Design and Analysis of Memristor Based Reliable Crossbar Architectures

---

by

Adedotun Adeyemo

Thesis submitted in partial fulfilment of the requirements of the award of

Doctor of Philosophy

in

School of Engineering, Computing and Mathematics
Faculty of Technology, Design and Environment
Oxford Brookes University, Oxford, UK.

June 29, 2018

Abstract

The conventional transistor-based computing landscape is already undergoing dramatic changes. While transistor-based devices' scaling is approaching its physical limits in nanometer technologies, memristive technologies hold the potential to scale to much smaller geometries.

Memristive devices are used majorly in memory design but they also have unignorable applications in logic design, neuromorphic computing, sensors among many others. The most critical research and development problems that must be resolved before memristive architectures become mainstream are related to their reliability. One of such reliability issue is the sneak-paths current which limits the maximum crossbar array size. This thesis presents various designs of the memristor based crossbar architecture and corresponding experimental analysis towards addressing its reliability issues.

Novel contribution of this thesis starts with the formulation of robust analytic models for read and write schemes used in memristive crossbar arrays. These novel models are less restrictive and are suitable for accurate mathematical analysis of any $m \times n$ crossbar array and the evaluation of their performance during these critical operations. In order to minimise the sneak-paths problem, we propose techniques and conditions for reliable read operations using simultaneous access of multiple bits in the crossbar array. Two new write techniques are also presented, one to minimise failure during single cell write and the other designed for multiple cells write operation. Experimental results prove that the single write technique minimises write voltage drop degradation compared to existing techniques. Test results from the multiple cells write technique show it consumes less power than other techniques depending on the chosen configuration.

Lastly, a novel Verilog-A memristor model for simulation and analysis of memristor's application in gas sensing is presented. This proposed model captures the gas sensing properties of titanium-dioxide using gas concentration to control the overall memristance of the device. This model is used to design and simulate a first-of-its-kind sneak-paths free memristor-based gas detection arrays. Experimental results from a 8×8 memristor sensor array show that there is a ten fold improvement in the accuracy of the sensor's response when compared with a single memristor sensor.

Publications

Research Papers from this Thesis

Adeyemo, Adedotun, A. Jabir, J. Mathew, E. Martinelli, C. Di Natale, and M. Ottavi, "Efficient sensing approaches for high-density memristor sensor array," *Journal of Computational Electronics*, 2017, in-review

Adeyemo, Adedotun, A. Jabir, and J. Mathew, "Minimizing impact of wire resistance in low-power crossbar array write scheme," *Journal of Low Power Electronics*, vol. 13, no. 4, pp. 649–660, 2017 **(Figure 5 selected as journal's cover page)**

Adeyemo, Adedotun, A. Jabir, J. Mathew, E. Martinelli, C. Di Natale, and M. Ottavi, "Reliable gas sensing with memristive array," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017, pp. 244–246

Adeyemo, Adedotun, X. Yang, A. Bala, and A. Jabir, "Analytic models for crossbar write operation," in *Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on*, 2016, pp. 313–317

Adeyemo, Adedotun, X. Yang, A. Bala, J. Mathew, and A. Jabir, "Analytic models for crossbar read operation," in *IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2016, pp. 3–4

Adeyemo, Adedotun, J. Mathew, A. M. Jabir, and D. Pradhan, "Exploring error-tolerant low-power multiple-output read scheme for memristor-based memory arrays," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2015, pp. 17–20

Adeyemo, Adedotun, J. Mathew, A. Jabir, and D. Pradhan, "Write scheme for multiple complementary resistive switch (crs) cells," in *IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2014, pp. 1–5

Research Papers in Allied Areas

X. Yang, **Adeyemo, Adedotun**, A. Bala, and A. Jabir, "Parasitic effects on memristive logic architecture," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2017, pp. 1–5

X. Yang, **Adeyemo, Adedotun**, A. Bala, and A. Jabir, "Novel techniques for memristive multifunction logic design," *The VLSI Journal on Integration*, 2017

X. Yang, **Adeyemo, Adedotun**, A. Jabir, and J. Mathew, "High-performance single-cycle memristive multifunction logic architecture," *Electronics Letters*, vol. 52, no. 11, pp. 906–907, 2016

A. Bala, **Adeyemo, Adedotun**, X. Yang, and A. Jabir, "High level abstraction of memristor model for neural network simulation," in *Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on*, 2016, pp. 318–322

X. Yang, **Adeyemo, Adedotun**, A. Bala, and A. Jabir, "Novel memristive logic architectures," in *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016 26th International Workshop on*, 2016, pp. 196–199 **(paper won best poster award)**

Patents

Adeyemo, Adedotun, A. Jabir, J. Mathew, E. Martinelli, C. Di Natale, and M. Ottavi, *Memristive sensor array*, UK Patent App. 1616837.9, Oct. 2016

A. Jabir, **Adeyemo, Adedotun**, and X. Yang, *Memristive multifunction logic architecture*, UK Patent App. 1603089.2, Feb. 2016

Declaration

This thesis is submitted to the Oxford Brookes University in accordance with the requirements of the award of Doctor of Philosophy in the Faculty of Technology, Design and Environment. It has not been submitted for any other degree or diploma of any examining body. Some parts of the work presented in this thesis have previously appeared in the published papers listed in the publications section. Except where specifically acknowledged, it is all the work of the Author.

Adedotun Adeyemo, February, 2018.

Acknowledgement

My utmost gratitude to God Almighty for his unfailingly love all through the years.
This project is a result of weeks of hard work, perseverance and dedication alongside support from good people around me.

My sincere appreciation to my supervisors and research team members, Dr. Abusaleh Jabir, Dr. Jimson Mathew, Prof. Dhiraj Pradhan, Xiaohan Yang and Anu Bala who are always willing and able to give me the required support for the duration of this work.

I am thankful to my wife - Isabel Adeyemo, my friends - Dr. Mayokun Adetoro, Abraham Fakolade and all GHIC members. They were always there for me even during the course of writing this dissertation.

I sincerely offer my heartfelt gratitude to my wonderful families (Adeyemo, Omotosho, Animashaun, Akintayo e.t.c) and my sweet siblings (Niran, Tunde, Sekinat, Busola, Yetunde, Taiwo, Kenny.); they all believed in God for me. Thank you all for the encouragement and love towards me.

Table of Contents

List of Tables

List of Figures

Listings

List of Acronyms

1D1M	One Diode and One Memristor
1M1M	One Memistor and One Memristor
1T1M	One Transistor and One Memristor
Al_2O_3	Aluminium oxide
CMOS	Complementary Metal-Oxide-Semiconductor
CNT	Carbon Nanotubes
CTO	Chromium titanate
Cu_2O	Copper oxide
DSGB	Double-Sided Ground Biasing
E-b-R	ERASE-before-RESET
E-b-S	ERASE-before-SET
EDA	Electronic Design Automation
FRAM	Ferroelectric Random Access Memory
FWFB	Floating Wordlines and Floating Bitlines
FWGB	Floating Wordlines and Grounded Bitlines
GWFB	Grounded Wordlines and Floating Bitlines
GWGB	Grounded Wordlines and Grounded Bitlines
I-H	Inverted-Hamming
I-V	Current-Voltage
ITRS	International Roadmap for Semiconductor
MLMM	Multi-Level Memristor Memory
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MRAM	Magnetoresistive Random Access Memory
NEMS	NanoElectroMechanical Switches

PCRAM	Phase Change Random Access Memory
ppm	Parts per million
ReRAM	Resistive Random Access Memory
S-b-R	SET-before-RESET
SiO	Silicon oxide
SnO_2	Tin dioxide
STTRAM	Spin-Transfer-Torque RAM
TiO_2	Titanium dioxide
VPWL	Piece-wise Linear Voltage source
ZnO	Zinc oxide

Chapter 1

Introduction

1.1 Problem Background

Over the years, area reduction, power conservation, fast operation, increased reliability and eventually cost reduction has been the main focus of the semiconductor industry when designing any integrated circuit. The industry has been able to achieve all these by striving to follow the trend put forward by Gordon Moore in 1956 in which he predicted that the number of transistors on a chip will double every eighteen months [15]. Information processing and storage technologies have benefited immensely from the continuous scaling of Complementary Metal Oxide Semiconductor (CMOS) devices over the years. However, scaling rate have since slowed down as transistor scaling becomes increasingly difficult according to the International Roadmap for Semiconductor (ITRS) based on data gathered from leading semiconductor manufacturers (Fig. 1.1). The scaling problem is caused by various factors, such as the leakage currents that arise from the device getting smaller, insulation and conduction challenges with dielectric and wiring materials among others. CMOS devices are also synonymous with parasitic capacitances that limit their performance. The effect of these parasitic capacitances will also increase as the device features are brought closer to each other [16]. Economic challenges is another hinderance to CMOS scaling due to the rising cost of fabs and testing of new techniques.

As it gradually becomes inevitable to prevent CMOS from reaching its fundamental dimensional and functional limits, the industry has been tasked with the huge responsibility of finding alternative devices and architectures that could sustain the historical growth of integrated circuit. The aforementioned necessities have led to the unraveling of various

"emerging technologies" in recent years—technologies that could usher in the "Beyond CMOS" era and ensure continuous evolving of information processing and computing devices (Fig. 1.2). These emerging technologies could be subdivided into three broad categories according to the latest report by ITRS [17]:

- Logic Devices: Carbon Nanotubes (CNT) [18], nanoelectromechanical switches (NEMS) [19].
- Memory Devices: Devices in this category include Resistive Random Access Memory (ReRAM) which this work focusses on, Phase Change RAM (PCRAM) [20, 21], Magnetoresistive RAM (MRAM) [22, 23] and ferroelectric RAM (FRAM) [24, 25], Spin-Transfer-Torque RAM (STTRAM) [26].
- Architectures: Cellular automata, Processor-in-memory, memory-in-logic, computational memory, cognitive computing (machine learning and neuromorphic)

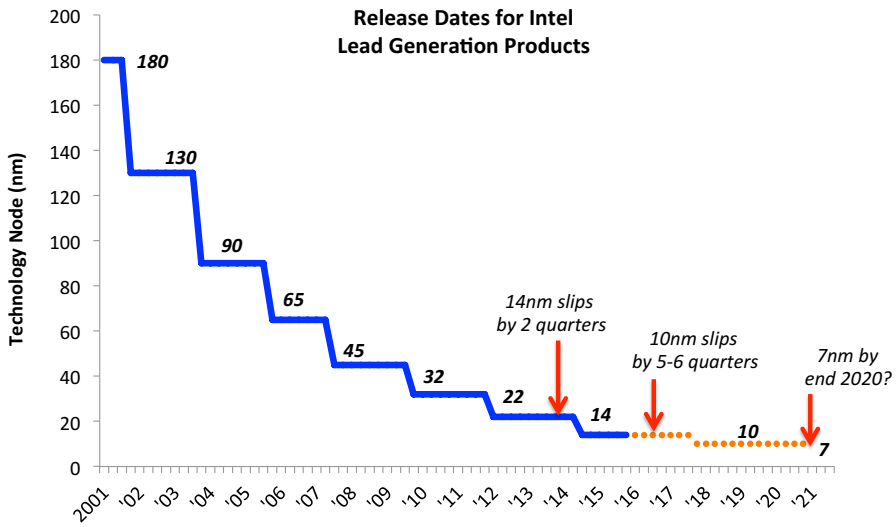


Figure 1.1: Trend showing the transistor scaling timeline of Intel's product in recent years [27].

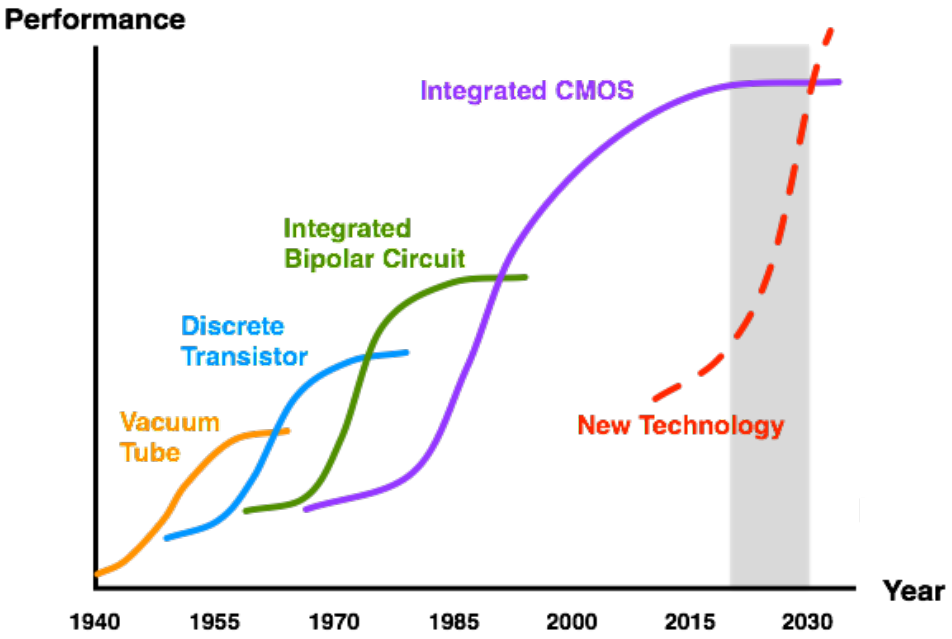


Figure 1.2: Evolvement of the transistors over the years. New technologies have emerged in recent years with the aim of surpassing the popularity of CMOS [27].

1.2 Motivation

As the semiconductor industry moves closer to the inevitable end of transistor scaling, so does the traditional CMOS based systems such as DRAM, SRAM and flash scaling draw closer to their limit [28]. As computing devices get more portable and faster, so does the demand for robust memories increases (Fig. 1.3). A robust memory will incorporate features such as non-volatility, high read/write speed, small surface area, low-power consumption and high density. The industry needs to explore new technologies outside of CMOS to ensure evolvement of computing performances in order to meet present and future needs [17].

Among the emerging technologies previously mentioned, memristive devices which this thesis focusses on have showed great potential in the design of ReRAM, logic circuits, neural networks among many others. This is made possible because of their simple structure and the possibilities of building high density architectures with them [29, 30]. They have better operating speed and high density structure could be achieved through the use of memristor as cross point device in crossbar architecture [31, 32].

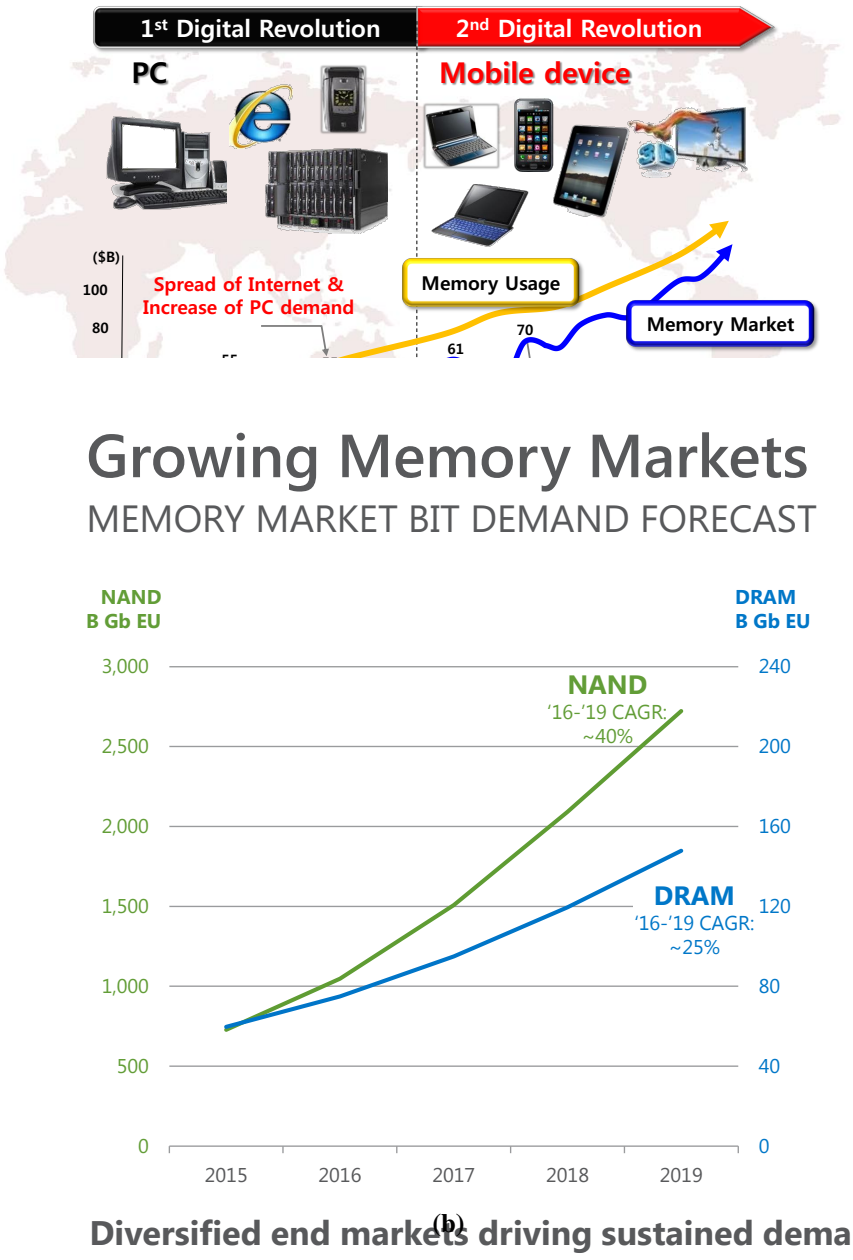


Figure 1.3: (a) Memory usage increases as computing devices get smarter and more powerful [33], (b) Demand for NAND Flash and DRAM has increased greatly over the years [34].

Regarding non-volatile memory design, memristive and other resistive memories bear the hope of extending the validity of Moore's law for several decades, in terms of both memory devices density and performance. As expected with any technology in its infancy, memristor-based architectures are not foolproof. Memristor-based architectures can successfully replace existing technologies if their reliability problems are effectively resolved. Recent research made widely evident that reliability aspects of resistive architectures radically differ from those of their traditional counterparts [35]. Although resistive memories and other architectures are more resilient to transient error but their yield and device fault rate are likely to be higher than conventional architectures [36]. Mem-

ristive architectures are highly susceptible to problems such as resistance drift or loss as memristor represents data as resistance values [37], process variability such as threshold voltage variation [35] and current leakages (sneak-path) during read and write operation that leads to excessive power consumption in the system [38]. All these problems negatively affects the integrity of the data stored and switching ability of the memristor and structures designed with it.

This work aspires to be a key enabler for the effective deployment of emerging memristive technologies in different segments of computing systems by innovations towards improving their weakest aspect: reliability. Once reliability and other related issues are effectively addressed, memristive devices can eventually combine the unique advantages of SRAM (speed), DRAM (density), and flash-memory (non-volatility) as portrayed in Fig. 1.4. Memristive device could also become an integral device in future integrated circuit, thus leading to an almost universal device class that will lead to disruptive effects in the electronics industry for several years to come. The reliability issues of these devices was also echoed by ITRS in their review of emerging devices [39].

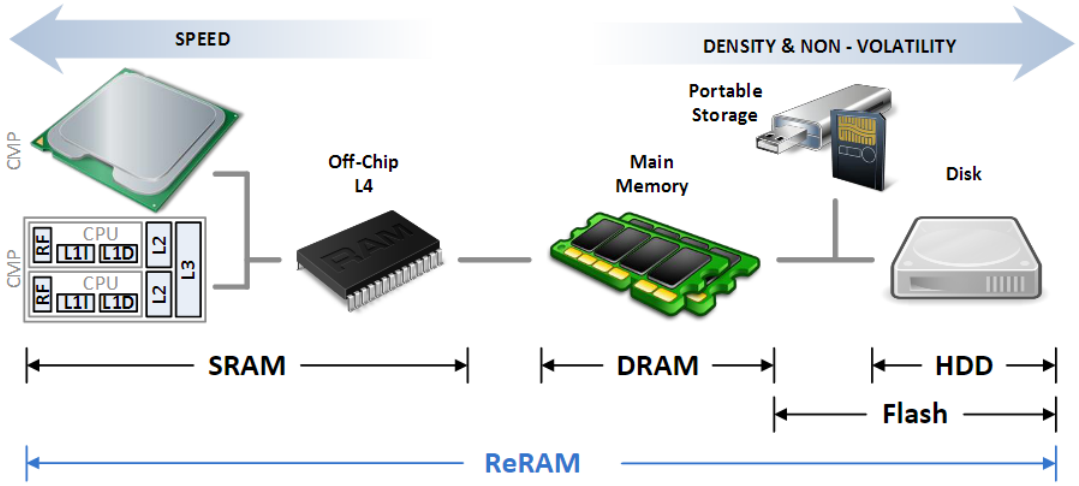


Figure 1.4: Domain of memristor-based Resistive RAM. A memory with the potential to combine the advantages of existing memory technologies if all reliability issues are successfully addressed.

1.3 Methodology and Presentation

This work spans across the research, technology and application categories. The extensive background study and literature review carried out in this work goes beyond theoretical summary of existing work. The background section includes conclusions drawn from the experimental demonstration using appropriate analytic and Electronic Design Automation

(EDA) tools.

