54] X. Fang, S. Duan, and L. Wang, "Memristive hodgkin-huxley spiking neuron model for reproducing neuron behaviors," *Frontiers in Neuroscience*, vol. 15, 2021.

[55] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: ThrEshold Adaptive Memristor Model," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211–221, 2013.

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

[57] C. Lammie, W. Xiang, B. Linares-Barranco, and M. R. Azghadi, "Memtorch: An open-source simulation framework for memristive deep learning systems," *Neurocomputing*, 2022.

[58] S. Singh, P. W. C. Prasad, A. Alsadoon, A. Beg, L. Pham, and A. Elchouemi, "Survey on memrister models," in *2016 Int. Conf. Electronics, Information, and Communications (ICEIC)*, 2016, pp. 1–7.

[59] S. Pi, C. Li, H. Jiang, W. Xia, H. Xin, J. J. Yang, and Q. Xia, "Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension," *Nature Nanotechnology*, vol. 14, no. 1, pp. 35–39, 2019.

[60] W.-H. Chen, C. Dou, K.-X. Li, W.-Y. Lin, P.-Y. Li, J.-H. Huang, J.-H. Wang, W.-C. Wei, C.-X. Xue, Y.-C. Chiu *et al.*, "Cmos-integrated memristive non-volatile computing-in-memory for ai edge processors," *Nature Electronics*, vol. 2, no. 9, pp. 420–428, 2019.

[61] W. C. Shen, Y. H. Tseng, Y. . Chih, and C. J. Lin, "Memristor Logic Operation Gate With Share Contact RRAM Cell," *IEEE Electron Device Letters*, vol. 32, no. 12, pp. 1650–1652, 2011.

[62] S. Pal, S. Bose, W. Ki, and A. Islam, "Design of Power- and Variability-Aware Nonvolatile RRAM Cell Using Memristor as a Memory Element," *IEEE J. the Electron Devices Society*, vol. 7, pp. 701–709, 2019.

[63] P. Chiu, M. Chang, C. Wu, C. Chuang, S. Sheu, Y. Chen, and M. Tsai, "Low Store Energy, Low VDDmin, 8T2R Nonvolatile Latch and SRAM With Vertical-Stacked Resistive Memory (Memristor) Devices for Low Power Mobile Applications," *IEEE J. Solid-State Circuits*, vol. 47, no. 6, pp. 1483–1496, 2012.

[64] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, 2010.

[65] D. Radakovits, N. TaheriNejad, M. Cai, T. Delaroche, and S. Mirabbasi, "A Memristive Multiplier Using Semi-Serial IMPLY-Based Adder," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1495–1506, 2020.

[66] Y. Wang, Y. Li, H. Shen, D. Fan, W. Wang, L. Li, Q. Liu, F. Zhang, X. Wang, M.-F. Chang *et al.*, "A Few-Step and Low-Cost Memristor Logic Based on MIG Logic for Frequent-Off Instant-On Circuits in IoT Applications," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 66, no. 4, pp. 662–666, 2018.

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

[68] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[69] H. Choi, H. Jung, J. Lee, J. Yoon, J. Park, D.-j. Seong, W. Lee, M. Hasan, G.-Y. Jung, and H. Hwang, "An electrically modifiable synapse array of resistive switching memory," *Nanotechnology*, vol. 20, no. 34, p. 345201, 2009.

[70] L. O. Chua, R. Tetzlaff, and A. Slavova, "Memristor computing systems," 2022.

[71] V. Constantoudis, G. Papavieros, P. Karakolis, A. Khiat, T. Prodromakis, and P. Dimitrakis, "Impact of Line Edge Roughness on ReRAM Uniformity and Scaling," *Materials*, vol. 12, no. 23, p. 3972, Nov 2019.

[72] M. A. Eldeeb, Y. H. Ghallab, Y. Ismail, and H. El-Ghitani, "A 0.4-v miniature cmos current mode instrumentation amplifier," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 3, pp. 261–265, 2017.

[73] F. Yuan, "Low-voltage cmos current-mode circuits: topology and characteristics," *IEE Proceedings-Circuits, Devices and Systems*, vol. 153, no. 3, pp. 219–230, 2006.