This thesis is typeset using the LaTeX document preparation system. TeXShop was used as the user interface (editor) to LaTeX. All the figures in this thesis are either drawn using Microsoft Visio or automatically generated from Cadence Virtuoso EDA tools and MATLAB by the author with the exception of Fig. 1.1, 1.2, 1.3, 1.4 and 2.8.

All design and simulation were done using a combination of spectre-based Cadence Virtuoso 6 and the C++ analytical tools developed by the author.

1.4 Author's Contributions to Subject Area

1. In an effort to prevent read failures in crossbar memories, an accurate modelling of the crossbar architecture is required irrespective of the array size or content of the storage device. The primary aim of this thesisthat floating was to accurately model the crossbar array read schemes without restriction on the resistance value of the crosspoint devices. The overall equivalent resistance model of the crossbar presented works without any restricting assumption [5].

2. A multiple-cells read solution to reduce the overall energy consumption when reading from a memory array is considered. A closed form expression for the noise margin effect is derived and analysis shows that there is zero sneak-path when sensing certain patterns of stored data. The multiple-cells readout method was thus used to analyse an energy efficient Inverted-Hamming (I-H) architecture capable of detecting and correcting single-bit write error in memristor-based memory array [6].

3. The author presented a circuit level analysis of write operation in memristor crossbar memory array with and without line resistance. Three write schemes: floating line, $V/2$ and $V/3$ are investigated. Analysis shows that the floating line scheme could also be considered reliable in arrays with aspect ratio of 1:1 and negligible line resistance just like the latter two schemes. Further analysis also shows that high density crossbar structures cannot be designed using any of the three schemes with worst case line resistance and data distribution within the array. These models are non-restrictive and are suitable for accurate analysis of crossbar arrays and the evaluation of their performance during write operation. The presented analysis

provides the necessary design models that will assist designers in implementation of write techniques in crossbar array in future systems [4].

4. Alongside analysis of three write schemes; floating line, $V/2$ and $V/3$, a novel voltage compensating technique was proposed for write voltage degradation caused by line resistance during write operation on crossbar array. This technique is able to enhance write voltage in the presence of worst case line resistance and thus enable the design of higher density and reliable crossbar array [2]. A new low power multiple bits write technique based on the $V/3$ write was also presented in this thesis.

5. A framework for efficient gas detection using memristor crossbar array is proposed and analysed. A novel Verilog-A based memristor model that emulates the gas sensing behaviour of doped metal oxides is developed for simulation and integration with design automation tools. Using this model, the author proposed and analysed three different gas detection structures based on array of memristor-based sensors. Simulation results show that depending on the organization of the memristive elements and the sensing method, the response of the sensor varies providing a broader design space for future designers. For instance, with a 8×8 memristor sensor array, there is a ten times improvement in the accuracy of the sensor's response when compared with a single memristor sensor but at the expense of extra area overhead [1, 3].

1.5 Outline of this thesis

Subsequent chapters of this project are organised as follow:

Chapter 2 extensively describes the background concepts related to memristor and the crossbar architecture. Existing memristor models are summarised. This chapter also experimentally demonstrated the non-volatility of the memristor as well as its read and write operation. Memristor-based crossbar architecture and the sneak-path current leakages are also explained.

Chapter 3 reviewed existing solutions to the sneak-path problem alongside review of other literature relevant to the contribution of this thesis.

Chapter 4 begins with an accurate remodelling of the crossbar architecture irrespective of the array size or content of the storage device, all in an effort to prevent failures in crossbar memories during the read and write operation. A comprehensive analysis and modelling of four existing crossbar read schemes was carried out. This chapter also presents a novel implementation and analysis of the multiple cells read scheme in memristor-based crossbar array.

In Chapter 5, a novel voltage compensating technique for write voltage degradation caused by line resistance during write operation on crossbar array is proposed. Various circuit level analysis of write operation in memristor crossbar memory array was carried out with and without line resistance. The floating line, $V/2$ and $V/3$ write schemes were further investigated. A new low power multiple bits write technique based on the $V/3$ write was also presented in this chapter.

Chapter 6 presents a framework for efficient gas detection using memristor crossbar array. A novel Verilog-A based memristor model that emulates the gas sensing behaviour of doped metal oxides is developed for simulation and integration with design automation tools. These Chapter benefits from the result obtained from the analysis carried out on the read and write operations in previous chapters.

Chapter 7 concludes and summarises the work presented in this thesis. Other possible extension of the work carried out in this thesis was also identified.

Appendix A includes the data for read-out voltages and read margin in crossbar read operations.

Appendix B includes code listing of the analytic tools developed for the accomplishment of the various tasks in this thesis.

Chapter 2

Background

2.1 Introduction

Memristor was initially formulated and theoretically described in 1971 by Leon Chua [29] in his paper titled "Memristor -The missing circuit element". In his paper, he postulated the existence of a solid state device which represents the missing relationship between the four basic variables. After almost four decades of Leon Chua's postulation, the 'Information and Quantum Systems Laboratory' at HP labs publicised the development of a functional memristor based on Leon Chua's initial description [30].

This chapter introduces the main device that this work relies on - memristor, the fourth circuit element. A brief history of the device, existing models, applications of the device as well as experiments to demonstrate its basic operation will be discussed. A full explanation and simulation of the crossbar architecture for memristors will also be presented here as well as the various problems in this architecture. This chapter details the prerequisite knowledge required for better understanding of the contribution of this thesis presented later on.

2.2 Memristor Overview

The memristor is the latest discovered two terminal fundamental circuit element alongside the existing trio of resistor, inductor and capacitor. The resistor was realised in 1827, the inductor in 1831 and the capacitor in 1745 [40]. The memristor was first theoretically

described by Leon Chua as the missing relationship between flux and charge [29]. He represented it with the symbol in Fig. 2.1.

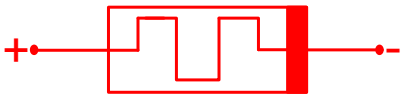


Figure 2.1: Symbol of the memristor originally proposed by Leon Chua.

Leon Chua argued that if the four basic circuit variables (voltage (v), current (i), charge (q) and flux (φ)) are arranged in symmetry as shown in Fig. 2.2, there are six possible relationships. Most of these relationships are already established except for the relationship between flux and charge. Capacitor (C) is defined by the relationship between charge and voltage as $dq = Cdv$, resistor (R) is defined by the relationship between voltage and current as $dv = Rdi$. Similarly, an inductor (L) is defined by the relationship between magnetic flux and current: $d\varphi = Ldi$. The two other relationships are $i = dq/dt$ (current is the time integral of charge) and $v = d\varphi/dt$ (Faraday's law — voltage is the time integral of φ).

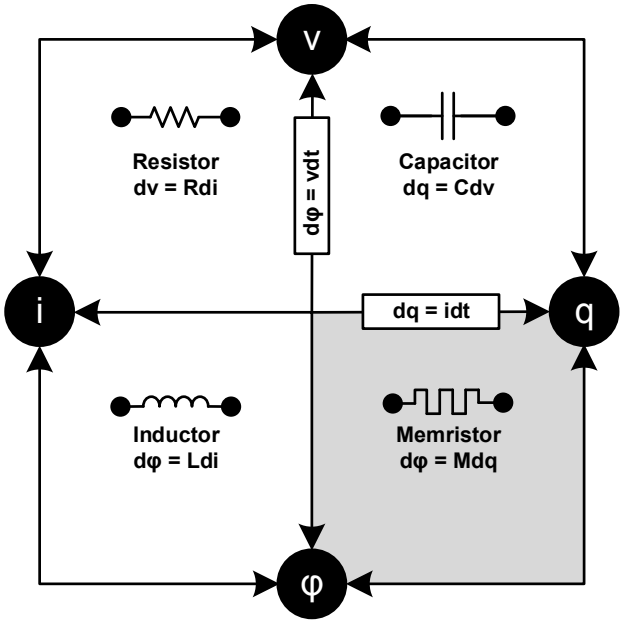


Figure 2.2: Circuit variable (v, i, φ and q) relationships. Adapted from [30].

Based on this argument, Leon Chua concluded that a long missing passive element defined by $d\varphi = Mdq$ must exist, where M stands for the memristance of the missing device. He named the device memristor coined from the words MEMory and ResISTOR because it behaves like a resistor with memory [29]. A memristor thus provides a function between the flux and the amount of electric charge passing through the device as shown in Eqn. 2.1.

$$M = \frac{d\varphi}{dq} \tag{2.1}$$

Flux and charge can be defined by Eqn. 2.2 and 2.3 as they represent the time integral of voltage and current respectively:

$$\varphi = \int v(t)\, dt \Rightarrow \frac{d\varphi}{dt} = v(t) \tag{2.2}$$

$$q = \int i(t)\, dt \Rightarrow \frac{dq}{dt} = i(t) \tag{2.3}$$

A charge dependent memristance can thus be defined as

$$M(q) = \frac{d\varphi/dt}{dq/dt} \tag{2.4}$$

Substituting Eqn. 2.2 and 2.3 into Eqn. 2.4 results in Eqn. 2.5 for a charge dependent memristance similar to resistance.

$$M(q) = \frac{v(t)}{i(t)} \tag{2.5}$$

Similarly, a flux dependent memristance (Eqn. 2.7) just like conductance can be derived from Eqn. 2.6

$$M(\varphi) = \frac{dq/dt}{d\varphi/dt} \tag{2.6}$$

$$M(\varphi) = \frac{i(t)}{v(t)} \tag{2.7}$$

Memristance thus depends on the time integral of voltage or current. In 1976, memristor's scope was extended to a large class of memristive devices and systems [41]. It was also proved in [42] that any resistive device that shows an I-V (current-voltage) characteristics curve in the form of a pinch-hysteresis loop (Fig. 2.3) is regarded as a memristor.

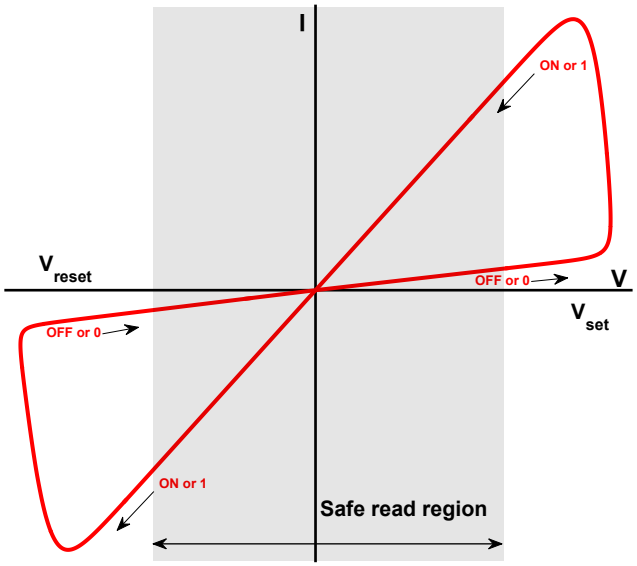


Figure 2.3: A memristor's linear I-V characteristics showing transitioning between high and low resistance state in the presence of sinusoidal voltage source.

Despite the theoretical demonstration of the existence of the fourth circuit element, it remained difficult to physically fabricate a device that exhibits the correct behaviour of a memristor.

2.2.1 HP Labs' Memristor

Despite the clarity of Chua's argument, a practical or experimental confirmation of this missing element remained elusive until 2008 when scientists at HP Labs announced the first physical realisation of Chua's theoretically predicted memristor [30]. HP Lab's memristor consists of a thin film of titanium dioxide (TiO_2) semiconductor sandwiched between two platinum electrodes as shown in Fig. 2.4. The titanium dioxide layer consists of two different regions: a region of stoichiometric TiO_2 and the other region consisting of oxygen deficient TiO_{2-x}. The TiO_{2-x} region is conductive due to the positively charged oxygen vacancies while the former region is less conductive. The overall structure can thus be regarded as a series combination of two resistors as shown in Fig. 2.5. The position of the barrier between the two regions depend on the polarity of the applied voltage at the terminals. As will be discussed later, a positive voltage applied to the surface of TiO_{2-x} region repels the positively charged vacancies further into the TiO_2 region thereby moving the barrier further down in order to make the device more conductive by forming a channel between the two platinum electrodes. A negative voltage attracts the oxygen vacancies out of the TiO_2 region, thereby making the device insulated because of the now dominant TiO_2 layer. In summary, the total memristance of the device is equivalent to the

resistance of the dominant region. When the highly resistive TiO_2 region occupies almost full length D, the device's equivalent resistance is R_{off}. Similarly, when the device is dominated by the highly conductive region, the equivalent resistance of the device is R_{on}.

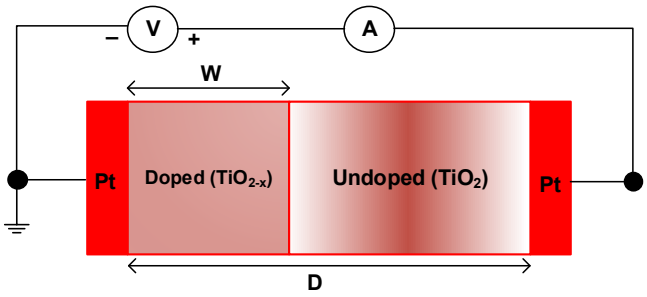


Figure 2.4: Physical structure of the memristor fabricated by HP Labs. It consists of two platinum electrodes and two regions of doped (highly conductive) and undoped (highly resistive) titanium dioxide. w is the state variable and D is the length of the device [30].

A memristor stores data as resistance and the resistance value of the device can be changed by applying a voltage greater than its threshold voltage. This causes the device to switch between High Resistance State (HRS), R_{off} and Low Resistance State (LRS), R_{on} depending on the amplitude, polarity and duration of the voltage applied. The switching from R_{on} to R_{off} is regarded as the RESETing process whereas the transition from R_{off} to R_{on} is known as the SETing process (the write voltage must be greater or equal to the SET (RESET) threshold voltage) as shown in Fig. 2.3. Reading from a memristor involves applying a smaller voltage not sufficient to SET or RESET the cell to one end and sensing the output at the opposite end of the cell. A voltage divider or any other suitable sensing circuit as listed in [32] can be used at the sensing terminal (section 2.2.6). The region shaded grey in Fig. 2.3 represents the safe range for the read voltage where the state of the selected cell cannot be accidentally perturbed.

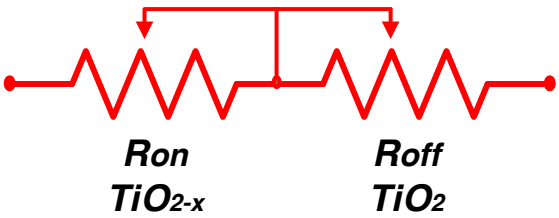


Figure 2.5: Equivalent circuit model of HP Lab's memristor [30].

The mathematical model for the memristor's resistance (memristance) M can be defined as:

$$M(t) = R_{on}\frac{w(t)}{D} + R_{off}\left(1 - \frac{w(t)}{D}\right)\right) \tag{2.8}$$

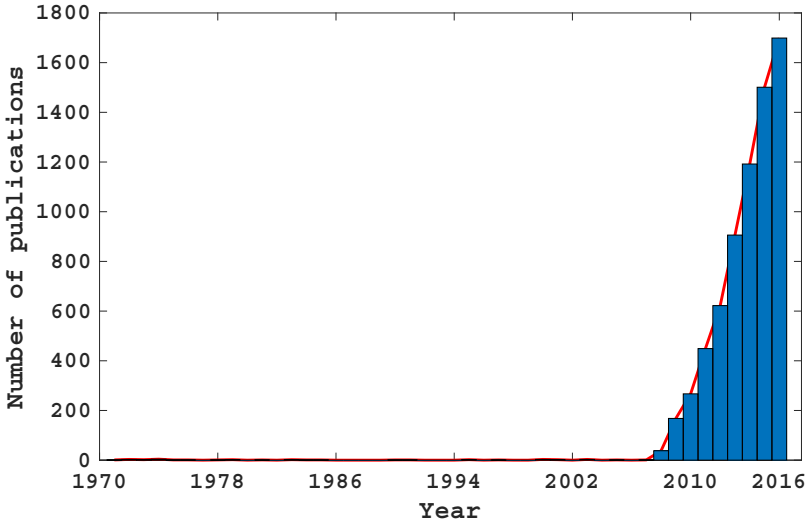
where $w(t)$ is the time dependent state variable and D is the length of the memristor. The state variable w is described by the equations below:

$$\frac{dw(t)}{dt} = \mu_v \frac{R_{on}}{D} i(t) \tag{2.9}$$

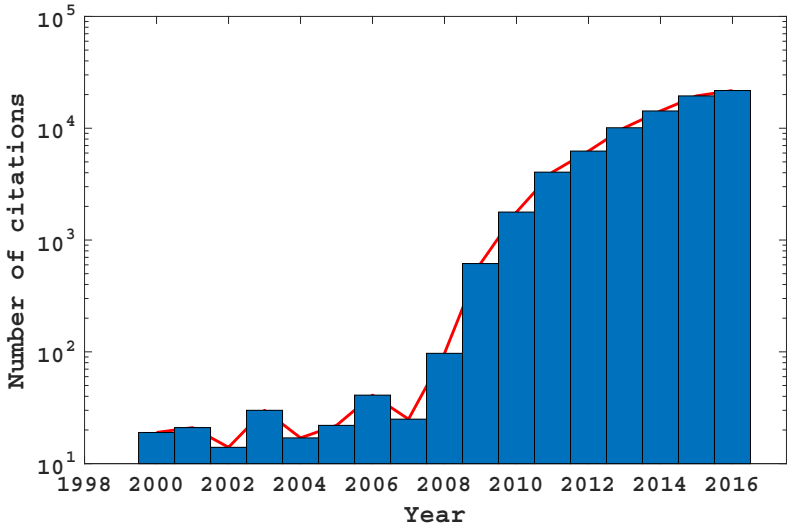
$$w(t) = \mu_v \frac{R_{on}}{D} \int_0^t i(t)dt \tag{2.10}$$

where μ_v is the dopant mobility rate of the switching material.

Since HP Labs discovery of the physical memristor, research on memristor have soared exponentially (Fig. 2.6) especially its memory application in ReRAM.



(a)



(b)

Figure 2.6: (a) Number of publications with memristor keyword (b) Number of citations with memristor keyword. Data obtained from www.scopus.com.

2.2.2 Properties of Memristor

Some major properties which make the memristor-based ReRAM unique and promising include the following:

- **Non-volatility** - The memristor has the ability to retain and remember its last state after power has been switched off for a period of time [43]. It has a variable resistance determined by the time integral of the current flowing through the device such that with zero current, the device exhibits a constant charge that results in a constant resistance of the device.

- **Fast access** - Fast read and write access is essential in non-volatile memory. Memristor has shown a write speed of up to 0.3ns [28]. Fast access time ultimately leads to low power consumption in memory array.

- **High density** - Density measures the amount of binary information that can be stored on a given surface area, high density allows large amount of data to be stored in a small area. According to [43], a capacity of up to $466\text{GB}/\text{cm}^2$ can be achieved with a memristor-based ReRAM for a $5nm$ cell based on a $116\text{GB}/\text{cm}^2$ from a $10nm$ cell [44]. Memristor memories have a footprint of $4F^2$ and can be further reduced to $4F^2/n$ by stacking n-layers of arrays or via 3-D integration, where F is the minimum feature size.

- **Compatibility** - Memristor can also be integrated with existing CMOS transistors to design hybrid CMOS/nano structures that could potentially be used as a universal memory and logic gates [45, 46].

2.2.3 Applications of Memristors

Memristors are two terminal nanoscale devices with unique characteristics, this has resulted in their applications in many areas. Memristors are well suitable for the design of computing memories, memory design is the most widely researched application area of the memristor. Denser memory structures can be designed using memristor as the main memory cell instead of the transistors as it is in conventional memory. Hybrid CMOS/memristor memory structures can also be designed by using CMOS building

blocks for the control and peripheral circuit and memristor as the memory unit [47]. Neuromorphic systems design is another application area of the memristor and it has received a huge boost since the realisation of the memristor. The biological synapse is a two-terminal structure just like the memristor. A synapse acts a conductor of electrical signal between the neurons in the human brain. Authors in [48] and [49] demonstrated the use of memristor to replicate the functionality of a synapse to create efficient neuromorphic hardware with high density and increased connectivity. Controlled voltage pulses are used to change the conductance (acting as weight) of the memristor just like neuron spikes are used to change synaptic weight during the learning process of the neural network. Logic circuit applications of memristors have become increasingly common in recent years [50, 51, 12]. Memristors are able to implement material implication logic such that Boolean logic states are represented as resistance state rather than voltage levels as in CMOS. Memristors have also shown great potential as an efficient sensing device, authors in [52, 53] demonstrated the viability of memristor as a biosensor for dried biological films. In [54], a copper oxide memristor was fabricated and its use as a gas sensor was demonstrated with limited performance. With this limitation in mind, we developed a verilog-A model of a memristor that can be used as a gas sensor in Chapter 6.

Programming the memristor involves the application of a suitable write voltage across it. The write voltage must be equal or greater than the SET (RESET) threshold voltage as shown by the number inequality in Fig. 2.7. Switching of the memristor between the two states can be either unipolar or bipolar. A memristor that exhibits a unipolar switching mode will rely only on the amplitude of the voltage and a switch to '1' or '0' can happen at the same voltage polarity. A bipolar memristor will depend on the amplitude and polarity of the applied voltage [55, 56].

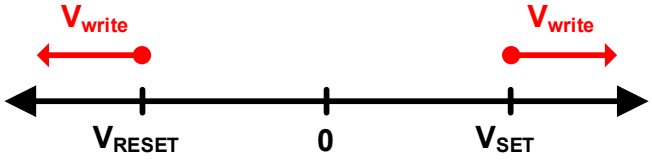


Figure 2.7: Programmable range of the write voltage. The voltage must be equal or greater than the threshold voltage of the memristor.

2.2.4 Writing a Logic 0 to the Memristor

A negative (positive) voltage $V_{write} \leq V_{reset}$ ($V_{write} \geq V_{set}$) is applied to the memristor for a given period of time, the effect of the amplitude and duration of the applied voltage

also depends on the dopant mobility rate (μ_v) of the thin film (Eqn. 2.9). The higher the mobility rate, the faster the memristor reacts to the applied voltage. The state variable w can take on any value between 0 and D (length of doped and undoped region) as shown in Fig. 2.4. However, for the sake of clarity and better understanding, it is assumed that w will be approximately 0 and D for logic **'0'** and **'1'** respectively.

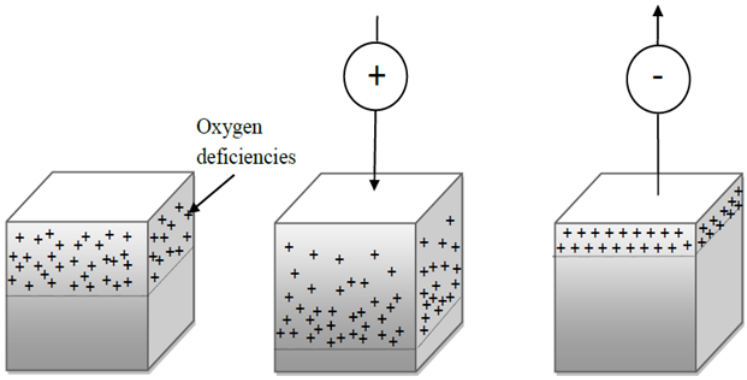


Figure 2.8: Reaction of the memristor cell to positive and negative voltages [43].

For purpose of analysis and simulation in this thesis, a parametric analysis was carried out each time a new model is designed/adapted to ascertain the threshold voltages (V_{set} and V_{reset}) of the memristor and the minimum duration required for the write operation to be completed. In terms of the physical operation of the device during this write **'0'** operation, the negative voltage will attract the positively charged de-oxygenated titanium dioxide (TiO_{2-x}) towards itself, thereby pulling it away from the undoped region (TiO_2) as shown in Fig. 2.8. Since the cell consists majorly of the undoped TiO_2, the overall resistance of the cell is high and it is regarded as state **'0'**.

2.2.5 Writing a Logic 1 to the Memristor

The same concept as writing a logic **'0'** applies to writing a logic **'1'** to the memristor except that a positive (negative) voltage $V_{write} \geq V_{SET}$ ($V_{write} \leq V_{RESET}$) is applied instead. Physically, the positive voltage repels the oxygen vacancies in the TiO_{2-x} region further into the TiO_2 layer, thereby making the device almost completely filled with the more conductive TiO_{2-x} from the surface as shown in Fig. 2.8.

2.2.6 Reading from a Memristor

The read operation is a bit complicated compared to the write operation. The read voltage V_{read} must be set such that it is less than the memristor's threshold voltage so as not to affect the memristance of the cell as shown in Fig. 2.9. A simple and effective way of reading the state of a memristor is through the use of a voltage divider sensing technique. A load resistor (R_{load}) is connected in series to the memristor in order to convert the current from the memristor into a voltage signal. The output voltage (V_{out}) that translates the content of the cell is tapped in between as shown in Fig. 2.10. The output voltage and current is computed according to Eqn. 2.11 and 2.12. It represents the output of the voltage divider consisting of the load resistor and the resistance (R_M) of the memristor itself. Reading from a memristor involves applying a small voltage not sufficient to SET or RESET the cell to one end and sensing the output at the opposite end of the cell.

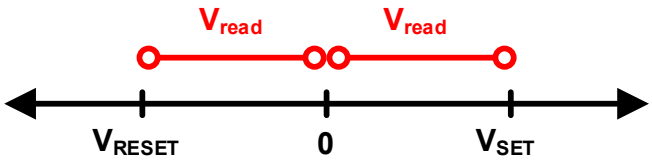


Figure 2.9: Safe range of the read voltage. Reading from a memristor involves applying a small voltage not sufficient to SET or RESET.

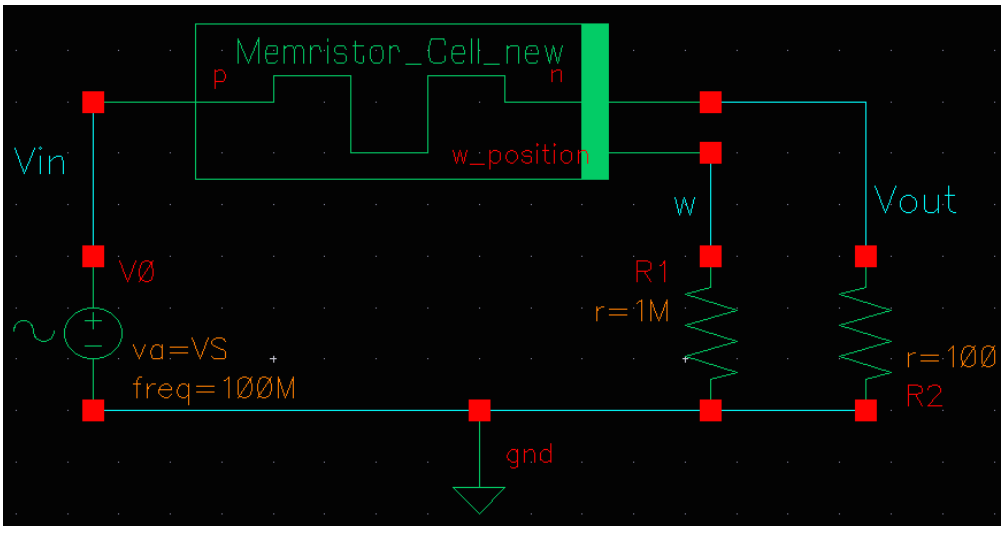


Figure 2.10: Testbench (schematic) of memristor read/write circuit. During the read operation, $R2$ (R_{load}) is set to an optimum value to act as a voltage divider with the resistance of the memristor, $Vin = V_{read}$. For the write operation $R2$ is set to 0 and $Vin = V_{write}$. An input voltage of 1V is used V_{read} and 3V for V_{write} unless otherwise stated.

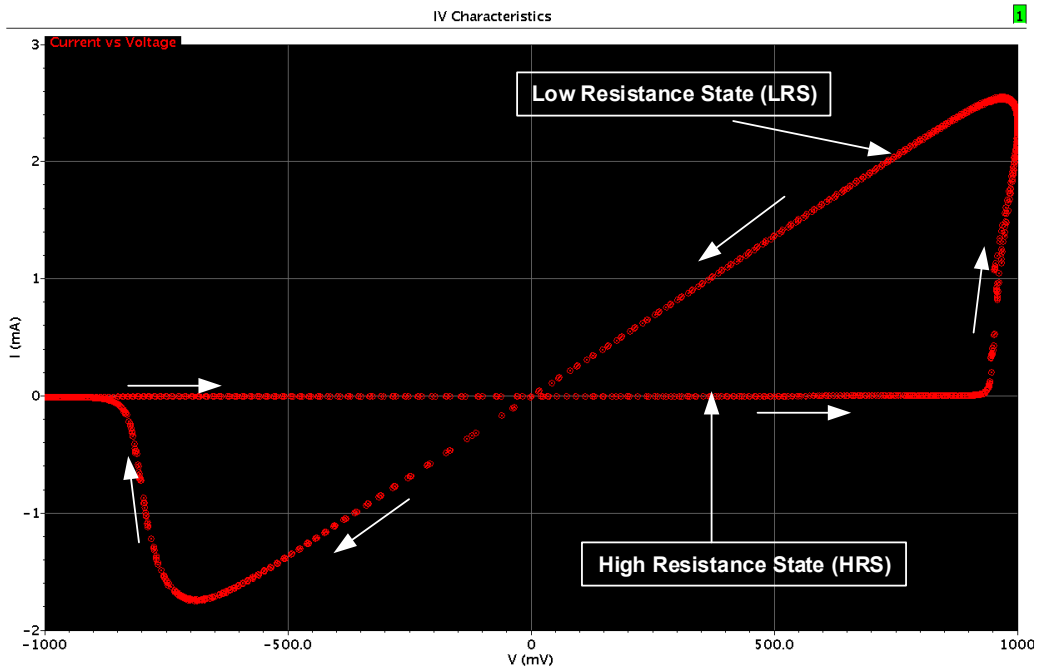


Figure 2.11: *I-V* **characteristics of the Memristor generated from the schematic in Fig. 2.10. Simulation done on Cadence Virtuoso with the following parameters;** $R_{off} = 200K\Omega$**,** $R_{on} = 100\Omega$**,** $Vin = 1V$**,** $V_{reset} = V_{set} \approx |0.9V|$**.**

$$V_{out} = V_{read} \times \frac{R_{load}}{R_{load} + R_M} \tag{2.11}$$

$$I_{out} = \frac{V_{read}}{R_{load} + R_M} \tag{2.12}$$

where $R_M \in \{R_{off}, R_{on}\}$ represents either the off or on state of the memristor.

2.2.7 Demonstration of Memristor's Non-volatility

A stated earlier, the memristor does not need power to retain information earlier stored in it; this section demonstrates this ability through a simulated experiment. Several models of the memristor havde been proposed to simulate the physical behaviour of the memristor [30, 57, 58]. In this experiment, we used the linear ion drift model described by HP labs' as it demonstrates characteristics similar to what was postulated by Leon Chua in [29].

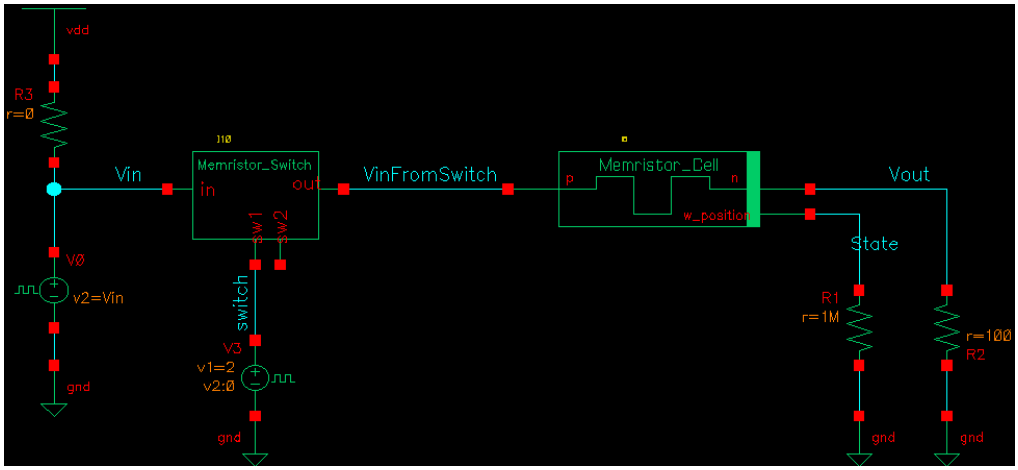


Figure 2.12: A schematic to show the non-volatility of the memristor, a switch is attached to the memristor to turn off power supply after 4ns.

Fig. 2.12 shows the schematic of the test bench used for the simulation. It consists of a memristor with a connected power supply switch. The switch was designed to be ON initially then go OFF after a 4ns to ascertain the reaction of the memristor cell to power supply cut-off.

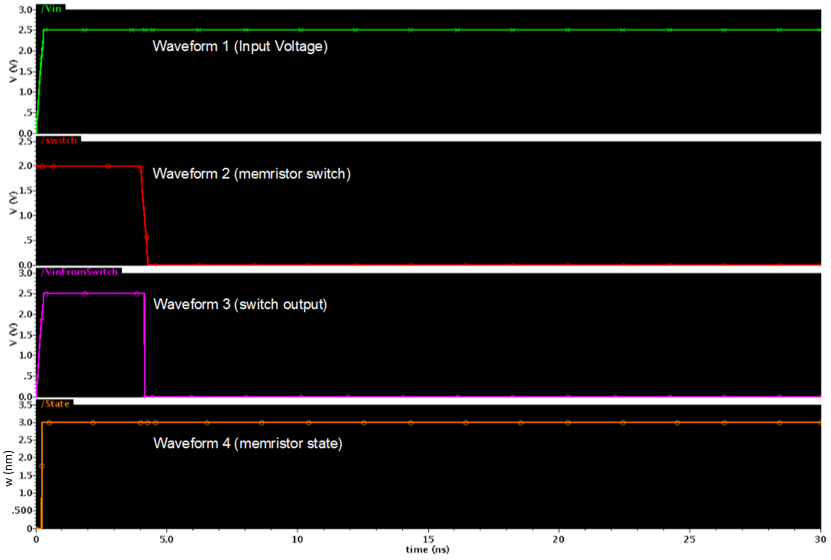


Figure 2.13: Waveform showing the non-volatility of the memristor.

The first 4ns shows a high input voltage (waveform 1), a ON switch (waveform 2), a high switch output (waveform 3) and the memristor switching from HRS (OFF) to LRS (ON) (waveform 4) as a result (Fig. 2.13). When the switch goes OFF after 4ns and power supply to the memristor is cut off, the memristor still retained its previous high state. With the experiment above, the non-volatile ability of the memristor model has been successfully validated.

2.2.8 Memristor Models and Window Functions

Although the rate of research on memristor kept soaring in recent years, there is however no general consensus on the best memristor model for industrial and research purpose. There are various memristor models in the literature which attempt to mimic its physical description as much as possible. Various memristor models have been developed in the past. They are discussed in this section. Table 2.1 shows a comparison of the models. For a memristor model to be termed effective, it has to meet the following criteria [59]:

1. Accurate and computationally efficient.
2. Generic so as to fit different types of memristive applications.
3. Must work with thresholds (voltage and/or current) to enable accurate read and write.

Memristor Models	**Linear Ion Drift Model [30]**	**Non-Linear Ion Drift Model [60]**	**Simmon Tunnel Barrier Model [61, 62]**	**TEAM [59]**	**VTEAM [63]**
State variable (w)	$0 \leq w \leq D$	$0 \leq w \leq 1$	$a_{off} \leq w \leq a_{on}$	$x_{off} \leq w \leq x_{on}$	$w_{off} \leq w \leq w_{on}$
I-V characteristics	Explicit	Explicit	Ambiguous	Explicit	Undefined (freely chosen)
Threshold mechanism	Current	Voltage	Voltage	Current	Voltage
Matched theorized definition	Yes	No	No	Yes	Yes

Table 2.1: Comparison of memristor models.

2.2.8.1 Window Functions

The main aim of using a window function $f(w)$ is to ensure that the state variable w does not go beyond its specified boundary and ensures its non-linearity. Table 2.2 lists the major window functions; Biolek [57], Joglekar [64], Prodromakis [65], TEAM [59], VTEAM [63] and their properties.

Table 2.2: Comparison of window functions.

Window Function	Biolek	Joglekar	Prodromakis	TEAM	VTEAM
Function $f(w)$	$1-(w/D-stp(-i))^{2p}$	$1-(2w/D-1)^{2p}$	$j(1-[(w-0.5)^2+0.75]^p)$	$exp[-exp(\lvert x-x_{on,off}\rvert W_c)]$	Eqn. 2.13
Maintains Boundary Conditions	Yes	No	Yes (Practically)	Yes (Practically)	Yes (Practically)
Compatible Memristor Models	Linear, Nonlinear ion drift & TEAM	Linear, Nonlinear ion drift & TEAM	Linear, Nonlinear ion drift & TEAM	TEAM	VTEAM
Symmetric	Yes	Yes	Yes	No	No

$$\frac{dw(t)}{dt} = \begin{cases} k_{off} \cdot \left(\frac{v(t)}{v_{off}} - 1\right)^{\alpha_{off}} \cdot f_{off}(w), & 0 < v_{off} < v \\ 0, & v_{on} < v < v_{off} \\ k_{on} \cdot \left(\frac{v(t)}{v_{on}} - 1\right)^{\alpha_{on}} \cdot f_{on}(w), & v < v_{on} < 0 \end{cases} \tag{2.13}$$

2.3 The Crossbar Architecture

The crossbar architecture consists of a set of parallel nanowires perpendicularly placed on another set of parallel nanowires with a memristor cell inserted at every intersecting point of the wires [38, 66, 67]. Crossbar architecture offers the possibility of building a highly dense device structure, it's meant to enable independent access to each device without interference from other devices in the array. Cells in the same row are connected together by one of the nanowires and cells in the same column are connected together by another nanowire. In a crossbar array, the content of each cell can be sensed or programmed by applying a balanced voltage to the row (column) that has the desired cell and grounding the corresponding column (row). All the other unselected lines can be left floating or partially biased as discussed in later sections.

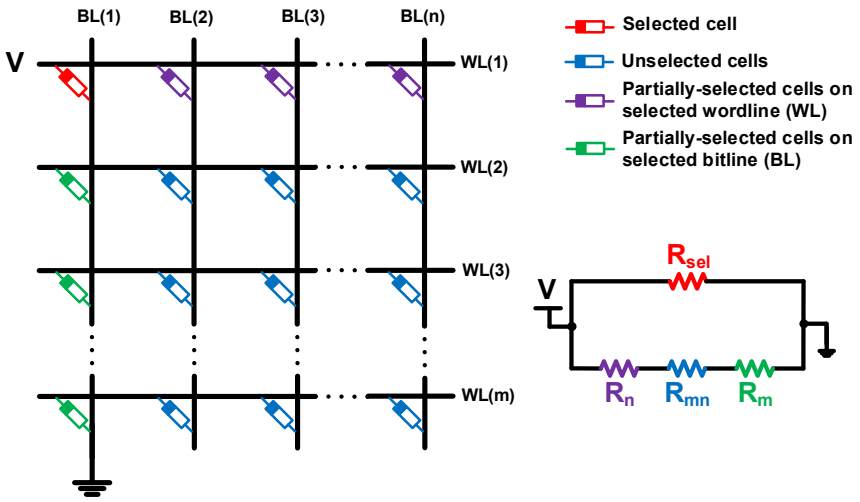


Figure 2.14: An $m \times n$ memristor-based crossbar memory structure set-up for write operation and its equivalent circuit model that depict the effect of all resistances in the array during the real operation. R_{sel} represents the selected cell $R_{1,1}$. R_m and R_n represent the row and column partially-selected cells respectively and R_{mn} simplify the cells in the unselected lines [68].

Fig. 2.14 shows a typical m rows (wordlines) and n columns (bitlines) crossbar array of memristors. Each memristor is represented by $R_{i,j}$, where i and j are the row and column index respectively. Assuming memristor $R_{1,1}$ in red is selected for read/write operation, cells on the same wordline $WL(1)$ and bitline $BL(1)$ are classified as *partially-selected* cells while others are unselected. In this thesis, all unselected and partially-selected cells are classified together as *unselected* cells except otherwise stated. In summary, during read/write operation on a single cell, the crossbar array can be categorised into four groups of memristors [68]. The groupings are made possible because all memristors in each group are in parallel with each other. If line resistances are ignored, the overall resistance of any $m \times n$ crossbar array can be simplified to the equivalent circuit shown in Fig. 2.14 with four resistance values.

- Group I (R_{sel}): Cell selected for writing (red).
- Group II (R_m): Partially-selected $m-1$ cells on the selected bitline (green).
- Group III (R_n): Partially-selected $n-1$ cells on the selected wordline (purple).
- Group IV (R_{mn}): Unselected $(m-1)(n-1)$ cells that are neither on the selected bitline nor wordline (blue).

$$R_m = \frac{R}{(m-1)} \tag{2.14}$$

$$R_n = \frac{R}{(n-1)} \tag{2.15}$$

$$R_{mn} = \frac{R}{(m-1)(n-1)} \tag{2.16}$$

Eqn. 2.14 - 2.16 are the closed form formula for each group of resistance as detailed in [68]. Each equation is derived from the assumption that all memristors in each group have a similar resistance value of R. These equations can be further enhanced to capture the more realistic case of each memristor having a value of R_{on} or R_{off}, this will be presented in Chapter 4.

As mentioned earlier, the memory application of the memristor is its most explored area. The crossbar architecture offers the possibility of designing a memristor-based universal memory [30] - a memory that combines the density of DRAM, non-volatility of Flash and speed of SRAM among others. The semiconductor industry has been charged with the task of exploring new technologies in order to ensure evolvement of computer memories. Memristor-based resistive RAM has demonstrated potentials capable of replacing the existing transistor-based memories. Other emerging and existing memory technologies are shown in Fig. 2.15 and compared in Table. 2.3.