[74] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[75] A. Wheeldon, R. Shafik, T. Rahman, J. Lei, A. Yakovlev, and O.-C. Granmo, "Learning automata based energy-efficient ai hardware design for iot applications," *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2182, p. 20190593, 2020.

[76] S. Draghici, "Neural networks in analog hardware—design and implementation issues," *International journal of neural systems*, vol. 10, no. 01, pp. 19–42, 2000.

[77] G. Tagliavini, A. Marongiu, D. Rossi, and L. Benini, "Always-on motion detection with application-level error control on a near-threshold approximate computing platform," in *ICECS*. IEEE, 2016, pp. 552–555.

[78] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "A carry-free 54b/spl times/54b multiplier using equivalent bit conversion algorithm," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1538–1545, 2001.

[79] S. Getzlaff and R. Schüffny, "A mixed signal multiplier principle for massively parallel analog VLSI systems," in *Recent Advances in Signal Processing and Communication; Proceedings of the 3rd IMACS Int. Multiconference on Circuits, Systems, Communications and Computers CSCC'99*. Athens, Greecs, 1999.

[80] T.-B. Juang and S.-F. Hsiao, "Low-error carry-free fixed-width multipliers with low-cost compensation circuits," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 52, no. 6, pp. 299–303, 2005.

[81] H. Ling, "High-speed binary adder," *IBM Journal of Research and Development*, vol. 25, no. 3, pp. 156–166, 1981.

[82] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Rap-cla: A reconfigurable approximate carry look-ahead adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 8, pp. 1089–1093, 2018.

[83] B. Govoreanu, G. S. Kar, Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I. Radu, L. Goux, S. Clima, R. Degraeve *et al.*, "10× 10nm 2 hf/hfo x crossbar resistive ram with excellent performance, reliability and low-energy operation," in *2011 International Electron Devices Meeting*. IEEE, 2011, pp. 31–6.

[84] G. G. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825–1854, 2000.

[85] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSI Design (VLSI Design), 2011 24th Int. Conf.* IEEE, 2011, pp. 346–351.

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

[87] Z. Li, D. Yu, Z. Ye, H. H. Iu, and T. Fernando, "Memristor-based logic gate and its application in pulse train controlled buck converter," *International Journal of Circuit Theory and Applications*, 2022.

[88] L. Guckert and E. Swartzlander, "Mad gates—memristor logic design using driver circuitry," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 64, no. 2, pp. 171–175, 2016.

[89] L. Guckert and E. Swartzlander, "Optimized memristor-based ripple carry adders," in *2016 50th Asilomar Conf. Signals, Systems and Computers*. IEEE, 2016, pp. 1575–1579.

[90] W.-T. Lin, H.-Y. Huang, and T.-H. Kuo, "A 12-bit 40 nm dac achieving sfdr¿ 70 db at 1.6 gs/s and imd¡–61db at 2.8 gs/s with demdrz technique," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 3, pp. 708–717, 2014.

[91] L. K. Muller and G. Indiveri, "Rounding Methods for Neural Networks with Low Resolution Synaptic Weights," *arXiv e-prints*, p. arXiv:1504.05767, Apr 2015.

[92] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization," *arXiv e-prints*, jun 2021. [Online]. Available: http://arxiv.org/abs/2106.08295

[93] K. Chahal, "Aggressive Quantization: How to run MNIST on a 4 bit Neural Net using Pytorch," Oct 2019, avaiable: https://karanbirchahal.medium.com/aggressive-quantization-how-to-run-mnist-on-a-4-bit-neural-net-using-pytorch-5703f3faa599.

[94] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[95] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 2, pp. 1–23, 2018.

[96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[97] N. Nedjah and L. de Macedo Mourelle, "A review of modular multiplication methods ands respective hardware implementation," *Informatica*, vol. 30, no. 1, 2006.

[98] M. Ibraimov, S. Tynymbayev, A. Skabylov, Y. Kozhagulov, and D. Zhexebay, "Development and design of an fpga-based encoder for npn," *Cogent Engineering*, vol. 9, no. 1, p. 2008847, 2022.