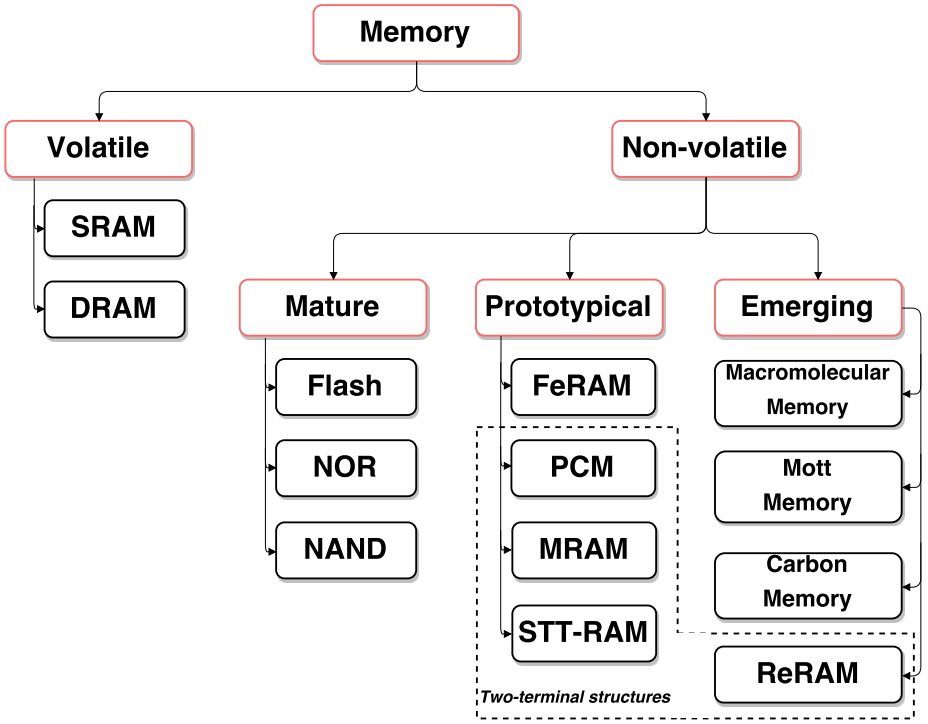


Figure 2.15: Types of memories including emerging ones.

	Traditional Memories				Other Emerging Memories			
	DRAM	**SRAM**	**NOR Flash**	**NAND Flash**	**FRAM**	**MRAM**	**PCRAM**	**ReRAM (Memristor)**
Cell Element[a]	1T1C	6T	1T	1T	1T1C	1(2)T1R	1T(D)1R	1R
Feature Size (nm)	36-65	45	90	22	180	65	45	<5
Density $(Gbit/cm^2)$	0.8-13	0.4	1.2	52	0.14	1.2	12	154-309
Read Time (ns)	2-10	0.2	15	100	45	35	12	<10
Write Time (ns)	2-10	0.2	10^7	10^6	65	35	100	0.3
Retention Time	$4-64ms$	N/A	10 Yrs	10 Yrs	10 Yrs	>10 Yrs	>10 Yrs	>10 Yrs
Endurance (cycles)	$>10^{15}$	$>10^{15}$	$>10^5$	$>10^5$	$>10^{12}$	$>10^{15}$	$>10^9$	$>10^12$
Non-volatile	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Multilevel Capacity	No	No	Yes	Yes	Yes	Yes	No	Yes

Table 2.3: Comparison between Conventional Memories and Emerging Memories [69].

[a] C-Capacitor, T-Transistor, R-Resistor, D-Diode.

2.3.1 Write Operation in Crossbar Array

As described previously, in the case of a single memristor, the memristor is in the ON state when it is fully doped ($w \approx D$) and OFF when undoped ($w \approx 0$). A voltage $V_{write} = V_{SET}$ switches the memristor from the OFF state to the ON and $V_{write} = V_{RESET}$ switches it from ON to OFF state. V_{RESET} and V_{SET} are of opposite polarity. A memristor that is already in an OFF (ON) state will not react when V_{RESET} (V_{SET}) is applied. In conventional write operation on crossbar memory with memristor as the storage unit, the write voltage is applied to the selected row (column) while the selected column (row) is grounded or biased. The voltage across the memristor must be greater than its threshold voltage in order to programme the memristor as shown in Fig. 2.16. The conventional write operation is not very effective as partially-selected memristors can experience resistance loss (partial change of state) as a result of the half voltage reaching them [37]. The $V/2$ and $V/3$ write scheme was first introduced in [70]. These schemes bias the floating lines to result in a maximum voltage of $V_{write}/2$ and $V_{write}/3$ across the unselected memristors respectively. These write schemes will be further explored in Chapter 5 in addition to the proposal of novel techniques to improve the reliability of crossbar write operation.

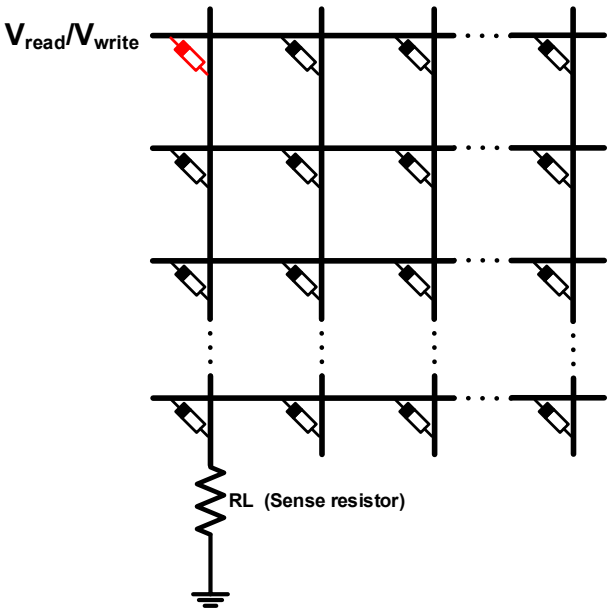


Figure 2.16: Setup of read/write operation in memristor crossbar architecture.

2.3.2 Read Operation in Crossbar Array

In order to read the data stored by a memristor in a crossbar array, a read voltage $V_{RESET} < V_{read} < V_{SET}$ is applied to the selected row (column) and the selected column (row) is grounded such that the voltage drop across the memristor is insufficient to switch the memristor (see Fig. 2.16). The output can be sensed using most of the existing sensing circuits [32]. This work uses a basic voltage divider technique for converting the current into voltage signal at the output. An output voltage of approximately zero indicates a memristive state of OFF (logic '0') while a voltage approximately half of the write voltage indicates an ON (logic '1') memristor. The crossbar architecture suffers from current leakage paths during read operation in the absence of a supportive gating device thereby making it difficult to independently access each crosspoint memristor without extra circuitry. In practical applications, current sneaks through cells parallel to the desired cell(s) and severely degrades the read output and makes it difficult to differentiate between the resistance states of the desired cell(s). The effect also becomes severe as the array size and the number of ON memristors in the array increases.

2.3.3 Sneak-path Leakage in Crossbar Array

As good as the crossbar architecture is, it has a major limitation with the use of memristor; it is prone to current leakages (sneak-path). Sneak-paths are undesired current paths

through neighbouring cells when a particular cell in the crossbar is selected for read or write operation as portrayed in Fig. 2.17 [38]. Effect of the sneak-path issue includes high power consumption, limitation on the maximum array size, decline in the effective read margin among others. As cells in each row are connected by their top electrode and cells in each column are connected by the bottom electrode, sneak-path becomes a major challenge when designing high density crossbar structures. Sneak-paths particularly lead to erroneous reading of data stored in a memory cell because of interference of undesired currents from the unselected neighbouring cells. Resistance of unselected cells combine to form a parallel resistance path with the resistance of the desired cell as illustrated by the equivalent circuit in Fig. 2.17. I_{sneak} (the current through the unselected cells) is data-dependent as well as array size dependent. A larger array with more low resistance unselected cells will result in high I_{sneak} value. During write operation, sneak-paths can cause unselected cells to be accidentally overwritten. One can easily conclude that all other problems associated with this architecture are a rippling effect of the sneak-path issue.

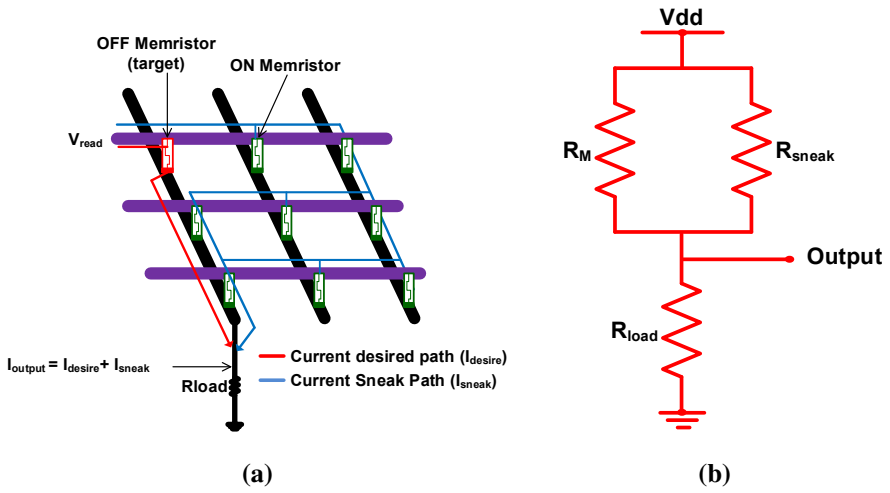


Figure 2.17: Demonstration of sneak-paths effect in crossbar architecture (a) Reading from a cross-bar Array. I_{desire} is the current of the desired cell (Red cell) while I_{sneak} is the sneak-path current from undesired cells, both currents combine to form I_{output} which is not the desired output (b) Equivalent circuit model of sneak-paths effect.

The floating row and column biasing scheme is applied to the schematic in Fig. 2.16 and 2.17. The selected row and column are connected to the read voltage (V_{read}) and ground respectively and other lines are left floating. The sensing mechanism in this structure is the voltage divider formed by the selected cell and the connected load resistance (R_L). The output voltage should ideally depend largely on the content of the selected cell in red but for sneak-path which causes other unselected cells to contribute to the output. The read

voltage V_{read} was designed to satisfy the condition in Fig. 2.9 depending on the direction of flow of current in order to prevent accidental corruption of the cell.

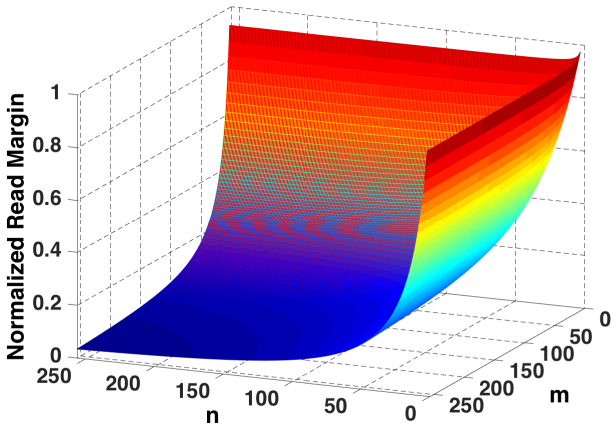


Figure 2.18: Read margin degrades rapidly as array size increases in the presence of sneak-path as computed using Eqn. 2.19.

The total sneak-path resistance R_{sneak} in the array is a series combination of R_m, R_n and R_{mn} according to Eqn. 2.17. R_m, R_n and R_{mn} are explained in Section 2.3 and defined by Eqn. 2.14, 2.15 and 2.16

$$R_{sneak} = R_m + R_n + R_{mn} \tag{2.17}$$

The sneak-path resistance R_{sneak} combines in parallel with that of the selected cell R_{mem} (R_{off} or R_{on}) to result in the effective resistance that forms a voltage divider with the load resistance R_L.

$$R_{mem\_eff} = R_{mem} || R_{sneak} \tag{2.18}$$

The read margin (ΔV) parameter represents the figure of merit that makes it possible to differentiate between the two possible logic states 0 (R_{off}) and 1 (R_{on}) of a memristor cell. The sensing margin is calculated as the difference between the output voltage of the R_{on} and R_{off} states of the memristor as shown below:

$$\Delta V = V_{out}^{1} - V_{out}^{0} \tag{2.19}$$

where V_{out}^{1} and V_{out}^{0} are the output when R_{mem} = R_{on} and R_{off} respectively. Ideally, V_{out} should be computed according to Eqn. 2.20 where the output is strictly dominated by the selected memristor and the known load resistor and not Eqn. 2.21 but for sneak-paths. Fig. 2.18 shows the effect of sneak-path on the read margin of crossbar arrays. The read margin deteriorates rapidly as the size increases, the worst and best case scenarios for

sneak-path are discussed in section 4.4.

$$V_{out} = V_{read} \times \frac{R_L}{R_L + R_{mem}} \tag{2.20}$$

$$V_{out} = V_{read} \times \frac{R_L}{R_L + R_{mem\_eff}} \tag{2.21}$$

The maximum array size for a crossbar in the presence of sneak-path will be derived in section 4.2.

2.4 Summary

This chapter extensively describes the background concepts related to memristor and the crossbar architecture. Existing memristor models are summarised. This chapter also experimentally demonstrated the non-volatility of the memristor as well as its read and write operations. Memristor-based crossbar architecture and the sneak-path current leakages are also explained in this chapter. The next chapter will review existing related work that are relevant to the contribution of this thesis.

Chapter 3

Review of Related Work and Baseline Research

This section reviews some of the existing architectures and techniques for ensuring reliability of the memristor-based devices in addition to those discussed in the background section. Literature involving memristor was almost non-existent before the physical realisation of the device by HP labs in 2008. Section 3.1 presents a review of some of the existing solutions to the read and write problem with memristor-based crossbar arrays. Section 3.2 reviews a recent emerging application area of the memristor - gas sensing. Most of the literature items in Section 3.2 are related to the use of metal-oxide in gas sensing which set a strong motivation for the use of TiO_{2-x}-based memristor for gas sensing.

3.1 Read and Write Operations in Memristor Based Crossbar Arrays

As earlier mentioned, reliability of the read and write operation in memristor based crossbar arrays is critical for the commercial realisation of memristors in mainstream devices. Several techniques have been proposed towards achieving improved reliability for read and write operation in crossbar arrays. One major factor affecting the reliability of read and write operation in crossbar array is the sneak-path problem described in Section 2.3.3. Write operation particularly suffers from the voltage loss problem. This problem arises from the voltage degradation due to line resistances in the crossbar array [71]. The ob-

vious solution to the voltage drop problem caused by line resistance will be to increase the write voltage but this also increases the maximum voltage reaching the unselected cells [72]. One approach to solving this problem is the double-sided ground biasing (DSGB) approach where both sides of the selected wordline array are grounded and the selected bitline is connected to the write voltage [73]. Another solution uses the *dual voltage source* design where voltage is delivered via both sides of the selected wordlines [71]. However, both techniques incur additional chip area overhead thereby reducing the array efficiency without offering much in terms of preventing voltage degradation. Chapter 4 and 5 are dedicated to read and write operations in crossbar arrays respectively. In these chapters, various solutions are proposed to some of the problem that arises directly and indirectly from the sneak-path and voltage loss problem.

3.1.1 Sneak-path Elimination Techniques

Various methods have been proposed for addressing the sneak-paths problem in crossbar architectures. Each of these methods have their pros and cons and there is not a generally accepted solution to this problem without tradeoff of one or more of the performance metrics. This section presents a review of some of these methods. The existing sneak-path elimination methods can be subdivided into three categories as explained in the following sections. Table 3.1 presents a summary of all the discussed techniques.

3.1.1.1 Additional Crosspoint Device

As mentioned earlier, sneak-path can be prevented by including an isolating device alongside the memristor at every crosspoint of the crossbar, in the absence of an access device, sneak-path becomes more prominent and current flows freely within the array [74]. In this category, there is the one transistor and one memristor (1T1M) memory structure [75], one diode and one memristor (1D1M) [76], and one memistor and one memristor (1M1M) [77]. These techniques are able to minimise sneak-path in the array but at the expense of area per cell, array density, 3-D integration, fabrication issues as well as power inefficiency among other shortcomings.

One Transistor and One Memristor (1T1M)

The aim of this technique is to isolate/block unselected memristor cells during write/read operation so that current does not flow through them. This is achieved according to [78]

by connecting a CMOS transistor in series with every memristor in the array as shown in Fig. 3.1. The 1T1M structure was modelled after the DRAM setup but for capacitor that is being replaced with a memristor in this case. This architecture prevent sneak-paths by restricting current access to other cells but the selected cell. If the first element in the first row of Fig. 3.1 is selected for read, the transistors in the other rows are switched off. Although the transistors in the first row are on, the current reaching the other unselected cells in this row are shunted out and a complete circuit like Fig. 2.14 is impossible. Even though this technique minimises sneak-path, the major drawback to this technique is the extra area that will be used up by these added transistors; this has an adverse effect on the array density and footprint of the architecture. The power consumption of this architecture is minimal as it is only the selected cells that have current paths.

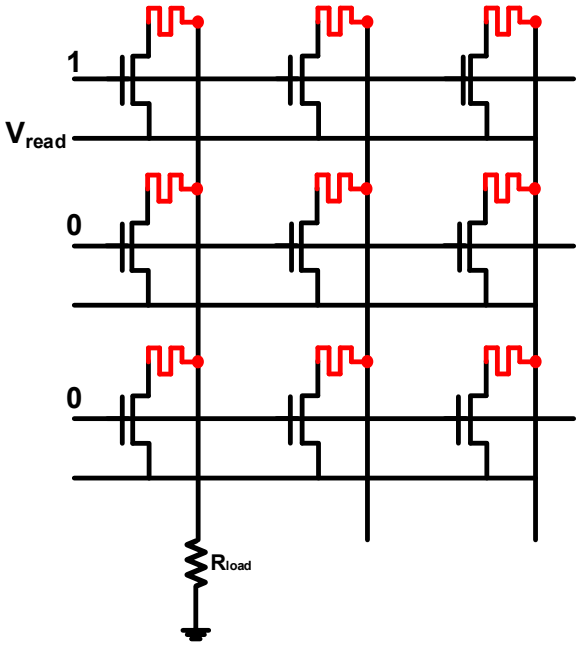


Figure 3.1: The 1T1M structure.

One Diode and One Memristor (1D1M)

The 1D1M is another technique similar to the 1T1M above that solves the sneak-path problem. This technique was introduced in [76] as a preamble to the design of hybrid CMOS/Nano multi-level memristor memory (MLMM) device and it works by connecting a diode in series with each memristor at every cross point as shown in Fig. 3.2. It minimises leakages by making the combination (diode and memristor) unidirectional as memristor alone are bidirectional. If the first element in the first row of Fig. 3.2 is selected for read, current does not flow though the other memeristors in unselected row because of the opposing diodes. Although current flow through the other unselected memristors in

the selected row but they are eventually shunted out and a complete circuit like Fig. 2.14 is impossible. However, as good as the technique looks, these diodes introduce parasitic capacitances into the structure and this causes extra delay as well as a reduced output swing due to the threshold voltage of the diode [45].

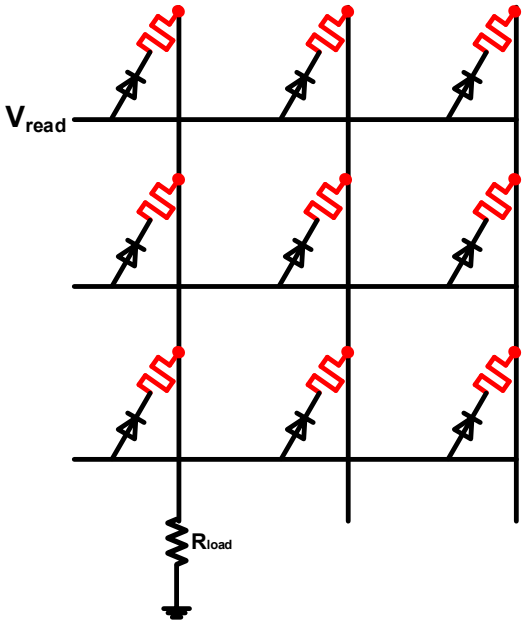


Figure 3.2: The 1D1M structure.

One Memistor and One Memristor (1M1M)

The memistor was introduced before the memristor by Prof. Bernard Widrow in 1960 for use in adaptive circuit. A memistor is a three-terminal device that has the resistance between two of its terminals controlled by instantaneous control current in the third terminal, but by the time integral of the current [79]. Meaning that the memistor can retain its state without an active bias on the third terminal. In this gating method as proposed in [77], a memistor was connected in series with a memristor just like it was done with a transistor and diode but with the addition of an extra column in order to connect the third terminal (Fig. 3.3). In order to access a desired cell, the memistor connect to the memristor cell is turned 'ON' with all others turned OFF, meaning that all other undesired cell where sneak-path could occur are always in high resistance (includes the resistance of the 'OFF' memristor itself and the resistance of the other two terminals of the memistor) to greatly limit the effect of sneak-paths. This technique consumes less power because of the bias-less switching of the memistor but extra wires introduced because of the third terminal of the memistor will introduce extra line resistance.

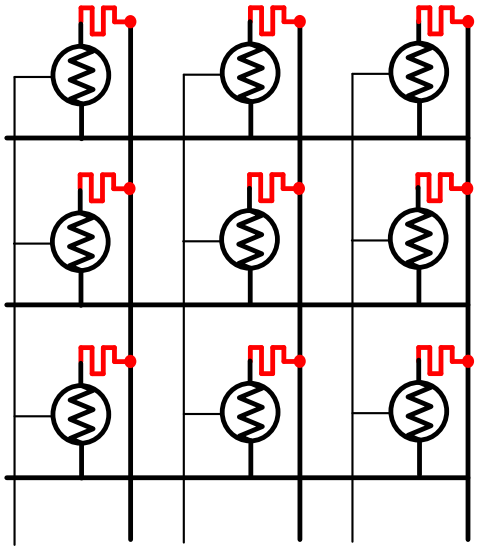


Figure 3.3: The 1M1M structure.

Complementary Resistive Switch

The Complementary Resistive Switch (CRS) introduced in [38] seems to have gained more popularity over others in recent times as it is devoid of the weaknesses associated with other crosspoint techniques. Rather than combine the usual active device with the memristor as in the case of 1T1M and 1D1M, this method instead combines two memristor cells together anti-serially into a CRS (Fig. 3.4). The logic states of the CRS result in high resistance values ($R_{off} + R_{on}$) to ensure sneak-path is reduced to a minimal level. There will be less flow of leakage current through the neighbouring cells because of their high resistances. The CRS method offers much reduced area per cell than the other access device but could be susceptible to resistance loss [37]. The CRS is further discussed in section 3.3.

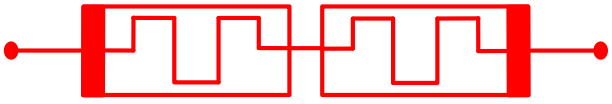


Figure 3.4: Symbol of the CRS.

3.1.1.2 Multi-step Reading

Multi-step reading involves executing the read operation over a minimum of two or more stages before the correct content of the desired cell can be correctly determined. In [80], the authors introduced a two-step read operation where the first step senses the leakage current only and the second step senses the overall current. The difference between

the output of both stages represent the current from the desired cell. Other multi-step techniques were reported in [81] with a three steps read and in [82] with seven steps. Multi-step reading techniques however have a downside to it as it introduces extra sensing circuitry and increases the average reading time.

3.1.1.3 Line Biasing

Sneak-path can also be minimised via the voltages applied to the word and bit lines. Authors in [76, 83] discussed four different techniques in this category: Floating rows and columns (FRFC) (Fig. 3.5(a)), grounded columns and rows (GRGC) (Fig. 3.5(d)), grounded columns and floating rows (FRGC) (Fig. 3.5(b)) and floating columns and grounded rows (GRFC) (Fig. 3.5(c)). These biasing techniques could however not ensure a sufficient read margin beyond an array size of 64×64 [81]. GRGC, GCFR and FCGR also consumes excess power because of their connection to the ground. These techniques will be further investigated in Section 4.3. Authors in [37] proposed a two-step write scheme to minimise the effect of sneak-path in the crossbar array by timing the duration of write voltage application using additional control circuitry. Authors in [71, 73] also propose a technique to apply voltages from both sides of the crossbar during the write operation in order to compensate for current lost to line resistance.

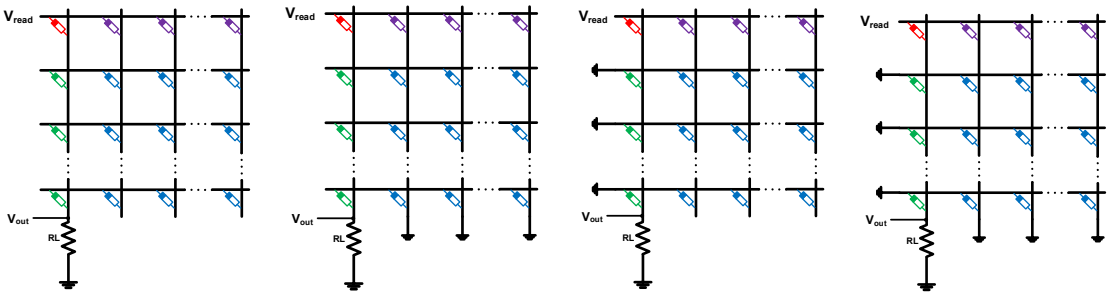


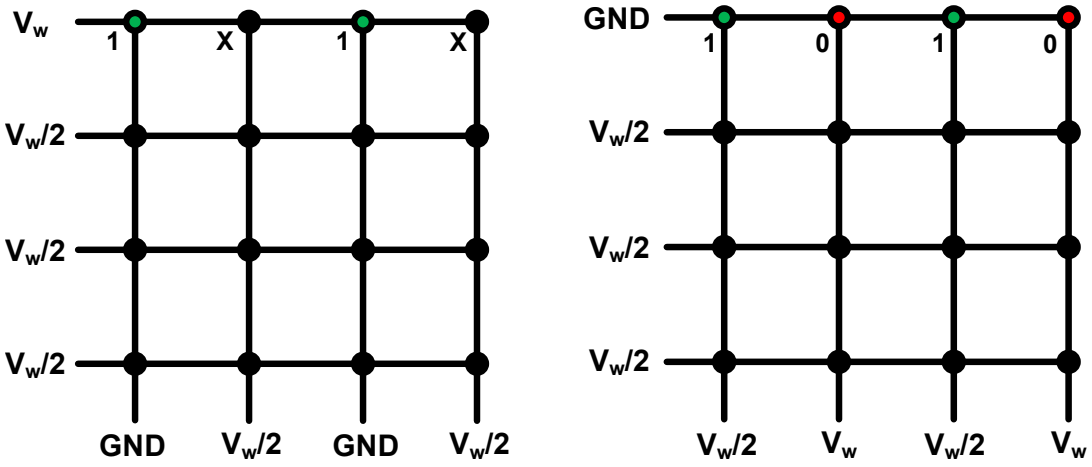
Figure 3.5: Setup of the four line biasing methods for unselected lines (a) Floating rows and floating columns (b) Floating rows and grounded columns (c) Grounded rows and floating columns (d) Grounded rows and grounded columns.

Table 3.1: Comparison of various sneak-path prevention techniques.

Technique	Power Consumption	Area Efficiency	Speed	Ease of Fabrication	Number of Devices	Extra Circuitry
1T1M	Average	Poor	Average	Fairly complex	2	Yes
1D1M	Average	Poor	Average	Fairly complex	2	Yes
1M1M	Average	Poor	Average	Fairly complex	2	Yes
FRFC	Good	Good	Fast	Easy	1	No
FRGC	High	Good	Fast	Easy	1	No
GRFR	High	Good	Fast	Easy	1	No
GRGC	High	Good	Fast	Easy	1	No
CRS	Moderate	Good	Average	Fairly Complex	2	No
Multistep reading	Moderate	Average	Slow	Easy	1	Yes

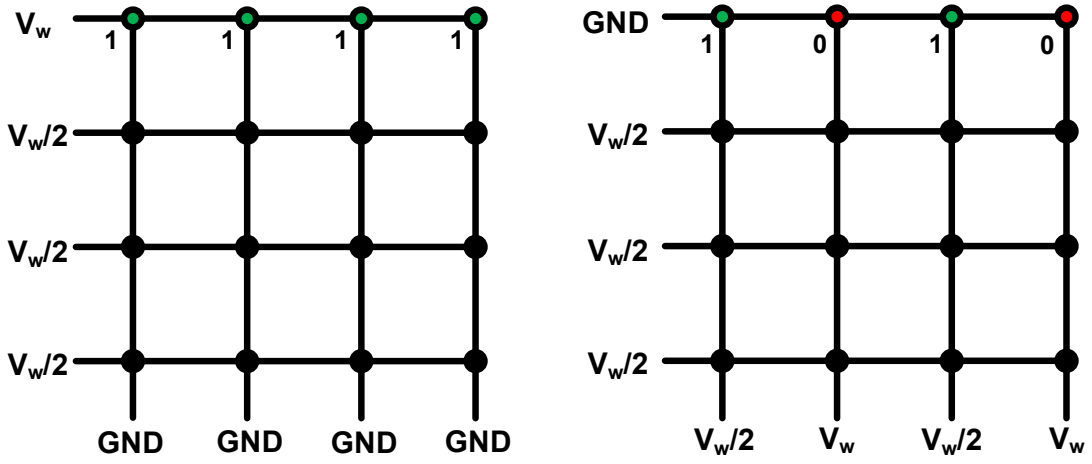
3.1.2 Write Operation in Multiple cells

Another area worthy of review is the multiple bits write in crossbar arrays. It is possible to write multiple cells in a crossbar row simultaneously by applying a write voltage to the selected rows and biasing the columns that have the target cells. This simple technique works well if only all the target cells are to be programmed to the same value. However, write operation becomes more complex when the target cells are to be programmed with different values. We assume the memristors are in the bipolar mode of operation (can only be switched ON (OFF) by a positive voltage and switched OFF (ON) only by a negative voltage). Writing non-similar data in multiple cells require the SET operation to write a logic 1 (ON) and the CLEAR (RESET) operation to write a logic 0 (OFF). Two write schemes for writing non-similar data in multiple cells in memristor-based array were proposed in [32]: SET-before-RESET (S-b-R) and ERASE-before-RESET (E-b-R) as shown in Fig. 3.6 and 3.7 respectively.



(a) SET Phase - Write of $1\,X\,1\,X$ **(b) RESET Phase - Write of** $X\,0\,X\,0$

Figure 3.6: Conventional SET-before-RESET technique using the $V/2$ **Write Scheme.**



(a) ERASE Phase - Write of $1\,1\,1\,1$ **(b) RESET Phase - Write of** $X\,0\,X\,0$

Figure 3.7: Conventional ERASE-before-RESET technique using the $V/2$ **Write Scheme.**

SET (write of 1) and RESET (write of 0) operations cannot be simultaneously executed in a single cycle on the same row in a crossbar array because of the opposing polarity of write voltages required to SET and RESET the cell. For instance, a 4×4 memory array that needs to have a data set of "0 1 0 1" in its first row will have to go through two phases for this to be achieved, all in a bid to separate the SET and RESET operations. In the first method, SET-before-RESET (Fig. 3.6); the first phase of this method writes "x 1 x 1" where 'x' is the original state of the cell (unchanged), followed by a write of "0 x 0 x" in the second phase to complete the "0 1 0 1" sequence. The second method; ERASE-before-RESET (Fig. 3.7) starts by writing "1 1 1 1" (ERASE) in the row so as to erase the original content of the cells, then a RESET of "0 x 0 x" is carried out in the second phase.

The S-b-R and E-b-R multiple cells write techniques proposed in [32] was designed using the $V/2$ write scheme described later in Section 5.2.1.2. The $V/2$ scheme has the advantage of consuming less power and also reducing the number of cells that can be perturbed. One major downside of the $V/2$ scheme during single cell write is the high probability of the unselected cells been disturbed. In Chapter 5 of this thesis, improved multiple cells write techniques is presented with the use of the more reliable $V/3$ write scheme described later in Section 5.2.1.3. The few cells $(m+n-2)$ exposed to voltage disturbance in $V/2$ have a 0.5 probability of being affected while the exposed $(mn-1)$ cells in the $V/3$ scheme have a 0.33 chance of being disturbed.

3.2 Gas Sensing with Memristor

The gas sensing properties of metal oxide semiconductors have been widely studied over the years [84, 85, 86, 87]. Some common metal oxide semiconductors sensor include tin dioxide (SnO_2), zinc oxide (ZnO), chromium titanate (CTO) among many others. With respect to other sensor technologies they are simple, inexpensive, miniaturizable and of good sensitivity [88]. Despite their simple operation, they are scarcely selective and might be required to work at high temperature [89]. However, advances have been made in some of these areas. Recently, authors in [90] demonstrated that nanomaterials metal oxide gas sensor can perform optimally at room temperature [91]. Gas sensors are particularly important in preventing the spread of gases that are harmful to organic life and also in the detection of oxygen deficiencies in environments where they are required. TiO_2 metal oxide semiconductor, which forms the crux of this thesis was used by HP Labs to fabricate the first physical memristor device [30]. HP's memristor was made of a thin film TiO_2 sandwiched between platinum electrodes. TiO_2 has specifically received extensive attention in various applications such as photovoltaics [92], photocatalysis [93] and sensors [94]. In [95], relative humidity sensors were tested by employing vertically aligned TiO_2 nanotubes array film produced using electro-chemical anodisation of titanium foil followed by a nitrogen-doping process. It was shown that the overall sensitivity is higher at lower frequency range of $109-1000Hz$.

The use of metal oxide nanoscale devices as gas sensors and biosensors has grown in recent years [95]. Memristor is among the nano-devices that have recently been investigated for their use as a sensing element because it primarily consists of metal oxide [53] (sec-

tion 6.2). Apart from TiO_2, memristive devices can be fabricated using different metal oxide semiconductors such as aluminium oxide (Al_2O_3) [96], copper oxide (Cu_2O) [97], silicon oxide (SiO) [98] among many others [56]. Memristor's resistance variation can be captured to indicate presence of gases and it is the building block of recently shown memristor sensors. Authors in [99] experimentally demonstrated that a $Pt/TiO_2/Pt$ memrsitor can effectively be used as an hydrogen sensor. When a certain concentration of gas is directed towards the surface of a memristor, its resistance might be altered depending on the semiconducting material (Fig. 6.3). This causes a change in output of the associated read circuitry. Modelling this resistance changing behaviour in a sensor array is the basis of a smart sensing system. By building different structural models of resistance changes and then modelling the read behaviour and sensitivity of the structure, it is possible to evaluate the performance of a smart sensing system.

Memristive-biosensors were reported in [53], where memristive effects were registered on silicon nanowire. They also fabricated nanowires using lithographic technique that allows precise and selective etching at the nanoscale. However, none of the existing literature directly addresses gas sensing using memristor crossbar array. This thesis therefore provides an initial framework for the use of memristor crossbar array for gas sensing using information and techniques developed from the crossbar read and write operation (chapter 6).

3.3 Baseline Research: Write Schemes for Multiple CRS cells

So far, this review has focussed on pure memristor based crossbar array. As part of the initial research presented in this thesis, we extend the multiple cells schemes to CRS-based crossbar array. As it is with single memristor based memory array described in Section 3.1.2, CRS-based memory also require a two-phase write operation when multiple cells are to be written in a row. Two write schemes for writing multiple cells in memristor-based array was proposed in [32]: SET-before-RESET (S-b-R) and ERASE-before-RESET (E-b-R) as discussed in Section 3.1.2.

Writing to a single CRS cell in a cross point structure can be achieved via the $V/2$ or $V/3$ scheme [100], both schemes however have their challenges. In the $V/2$ scheme, half-

selected cells usually have a voltage drop as high as $V/2$ which often leads to resistance loss if this voltage is applied over a long time [37]. $V/3$ prevents resistance loss but consumes more power as a result of the large $2V/3$ voltage applied to the unselected cells.

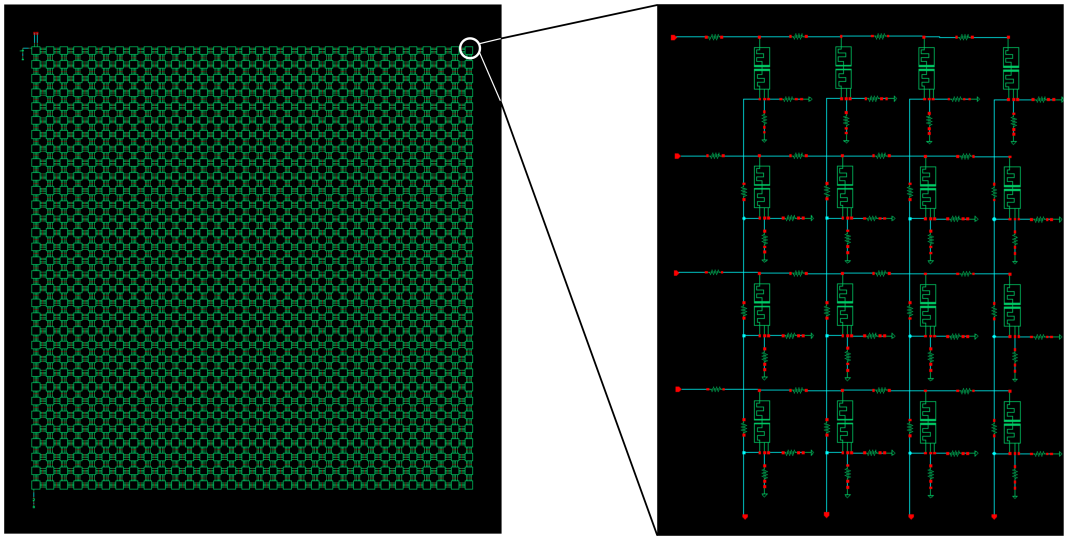


Figure 3.8: Schematic of a 4×4 CRS-based memory used to obtain the result in Table 3.2 and 3.3

3.3.1 E-b-R scheme with CRS-Based Memory Array

This section presents a direct implementation of E-b-R scheme on CRS-based memory array as shown in Fig. 3.9(a)- 3.9(c). This work differs from what was done in [32] where E-b-R was applied to single cell array. Our aim here is to write a "0 1 0 1" data pattern in the first row of a 4×4 CRS memory array with separate analysis for a logic 0 and logic 1 initialisation. The Verilog-A model proposed in [101] was used to simulate each CRS cell; this model combines the linear and non-linear memristor characteristics and has the advantage of ON/OFF symmetric voltages. Approximately twice the write voltage of memristor is required for the CRS because the CRS is a combination of two memristors.

Our design is similar to the implementation of this scheme on memristor-based array in [32] but with further consideration of the initial state of the cells. The effectiveness of the E-b-R scheme on CRS array is also affected by the initial state of the cells. Simulation results from this implementation are presented in Fig. 3.10. The E-b-R scheme executes successfully when cells are initialised to 1 but the scheme fails when the cells are initialised to 0 (3.9). When the cells that are meant to switch to 0 in the RESET phase are stuck in the ON state. The reason for this failure will be discussed and addressed in Section 3.3.2.

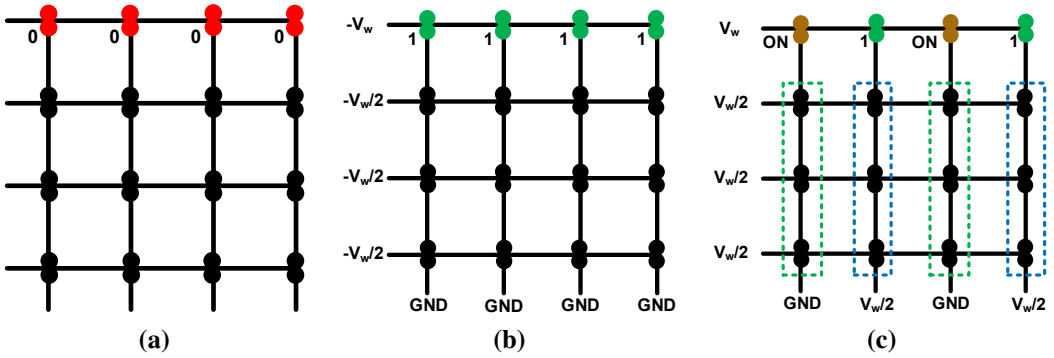
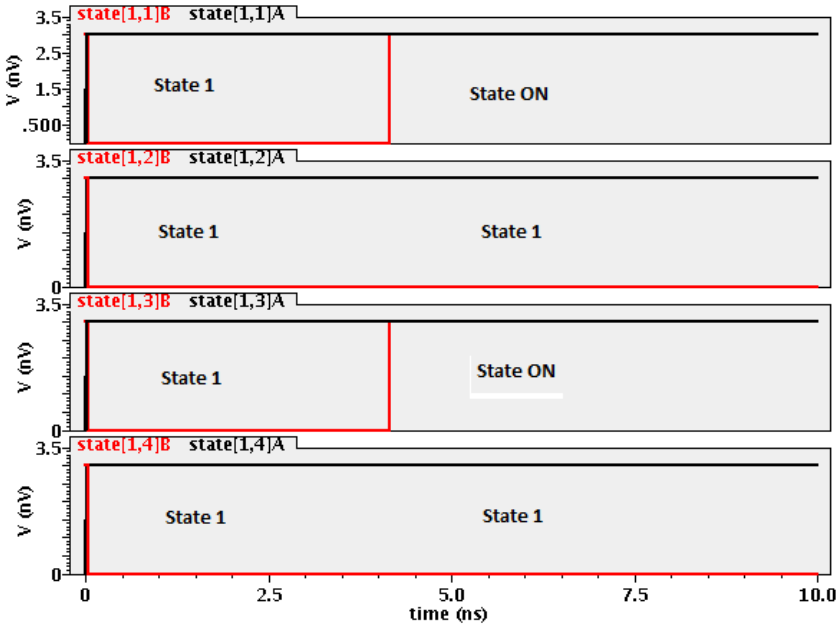
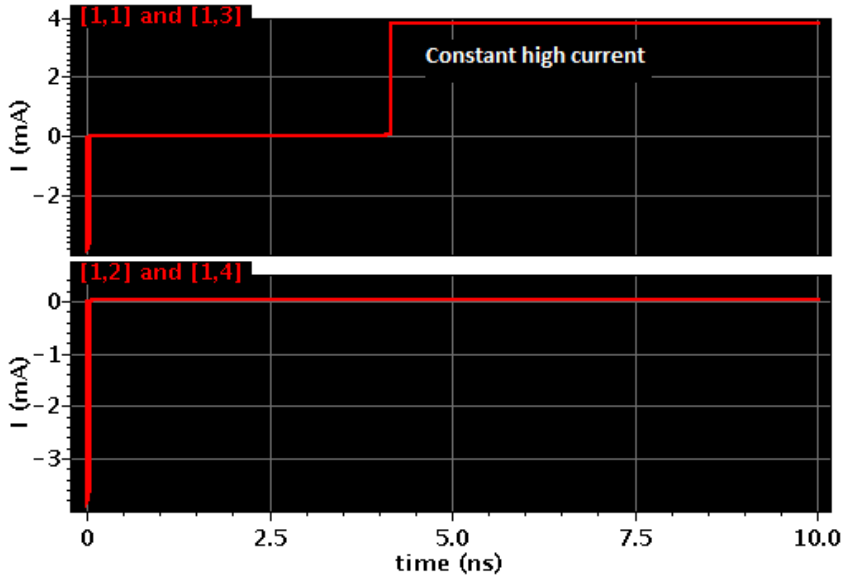


Figure 3.9: Direct implementation of E-b-R on CRS memory array. (a) Set-up Phase (b) ERASE Phase ("1 1 1 1") (c) RESET Phase ("0 x 0 x"), cells in blue dashed box represents the unselected cells while green represents the half-selected lines.

When all the cells in the first row are initialised to 0, cells (1, 2) and (1, 4) had a complete transition but cells (1, 1) and (1, 3) which have to go through two transitions ($0 \rightarrow 1$ and $1 \rightarrow 0$) get stuck in the ON state (LRS/LRS) during the second transition ($1 \rightarrow 0$). This is evident by the persistent high current in the array right from the start of the second phase (4ns) till the end of the simulation as shown in Fig. 3.10(a) and 3.10(b). This means that the applied voltage is not sufficient for the cells to complete switching in the second phase. The effect of these cells getting stuck at the ON state is increased power consumption in the array as a result of high current that is being generated at this transient state. The integrity of the array is also affected negatively as subsequent read operation will result in a false output.



(a)



(b)

Figure 3.10: Simulation outcome of E-b-R scheme on CRS-based memory array with cells initialised to 0. (a) Cells (1, 1) and (1, 3) that have to go through more than one transition are stuck in the ON state. (b) Current waveform. Spike at 0ns indicate state change across all four cells $(0 \rightarrow 1)$ in 1st phase but cells (1, 1) and (1, 3) generated a persistent high current all through the 2nd phase as a result of the cells been stuck in transient (ON) state in the process of executing the $(1 \rightarrow 0)$ transition.

3.3.2 CRS Compensated Write Voltage Technique

As earlier mentioned, more write energy is needed for the E-b-R scheme. This problem becomes more prominent with CRS arrays because of the doubled voltage required to write the CRS cells as well as the voltage degradation suffered by the CRS array after the first phase. The proposed method to correct the challenge faced by the E-b-R scheme

when the cells are initialised to 0 was achieved through the biasing voltages for both rows and columns of the array. The following manipulations were done to achieve an effective E-b-R scheme on CRS memory array irrespective of the initial state of the cells:

- As against the usual grounding of the column terminal of the cell(s) to be written to, an additional voltage source of $V_{write}/2$ was connected to the column of cells (1, 1) and (1, 3) in the second phase (Fig. 3.9c). This is to compensate for any amount of voltage that might have been lost. This will also ensure that there is sufficient voltage for the upper memristor in the CRS to switch from LRS to HRS, thereby the CRS switches to the required logic state 0.

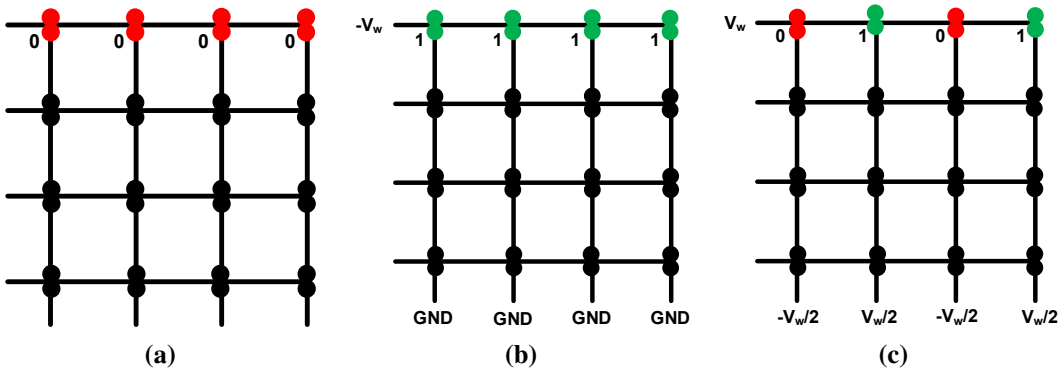


Figure 3.11: Proposed E-b-R scheme on CRS memory array showing complete transition. (a) Initial Phase (b) ERASE Phase ("1 1 1 1") (c) RESET Phase ("0 x 0 x").

- The redundant $V_{write}/2$ biasing that was applied to the terminals of other unselected rows in the first phase of the old scheme was also ignored in this new scheme, the unselected rows are left floating for both phases. This leaves the voltages of the unselected and half-selected cells at $-V_{write}/2$ and $V_{write}/2$ respectively in the second phase, this voltage would not be sufficient to switch the unselected and half-selected cells but will ensure we get the desired output for the selected cells.

- The power conservation ability of the improved scheme was achieved primarily in the second phase. For the first phase the unselected and half-selected cells have a net voltage of $0V$ across them while the selected cells have a voltage drop of V_{write} across them as against the old scheme that biased these unselected rows with a minimum voltage of $|V_{write}/2|$ in both phases.

Both cases of initialising cells to 0 and 1 are verified to be successful in their execution as shown by the simulation output in Fig. 3.12a and 3.12b. Tables 3.2 and 3.3 compare

the performance metrics. These results are generated using subset of the schematic in Fig. 3.8.

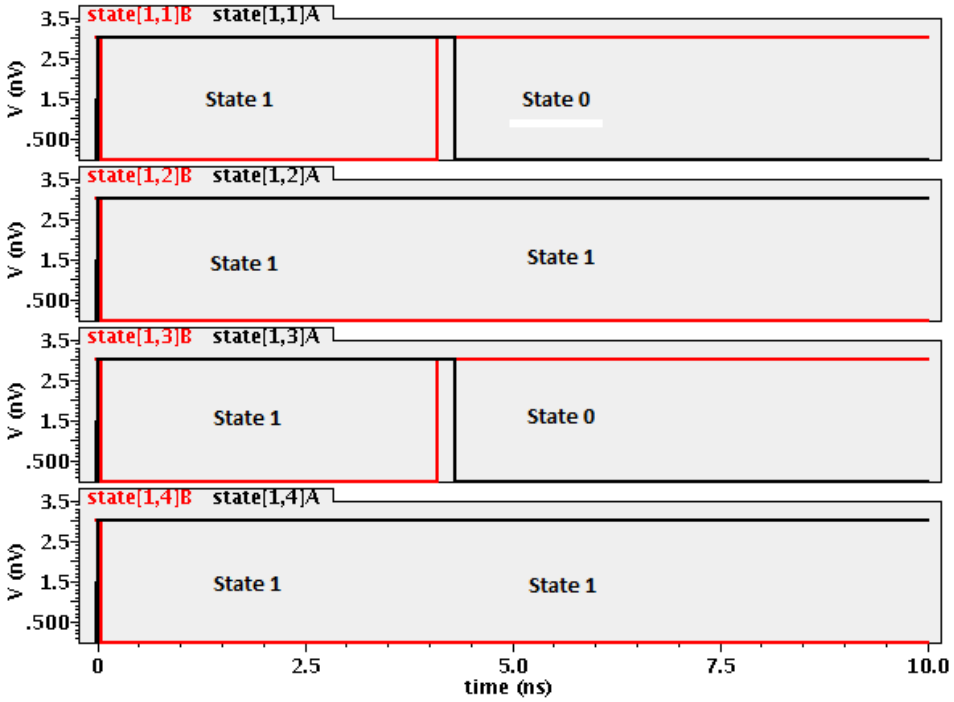
Table 3.2: Result of E-b-R write scheme with cells initialised to logic 0. New method compared against conventional.

Cells	**Proposed E-b-R write scheme**		**Conventional E-b-R write scheme**	
	1st Phase	**2nd Phase**	**1st Phase**	**2nd Phase**
$(1,1)$	$0 \rightarrow 1$	$1 \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow ON$
$(1,2)$	$0 \rightarrow 1$	$1 \rightarrow 1$	$0 \rightarrow 1$	$1 \rightarrow 1$
$(1,3)$	$0 \rightarrow 1$	$1 \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow ON$
$(1,4)$	$0 \rightarrow 1$	$1 \rightarrow 1$	$0 \rightarrow 1$	$1 \rightarrow 1$
Total power	**1.39mW**		**17.24mW**	
Write time	0.26ns		Incomplete transition	

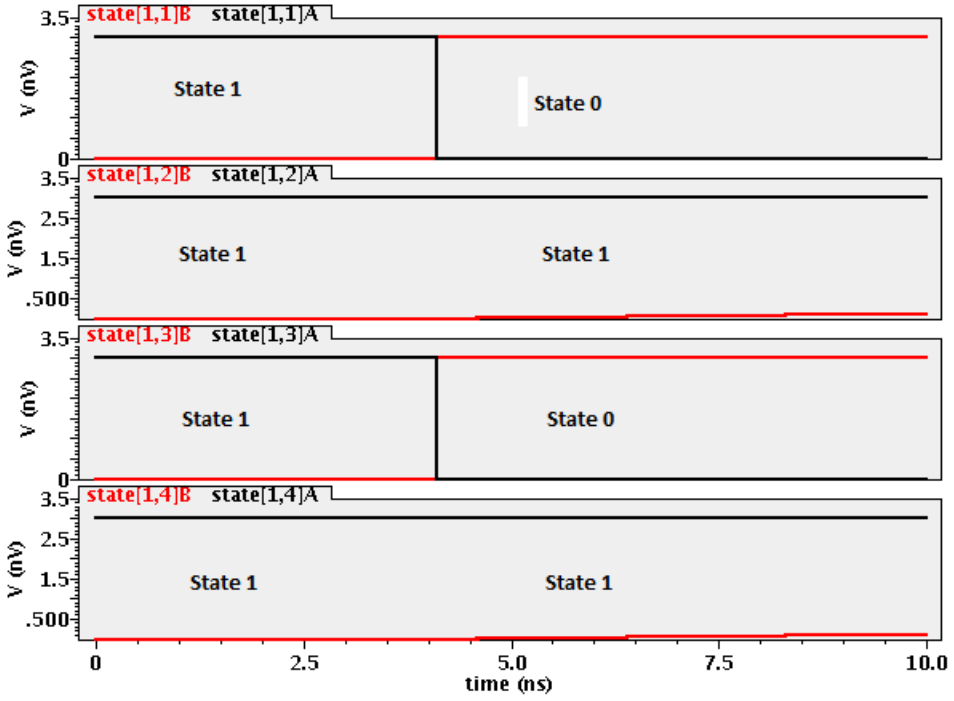
Table 3.3: Result of E-b-R write scheme with cells initialised to logic 1. New method compared against conventional.

Cells	**Proposed E-b-R write scheme**		**Conventional E-b-R write scheme**	
	1st Phase	**2nd Phase**	**1st Phase**	**2nd Phase**
$(1,1)$	$1 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 1$	$1 \rightarrow 0$
$(1,2)$	$1 \rightarrow 1$	$1 \rightarrow 1$	$1 \rightarrow 1$	$1 \rightarrow 1$
$(1,3)$	$1 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 1$	$1 \rightarrow 0$
$(1,4)$	$1 \rightarrow 1$	$1 \rightarrow 1$	$1 \rightarrow 1$	$1 \rightarrow 1$
Total power	0.29mW		0.34mW	
Write time	0.05ns		0.12ns	

In terms of power consumptions, our approach ensures steady and minimal power consumption (Fig. 3.13) during both phases of the operation as against the old approach. The old approach resulted in a sharp increase in power consumption at the start of the second phase especially when the cells are initialled to 0s (this is the worst case scenario when cells have to undergo more than one transition). This sharp increase is as a result of some of the cell's inability to complete their second transition and getting stuck in the transient state as earlier mentioned. The magnitude and duration of the write voltage as well as the current dissipated during the write operation are factors that contributes to the total power consumed. Also worthy of mention is the mobility rate of the switching material, a high mobility rate increases the switching speed and reduces power consumption in most cases.

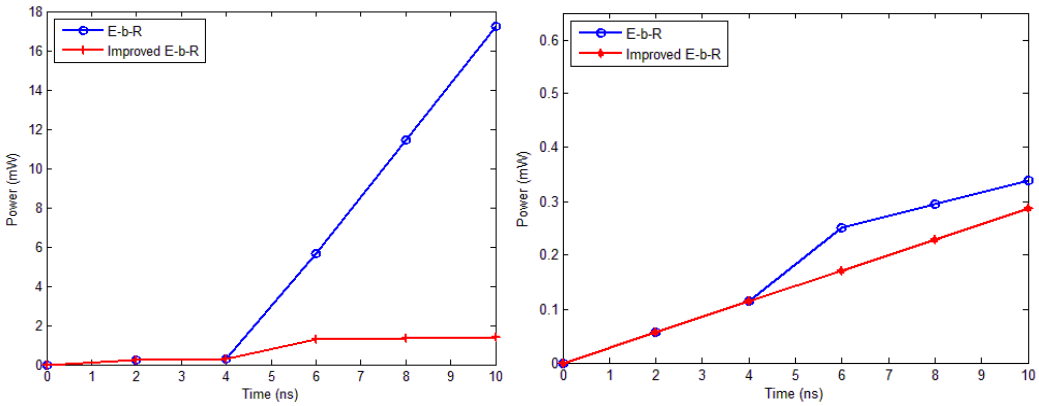


(a)



(b)

Figure 3.12: Simulation of improved E-b-R scheme with CRS. (a) With array initialised to 0s (b) With array initialised to 1s. All the four cells were successfully written with the desired "0 1 0 1" data pattern. State A and state B are the states of the two memristors in each CRS.



(a) (b)

Figure 3.13: Comparison of power consumption in memory array (a) With array initialised to 0s (b) With array initialised to 1s.

3.4 Summary

This chapter identified and briefly reviewed existing work related to read and write operations in crossbar arrays. Also reviewed is the application of memristors as a gas sensor. The chapter concludes with a presentation of our initial research result in the use of multiple cells read technique to improve the reliability of read operation in CRS crossbar array. All of the reviewed work have their pros and cons. The work done in this thesis is aimed towards improving the reliability of these major operations of the crossbar in order to widen the application area of the memristor. It is worth mentioning that a complete memory system will also require reliable control circuitry which have been referred to in the previous chapter.

Chapter 4

Improved Techniques for Crossbar Array Read Operation

4.1 Introduction

Reliable and high performance memory read techniques is one of the key factors in emerging memory technologies and it is becoming increasingly important as integration complexity continues to grow. Ideally, memristor-based memories are not susceptible to radiation induced soft errors but research has shown error might occur from the failure of their read and write operations [102]. An in-depth understanding of the memristor-based crossbar array and its read schemes are critical to the adoption of this technology in memory design and other related applications as such as gas sensing. One major challenge in successfully reading from memristor-based memories is the sneak-paths problem described in Chapter 2. In traditional memories, selection or isolation devices such as transistors and diodes are used to prevent the parasitic current paths in the system. As explained previously, extra component increases the footprint of the device and makes it difficult to design reliable high density memory structures. This chapter begins with the derivation of an improved model for the crossbar array in Section 4.2. This new model captures the practical behaviour of the crossbar array. Section 4.3 presents the resistance models of the existing crossbar array read schemes before the effects of sneak-paths on the read margin of the read schemes were comprehensively analysed in Section 4.4. The power consumption rate of these schemes are presented in Section 4.5. Section 4.6 proposes a novel multiple cells read scheme capable of eliminating sneak-paths depending on the data distribution in the array. The multiple cells read technique was also applied to a novel

error detection and correction technique for memristor-based memories. Most of the designs and simulations in this chapter are carried out on Cadence Spectre simulation tool using smaller arrays. In order to design and simulate much larger arrays, simulation tools were developed in C++ using existing and derived mathematical models of the designs. The results of the C++ tool were verified against that of the Cadence tool. The C++ scripts are listed in appendix B and are referred to in different sections and figure captions in this chapter. A typical 256×256 crossbar array can be simulated in less than a minute by the developed C++ script by simply updating the parameterised m and n values. This same array size requires extra design work with Cadence Virtuoso user interface with simulation time in hours.

4.2 Improved Crossbar Array Model

It is crucial to have an accurate and robust model that captures the behaviour of crossbar read operation that will enable designers to verify simulation and other related performances of the read operation. This section presents improved formulas that model the $m \times n$ crossbar array depicted in Fig. 4.1. Assuming each of the memristors have a resistance value $R_{i,j}$ ($i = 1...m$ and $j = 1...n$) that can either be R_{on} or R_{off} and $R_{1,1}$ is selected for read as depicted and explained in Fig. 4.1 and Section 2.3. The closed form formulas for R_m, R_n and R_{mn} in Section 2.3 [68] were derived with the assumption that all memristors in a group have an identical resistance state of either R_{on} or R_{off}. This is usually not the case, each group will contain both low and high resistance state cells in practical applications. Robust closed form formulas can be derived for each group of memristors which caters for randomness of resistance state within the group such that each group is free to work with both high and low resistance state cells simultaneously. In order to make the formula generic, the number of cells in the R_{off} and R_{on} state will be represented by K_{off} and K_{on} respectively.

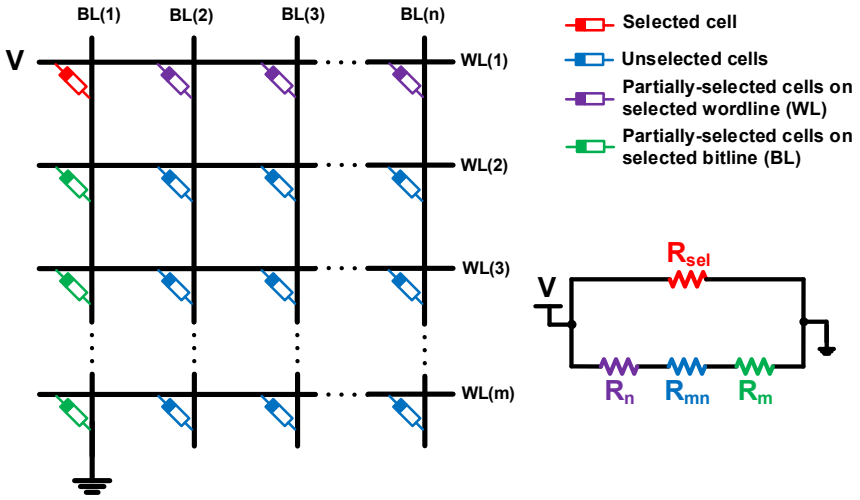


Figure 4.1: An $m \times n$ memristor-based crossbar memory structure and its equivalent circuit model that depict the effect of all resistances in the array. R_{sel} represents the selected cell $R_{1,1}$. R_m and R_n represent the row and column partially-selected cells respectively and R_{mn} simplify the cells in the unselected lines [68]. Duplicated from Fig. 2.14.

To derive the formula R_m for the resistance of the $m-1$ Group II (green) cells connected in parallel:

$$\frac{1}{R_m} = \frac{1}{R_{2,j_s}} + \frac{1}{R_{3,j_s}} + \ldots + \frac{1}{R_{m,j_s}} \tag{4.1}$$

since $R_{i,j} \in \{R_{on}, R_{off}\}$ and $j_s = 1$ is the index for selected column

$$\frac{1}{R_m} = \frac{1}{R_{on} \text{ or } R_{off}} + \frac{1}{R_{on} \text{ or } R_{off}} + \ldots + \frac{1}{R_{on} \text{ or } R_{off}} \tag{4.2}$$

$$R_m = \frac{R_{on}R_{off}}{K_{on}R_{off} + K_{off}R_{on}} \tag{4.3}$$

since $K_{on} + K_{off} = m-1$, $K_{on} = m-1-K_{off}$ and $K_{off} = m-1-K_{on}$

$$R_m = \frac{R_{on}R_{off}}{(m-1-K_{off})R_{off} + (m-1-K_{on})R_{on}} \tag{4.4}$$

$$= \frac{R_{on}R_{off}}{(m-1-K_{off})R_{off} + (K_{off}+K_{on}-K_{on})R_{on}} \tag{4.5}$$

$$= \frac{R_{on}R_{off}}{(m-1-K_{off})R_{off} + K_{off}R_{on}} \tag{4.6}$$

$$= \frac{R_{on}R_{off}}{(m-1)R_{off} - K_{off}R_{off} + K_{off}R_{on}} \tag{4.7}$$

Therefore,

$$R_m = \frac{R_{on}R_{off}}{(m-1)R_{off} - K_{off}(R_{off} - R_{on})} \tag{4.8}$$

For the resistance of the $n-1$ Group III (purple) cells connected in parallel, R_n;

$$\frac{1}{R_n} = \frac{1}{R_{i_s,2}} + \frac{1}{R_{i_s,3}} + ... + \frac{1}{R_{i_s,n}} \tag{4.9}$$

Similar derivation process used for R_m can be used to derive R_n except that, $K_{on} + K_{off} = n-1$. Therefore:

$$R_n = \frac{R_{on}R_{off}}{(n-1)R_{off} - K_{off}(R_{off} - R_{on})} \tag{4.10}$$

For the resistance R_{mn} of the $(m-1)(n-1)$ Group IV (blue) cells connected in parallel:

$$\begin{aligned}\frac{1}{R_{mn}} =& \frac{1}{R_{2,2}} + \frac{1}{R_{2,3}} + ... + \frac{1}{R_{2,n}} \\ &+ \frac{1}{R_{3,2}} + \frac{1}{R_{3,3}} + ... + \frac{1}{R_{3,n}} \\ &\vdots \\ &+ \frac{1}{R_{m,2}} + \frac{1}{R_{m,3}} + ... + \frac{1}{R_{m,n}}\end{aligned}$$

Similar derivation process as in R_m applies except that, $K_{on} + K_{off} = (n-1)(m-1)$. Therefore:

$$R_{mn} = \frac{R_{on}R_{off}}{(m-1)(n-1)R_{off} - K_{off}(R_{off} - R_{on})} \tag{4.11}$$

The existing set of Eqn. 2.14 - 2.16 and the newly derived Eqn. 4.8 - 4.11 for modelling the resistances in a crossbar array can be combined to form a new complete set as in Eqn. 4.12 - 4.14.

$$\begin{aligned}R_m &= \frac{\prod_{i=1, i\neq i_s}^{m} R_{i,j_s}}{\sum_{i=1, i\neq i_s}^{m} R_{i,j_s}} \\ &= \begin{cases} \dfrac{R}{m-1} & \text{if all } R_{i,j_s} \text{ are equal} \\ \dfrac{R_{on}R_{off}}{(m-1)R_{off} - K_{off}(R_{off} - R_{on})} & \text{otherwise} \end{cases}\end{aligned} \tag{4.12}$$

$$
\begin{aligned}
R_n &= \frac{\prod_{j=1, j\neq j_s}^{n} R_{i_s,j}}{\sum_{j=1, j\neq j_s}^{n} R_{i_s,j}} \\
&= \begin{cases} \dfrac{R}{n-1} & \text{if all} R_{i_s,j} \text{ are equal} \\ \dfrac{R_{on}R_{off}}{(n-1)R_{off} - K_{off}(R_{off} - R_{on})} & \text{otherwise} \end{cases}
\end{aligned} \tag{4.13}
$$

$$
\begin{aligned}
R_{mn} &= \frac{\prod_{j=1, j\neq j_s}^{n} \prod_{i=1, i\neq i_s}^{m} R_{i,j}}{\sum_{j=1, j\neq j_s}^{n} \sum_{i=1, i\neq i_s}^{m} R_{i,j}} \\
&= \begin{cases} \dfrac{R}{(m-1)(n-1)} & \text{if all } R_{i,j} \text{ are equal} \\ \dfrac{R_{on}R_{off}}{(m-1)(n-1)R_{off} - K_{off}(R_{off} - R_{on})} & \text{otherwise} \end{cases}
\end{aligned} \tag{4.14}
$$

Here, i and j are the word-line and bitline index respectively while i_s and j_s are the selected wordline and bitline respectively.

The overall worst case sneak-path in the array occurs when the resistance of the unselected cells are set to R_{on}. If we consider the floating line scheme discussed in section 4.3.1 - $R_{sneak} = R_n + R_{mn} + R_m$. R_n, R_{mn}, R_m are defined in Eqn. 4.12 - 4.14. These parameters are at the worst case when $K_{on} = m-1$ and $K_{off} = 0$

For worst case sneak-path to occur, Eqn 4.15 must hold:

$$
R_{sneak\_min} = R_{on}\left(\frac{m+n-1}{(m-1)(n-1)}\right) \tag{4.15}
$$

If we assume the resistance of the selected cell to be R_{mem_on} and R_{mem_off} when on and off respectively. In the presence of sneak-path, read failure occurs when $R_{mem\_off} || R_{sneak\_min} \leq$

R_{mem_on}. Let's find the maximum value of m and n for this condition to occur:

$$\frac{1}{R_{mem\_off}} + \frac{1}{R_{sneak\_min}} \geq \frac{1}{R_{mem\_on}} \tag{4.16}$$

$$\frac{R_{mem\_on}.R_{mem\_on}}{R_{mem\_off} - R_{mem\_on}} \leq R_{sneak\_min} \tag{4.17}$$

$$\frac{R_{mem\_on}.R_{mem\_off}}{R_{mem\_off} - R_{mem\_on}} \leq R_{mem\_on}\left(\frac{m+n-1}{(m-1)(n-1)}\right) \tag{4.18}$$

$$\frac{R_{mem\_off} - R_{mem\_on}}{R_{mem\_off}} \geq \frac{(m-1)(n-1)}{m+n-1} \tag{4.19}$$

Let the LHS of Eqn. 4.19 be represented as ρ and considering a square array where $n = m$:

$$\rho = \frac{(n-1)^2}{2n-1} \tag{4.20}$$

$$n^2 - 2(\rho+1)n + (\rho+1) = 0 \tag{4.21}$$

If we consider only the positive case:

$$n = \rho + 1 + \sqrt{(\rho+1)\rho} \tag{4.22}$$

In the case when $R_{mem\_off} \gg R_{mem\_on}$, $\rho \leq 1$. If we take the ultimate case of $\rho = 1$:

$$n_{max} = 2 + \sqrt{2} \tag{4.23}$$

Therefore the maximum array size with the floating read scheme is approximately 3×3

4.3 Modelling Crossbar Array Read Schemes

In a bid to achieve high density resistive memory structures, reliability of the read operation has depended on the treatment of the unselected lines rather than addition of selection devices. This section explores the possible voltage configurations of the unselected lines and the derivation of analytic models that can be used to evaluate each scheme. These models are used in conjunction with the equations of the group of crossbar resistances discussed in Section 4.2.

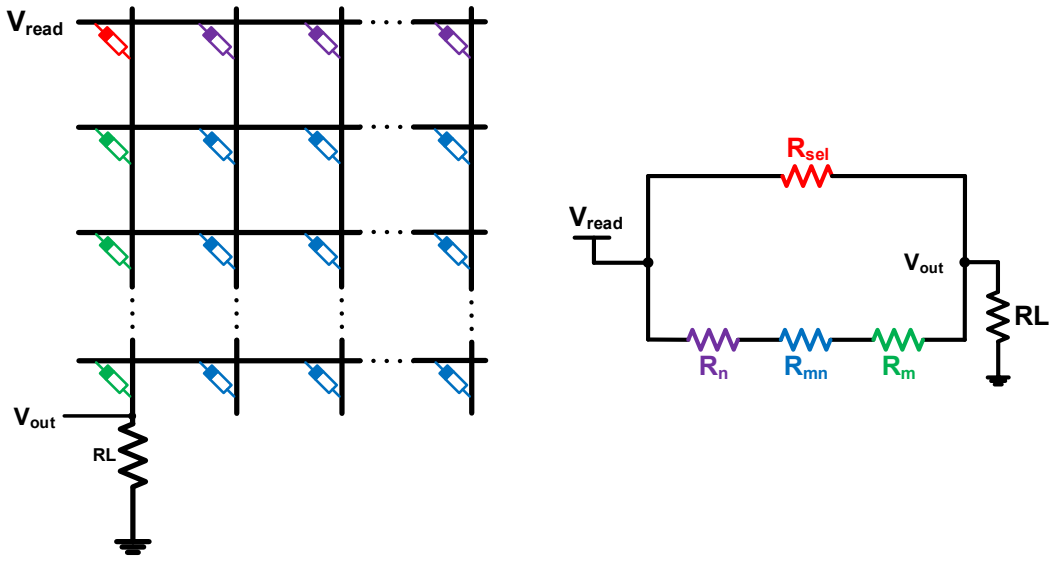
4.3.1 Floating Wordlines and Floating Bitlines

With the Floating Wordlines and Floating Bitlines (FWFB) read scheme, all the unselected lines are kept floating as shown in Fig. 4.2(a). Each memristor in the array is modelled as a resistor and the entire crossbar array will be grouped as discussed in Section 2.3 in order to simplify the array to the equivalent resistance model depicted by Fig. 4.2(b). The effect of other unselected cells on the target memristor (R_{sel}) can be computed according to Eqn. 4.24, where the effective read-out of the target selected cell can be dominated by all three groups of neighbouring cells ($mn-1$). The load resistor is unaffected in this scheme. For the floating wordlines and bitlines scheme, all the neighbouring groups of cells R_m, R_n and R_{mn} affect R_{sel} and R_L is unaffected.

$$R_{sel\_eff} = R_{sel} \,||\, (R_m + R_{mn} + R_n) \tag{4.24a}$$

$$R_{L\_eff} = R_L \tag{4.24b}$$

$$R_{total} = (R_{sel} \,||\, (R_m + R_{mn} + R_n)) + R_L \tag{4.24c}$$



(a) (b)

Figure 4.2: Structure of the FWFB read scheme and its corresponding equivalent circuit model that depicts the effect of all resistances in the array during the read operation. The output voltage is the output of the voltage divider between R_{sel} and R_L.

4.3.2 Floating Wordlines and Grounded Bitlines

The Floating Wordlines and Grounded Bitlines (FWGB) scheme minimises the impact of leakage current by grounding all the unselected bitlines as shown in Fig. 4.3(a), this ensures that some of the sneak-path currents are directed to ground instead of the read-out line. From the resistance model of this scheme shown in Fig. 4.3(b), the effect of R_n is directed to the ground. This is also reflected in the effective resistance of the selected memristor and the effective load resistance computed by Eqn. (4.25a) and (4.25b) respectively. The selected memristor is not affected by any of the neighbouring cells but undesired currents from R_m and R_{mn} are still able to reach the sensing circuit via the load resistor. For the floating wordlines and grounded bitlines scheme, R_m and R_{mn} are the only groups of cells that affect R_L. R_{sel} is unaffected. Undesired currents through R_n flow directly to the ground and does not interfere with V_{out}.

$$R_{sel\_eff} = R_{sel} \tag{4.25a}$$

$$R_{L\_eff} = R_L \,||\, (R_m + R_{mn}) \tag{4.25b}$$

$$R_{total} = (R_{sel} + (R_L \,||\, (R_m + R_{mn}))) \,||\, R_n \tag{4.25c}$$

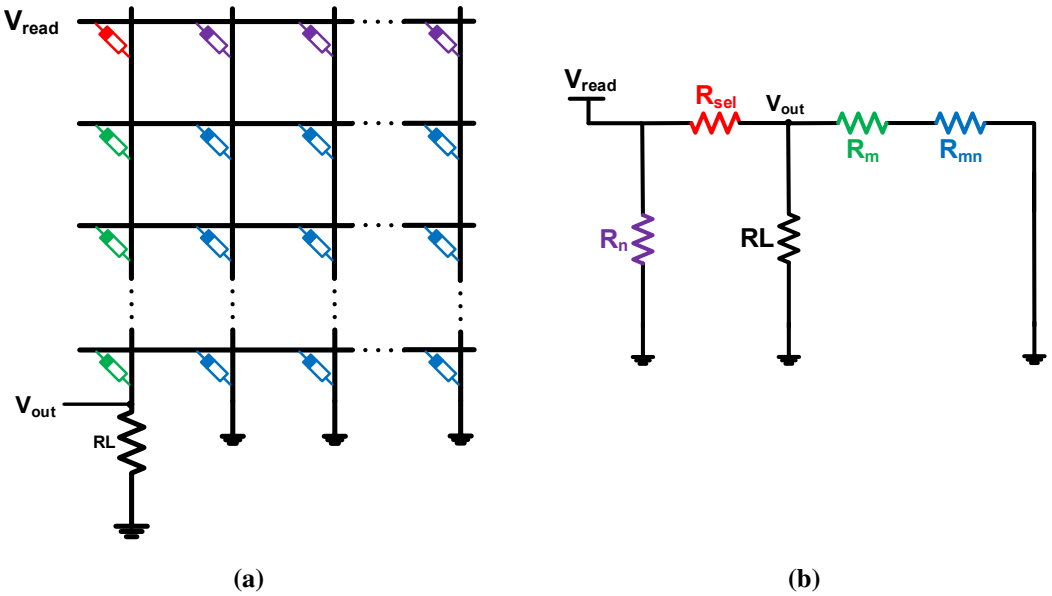


Figure 4.3: Structure of the FWGB read scheme and its corresponding equivalent circuit model that depicts the effect of all resistances in the array during the read operation. The output voltage is the output of the voltage divider between R_{sel} and R_L.

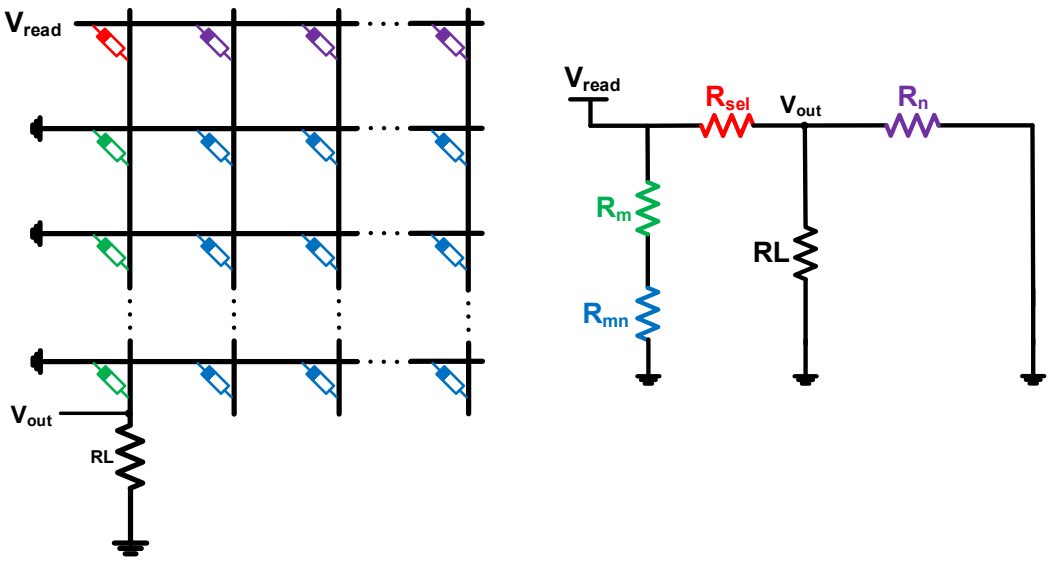
4.3.3 Grounded Wordlines and Floating Bitlines

The Grounded Wordlines and Floating Bitlines (GWFB) scheme minimises the impact of leakage current by grounding all the unselected wordlines as shown in Fig. 4.4(a), this ensures majority of the sneak-path currents are directed to ground instead of the read-out line. From the resistance model of this scheme shown in Fig. 4.4(b), the effect of R_m and R_{mn} are directed to the ground. This is also reflected in the effective resistance of the selected memristor and the effective load resistance computed by Eqn. (4.26a) and (4.26b) respectively. The selected memristor is not affected by any of the neighbouring cells but undesired currents from $n-1$ cells of the R_n group are still able to reach the sensing circuit through the load resistor. For the grounded wordlines and floating bitlines scheme, R_n is the only group of cells that affect R_L. R_{sel} is unaffected. Undesired currents through R_m and R_{mn} flow directly to the ground and does not interfere with V_{out}.

$$R_{sel\_eff} = R_{sel} \tag{4.26a}$$

$$R_{L\_eff} = R_L \,||\, R_n \tag{4.26b}$$

$$R_{total} = (R_{sel} + (R_L \,||\, R_n)) \,||\, (R_m + R_{mn}) \tag{4.26c}$$



(a) (b)

Figure 4.4: Structure of the GWFB read scheme and its corresponding equivalent circuit model that depicts the effect of all resistances in the array during the read operation. The output voltage is the output of the voltage divider between R_{sel} and R_L.

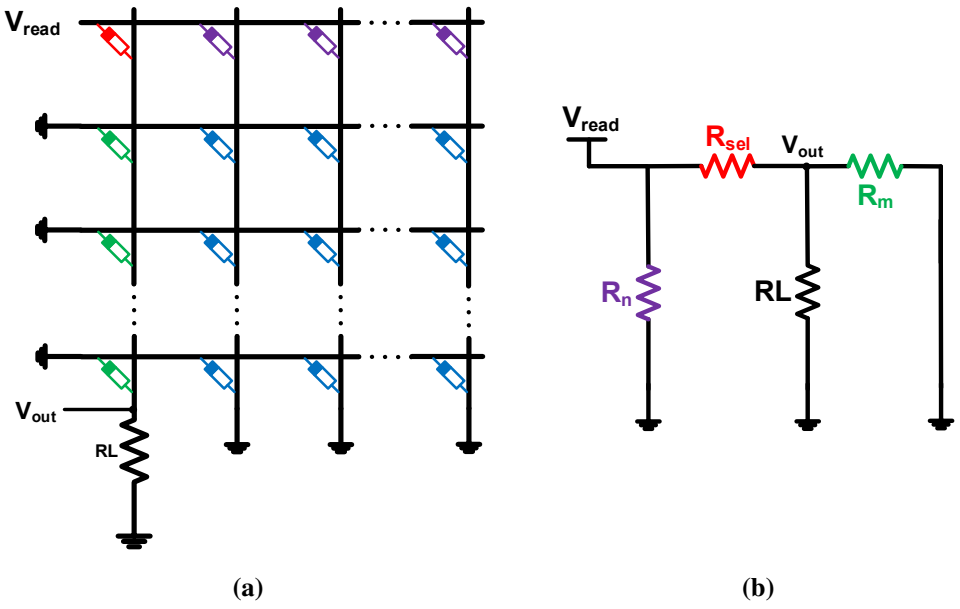
4.3.4 Grounded Wordlines and Grounded Bitlines

Similar to the GWFB scheme, the Grounded Wordlines and Grounded Bitlines (GWGB) minimises the impact of leakage currents by grounding all the unselected bitlines in addition to the grounded wordlines as shown in Fig. 4.5(a), this ensures majority of the sneak-path currents are directed to ground instead of the read-out line. From the resistance model of this scheme shown in Fig. 4.5(b), the current flowing through of R_m and R_{mn} are directed to the ground. This is also reflected in the effective resistance of the selected memristor and the effective load resistance computed by Eqn. (4.27a) and (4.27b) respectively. The selected memristor is not affected by any of the neighbouring cells but undesired currents from $m-1$ cells of the R_m group interfere with the output through the load resistor. For the grounded wordlines and grounded bitlines scheme, R_m is the only group of cells that affect R_L, R_{sel} is unaffected. Undesired currents through R_n and R_{mn} flow directly to the ground and does not interfere with V_{out}.

$$R_{sel\_eff} = R_{sel} \tag{4.27a}$$

$$R_{L\_eff} = R_L \,||\, R_m \tag{4.27b}$$

$$R_{total} = (R_{sel} + (R_L \,||\, R_m)) \,||\, R_n \tag{4.27c}$$



(a) (b)

Figure 4.5: Structure of the GWGB read scheme and its corresponding equivalent circuit model that depicts the effect of all resistances in the array during the read operation. The output voltage is the output of the voltage divider between R_{sel} and R_L.

4.4 Effects of Sneak-path on Crossbar Array Read Schemes

This section further analyse how sneak-paths affect the read margin of the previously described read schemes of Section 4.3. Read margin (ΔV_{out}) can be defined as the difference between the read output of a cell when it stores a logic 1 (V_{out_1}) and a logic 0 (V_{out_0}) as described by Eqn. 4.28.

$$\Delta V_{out} = V_{out\_1} - V_{out\_0} \tag{4.28}$$

$$V_{out} = V_{read} \times \frac{R_{L\_eff}}{R_{L\_eff} + R_{sel\_eff}} \tag{4.29}$$

V_{out_1} (when $R_{sel} = R_{on}$) and V_{out_0} ((when $R_{sel} = R_{off}$)) can be computed using Eqn. 4.29. Since the read margin is dependent on the read-out voltages (V_{out_1} and V_{out_0}), it is noteworthy to state that read margin of a crossbar is affected by the size of the array and the state of the unselected cells. The read-out voltage of a target cell will be greatly affected if the majority of its neighbours are in low resistance state because of the abundance of leakage currents that easily flow through low resistant cells. The best case read-out voltages are achieved in the presence of neighbours with high resistance value. Listed below are the four extreme cases (best and worst) of a read-out of '1' and '0' from a memristor cell in a crossbar with size greater than 1. The usual two cases of reading a logic 1 and 0 from a single memristor (crossbar size of 1) are also listed. For the rest of this chapter, the following defined notations will be used:

- $\mathbf{V_{out\_wc1}}$: Worst case read-out of logic 1 occurs when the selected cell is in LRS (1) and all other cells in the crossbar are in LRS (1)
- $\mathbf{V_{out\_bc1}}$: Best case read-out of logic 1 occurs when the selected cell is in LRS (1) and all other cells in the crossbar are in HRS (0)
- $\mathbf{V_{out\_wc0}}$: Worst case read-out of logic 0 occurs when the selected cell is in HRS (0) and all other cells in the crossbar are in LRS (1)
- $\mathbf{V_{out\_bc0}}$: Best case read-out of logic 0 occurs when the selected cell is in HRS (0) and all other cells in the crossbar are in HRS (0)
- $\mathbf{V_{out\_nc1}}$: normal and real case of reading a logic 1 from a single memristor (crossbar size of 1)

- $\mathbf{V_{out\_nc0}}$: normal and real case of reading a logic 0 from a single memristor (crossbar size of 1)

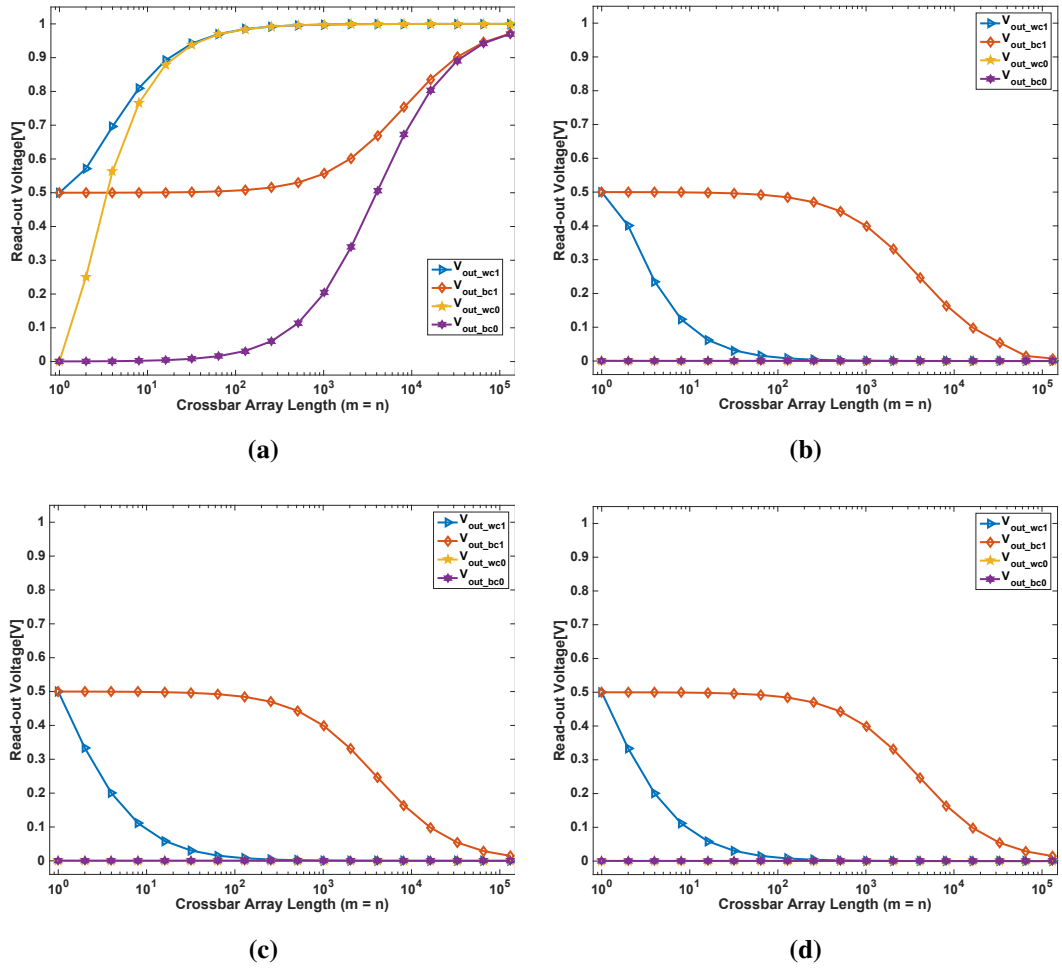


Figure 4.6: Simulation results of worst and best case read-out voltages for (a) FWGB (b) FWGB (c) GWFB (d) GWGB. Read voltage $V_{read} = 1V$, $R_{off} = 200K\Omega$, $R_{on} = 100\Omega$ and $R_L = 100\Omega$. A voltage divider sensing circuit was used and read-out voltages were calculated using Eqn. 4.29. Data for plots was generated using the read-out voltage values from the analytic tools listed in Appendix B.1, B.2, B.3 and B.4.

It is thus easy to conclude that highly resistive neighbours helps to prevent sneak-path in crossbar array irrespective of the content of the target cell. High resistance impedes the flow of current in any network, this explains why the the best case read of 1 and 0 occurs when the resistance of other cells are high. The worst case read-out occurs when the resistance of the neighbouring cells are low. Fig. 4.6(a) - 4.6(d) shows the best and worst read-out voltages for a target cell over a wide range of crossbar array (1×1 - $131K \times 131K$) for the four read schemes. Using any of the read schemes, an ideal case of reading a single memristor results in the following read-out voltages when the read voltage is 1V: $V_{out\_wc1} = V_{out\_bc1} = V_{out\_nc1} = 0.5V$ and $V_{out\_wc0} = V_{out\_bc0} = V_{out\_nc0} = 500\mu V$. Read-out voltages from a single memristor are used as the benchmark for the read-out voltages

from each memristor in arrays with more than one device. Examining the FWFB scheme, the negative effect of the neighbouring cells begin to kick in as the array size increases. At an array size of approximately 100×100, the worst cases are already outside of the range of the valid region ($500\mu V$ - $0.5V$) but the best cases are still decent as shown in Fig. 4.6(a). However, at a quite large array size of $131K \times 131K$, none of the read-outs are valid which effectively means the FWFB is not useful even with smaller array size in its worst case. The best cases for the FWFB degrades at large array sizes as the large values of m and n force the high resistance to a small value as a result of the parallel combination of the memristors. Similar examination of the three grounded schemes (FWGB, GBFB and GWGB) shows that all the read-out voltage cases remain within an acceptable range until an array size of approx. $1K \times 1K$ except for V_{out_wc1} (see Fig. 4.6(b), 4.6(c) and 4.6(d)). V_{out_wc0} and V_{out_bc0} remain desirably small because $R_{sel} = R_{sel\_eff}$ retains its high value and the small change in R_{L_eff} does not cause much change to V_{out}. As array size increases, the best case of a read-out of logic 1 fails and becomes approximately similar to the worst case read-out.

From the four read-out cases (V_{out_wc1}, V_{out_bc1}, V_{out_wc0} and V_{out_bc0}) of arrays greater than 1, four different cases of read margin (Eqn. 4.28) can also be calculated based on unique combination of these extreme scenarios according to Eqn. 4.31 - 4.34. Eqn. 4.30 represents the ideal case of a single memristor without interference from other cells.

$$\Delta V_{out\_c0} = V_{out\_nc1} - V_{out\_nc0} \tag{4.30}$$

$$\Delta V_{out\_c1} = V_{out\_wc1} - V_{out\_wc0} \tag{4.31}$$

$$\Delta V_{out\_c2} = V_{out\_wc1} - V_{out\_bc0} \tag{4.32}$$

$$\Delta V_{out\_c3} = V_{out\_bc1} - V_{out\_wc0} \tag{4.33}$$

$$\Delta V_{out\_c4} = V_{out\_bc1} - V_{out\_bc0} \tag{4.34}$$

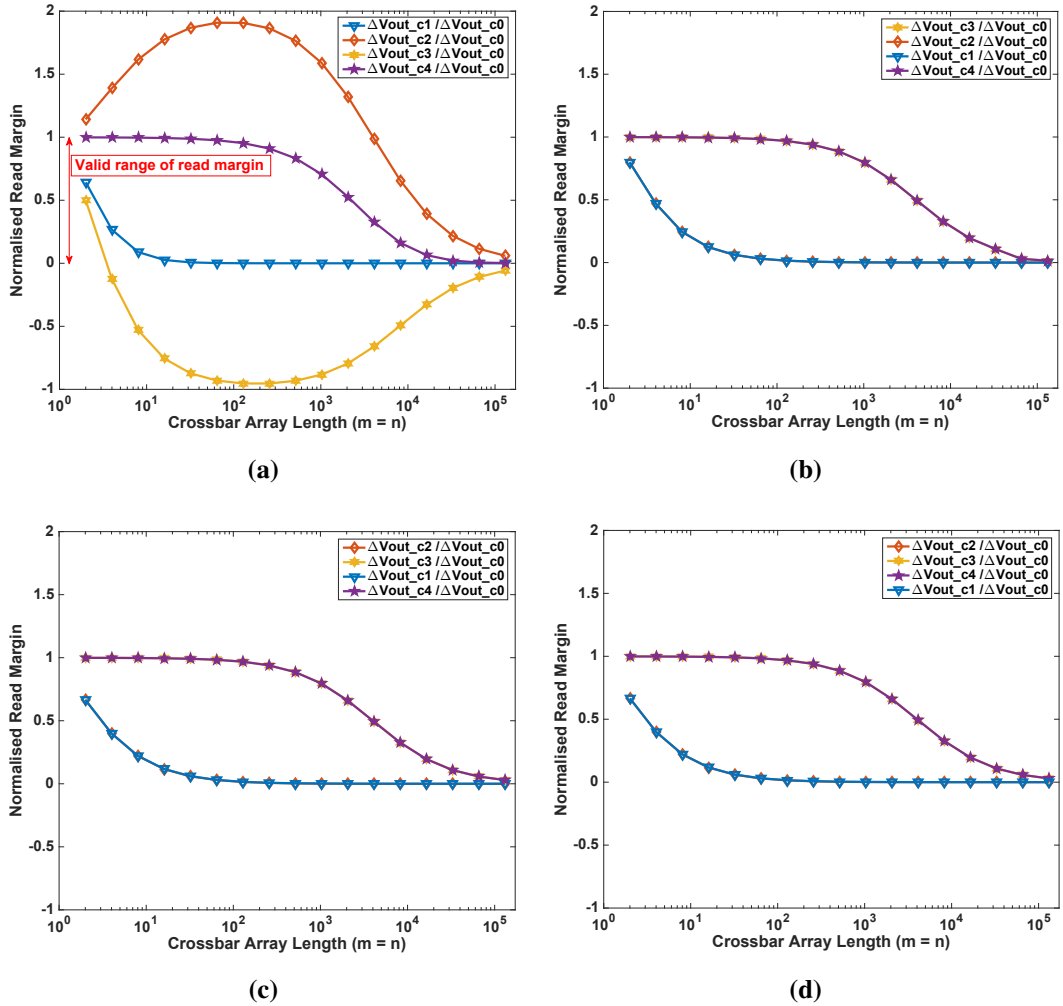


Figure 4.7: Simulation results of worst and best case read margins for (a) FWFB (b) FWGB (c) GWFB (d) GWGB. Read voltage $V_{read} = 1V$, $R_{off} = 200K\Omega$, $R_{on} = 100\Omega$ and $R_L = 100\Omega$. Data for plots was generated using the read margin values from the analytic tools listed in Appendix B.1, B.2, B.3 and B.4.

In order to fairly measure the read margin cases of the crossbar array across the four read schemes relative to a single memristor, we define a normalised figure of merit V_{out_norm} similar to previous work in [81] according to Eqn. 4.35

$$\Delta V_{out\_norm} = \frac{\Delta V_{out\_c<x>}}{\Delta V_{out\_c0}} \tag{4.35}$$

Where x in $\Delta V_{out\_c<x>}$ can range from 1 to 4 as defined in Eqn. 4.31 - 4.34

Fig. 4.7(a) - 4.7(d) show the simulation outcome of the four read margin cases on each of the read scheme. For the FWFB, only one read margin case (ΔV_{out_c4}) has optimal result before the crossbar array size approaches approximately 300×300. After this point, all the four cases begin to converge to 0 as the array size grows. ΔV_{out_c2} and ΔV_{out_c3} also deteriorate beyond the valid region because their read margin is a function of worst case

and best case read out voltages according to Eqn. 4.32 and 4.33. The probability of read failure in the FWFB scheme is 0.75 at an array size of 300×300. This probability rises to 1 swiftly as array size increases. The 'out of valid region' bend in the FWFB also further proves that the memory's behaviour could become unpredictable with this scheme in its worst case. The negative read margin of ΔV_{out_c3} is as a result of the expected low voltage V_{out_wc0} rising higher than the expected high voltage V_{out_bc1}. The grounded schemes have a read failure probability of 0.5 at array size of 300×300 or below as two cases (ΔV_{out_c3} and ΔV_{out_c4}) are well within the optimum normalised read margin of 1. The probability however degrades steadily at array size above 300×300. The GWFB and GWGB schemes will have the same read-out voltages and read margin measurements provided that $m = n$ as they are affected by R_n and R_m according to Eqn. (4.26b) and (4.27b) respectively. They will however show different behaviour once the aspect ratio of the array deviates from 1. The FWGB scheme shows little but negligible improvement over the similar GWFB and GWGB schemes.

Also examined is the case where the neighbouring groups have random numbers of high and low resistance cells that are not predetermined. For this simulation, a random number of memristors in each group of interfering neighbours are set to R_{off} state while the others are left in R_{on} state. This randomisation is made possible using the robust closed form formulas for R_m, R_n and R_{mn} derived in Eqn. 4.8- 4.11 where all the unselected memristors have differing resistance state. To constrain the randomness, we have assumed that for every random generation, the target cell is toggled between R_{on} and R_{off} and the read-out voltage measured with the same set of random neighbours. Fig. 4.8(a) - 4.8(h) shows the read margin result of simulating 1 million random samples of each array size using the FWFB read scheme. A worst case read out of '1' and '0' were used to compute the read margin ($V_{out\_wc1} - V_{out\_wc0}$). There is a large concentration of arrays with almost perfect read margin at an array size of 3×3 similar to the derivation in Eqn. 4.23. At an array size of 512×512, almost all the arrays have moved towards the midpoint of 0 where the read margin is at the minimum. Similar observations were made using the FWGB and GWGB read schemes as shown in Fig. 4.9 and 4.10 respectively. Table 4.1 shows a comparison of the four read schemes.

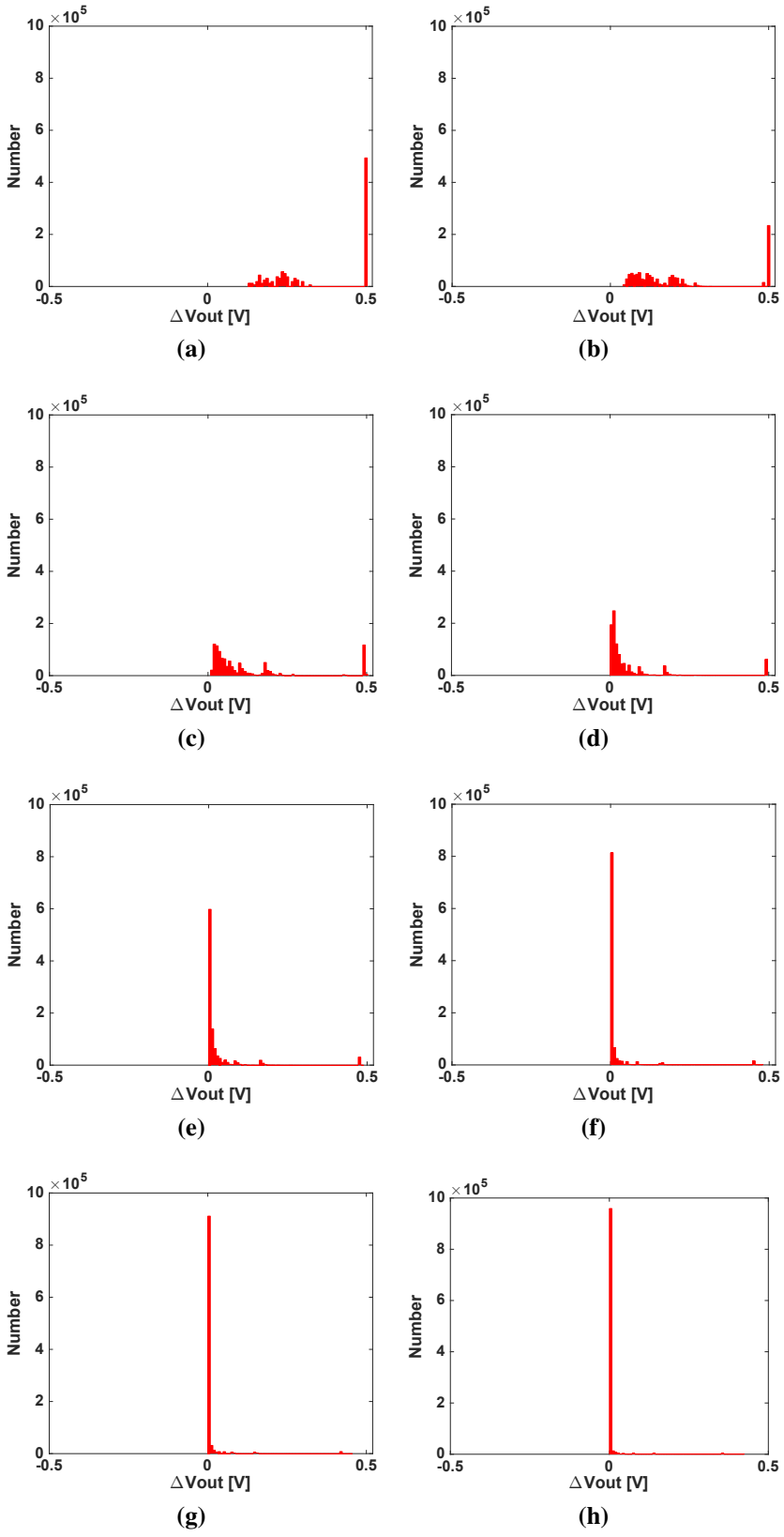


Figure 4.8: Read margin result of randomising the number of R_{off} and R_{on} cells in different array sizes using the FWFB read scheme. (a) 4×4 **(b)** 8×8 **(c)** 16×16 **(d)** 32×32 **(e)** 64×64 **(f)** 128×128 **(g)** 256×256 **(h)** 512×512**. Data for plots was generated using the read margin values from the analytic tools listed in Appendix B.5.**

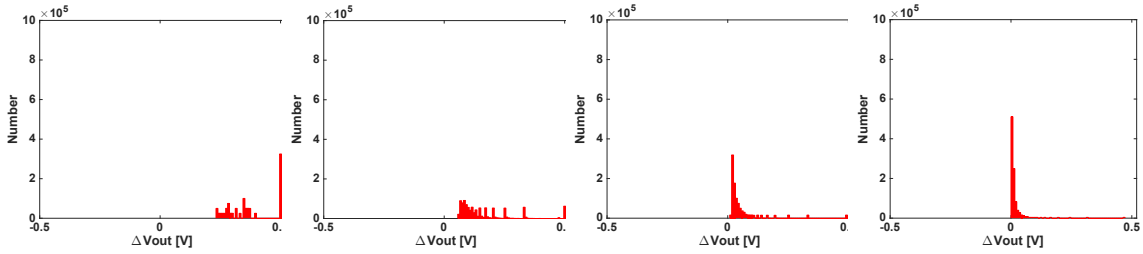


Figure 4.9: Read margin result of randomising the number of R_{off} and R_{on} cells in different array sizes using the FWGB read scheme. Read margin degrades as crossbar array size increases. (a) 4×4 (b) 16×16 (c) 64×64 (d) 256×256. Data for plots was generated using the read margin values from the analytic tools listed in Appendix B.6.

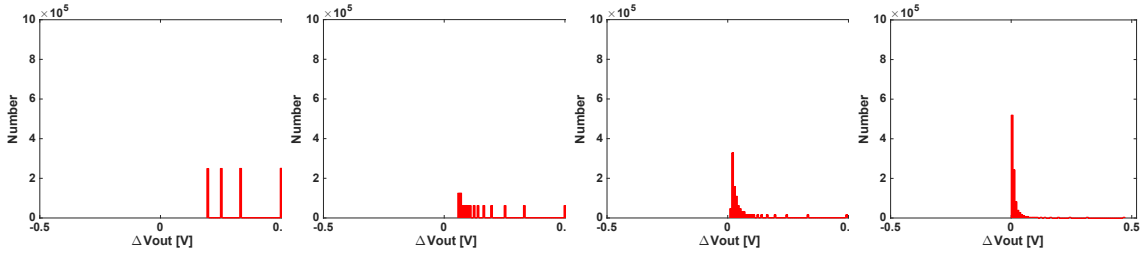


Figure 4.10: Read margin result of randomising the number of R_{off} and R_{on} cells in different array sizes using the GWGB read scheme. Read margin degrades as crossbar array size increases. (a) 4×4 (b) 16×16 (c) 64×64 (d) 256×256. Data for plots was generated using the read margin values from the analytic tools listed in Appendix B.6.

Table 4.1: Comparison summary of read schemes performance

	Unselected Wordline	Unselected Bitline	Effective Resistance of Selected Cell	Effective Resistance of Load Resistor	Ranking of Noise Margin Cases (Worst → Best)	Power Usage	Number of Sneak-path cells
FWFB	Floating	Floating	$R_{sel} \mid\mid (R_m + R_{mn} + R_n)$	NULL	$\Delta V_{out\_c2} \mid\mid \Delta V_{out\_c3} \rightarrow \Delta V_{out\_c1} \rightarrow \Delta V_{out\_c4}$	Low	$mn-1$
FWGB	Floating	Ground	NULL	$R_L \mid\mid (R_m + R_{mn})$	$\Delta V_{out\_c1} \mid\mid \Delta V_{out\_c2} \rightarrow \Delta V_{out\_c3} \mid\mid \Delta V_{out\_c4}$	High	$n(m-1)$
GWFB	Ground	Floating	NULL	$R_L \mid\mid R_n$	$\Delta V_{out\_c1} \mid\mid \Delta V_{out\_c2} \rightarrow \Delta V_{out\_c3} \mid\mid \Delta V_{out\_c4}$	High	$n-1$
GWGB	Ground	Ground	NULL	$R_L \mid\mid R_m$	$\Delta V_{out\_c1} \mid\mid \Delta V_{out\_c2} \rightarrow \Delta V_{out\_c3} \mid\mid \Delta V_{out\_c4}$	High	$m-1$

Tables in appendix A.1 shows the data generated from simulating crossbar arrays with non-square aspect ratio. This differs from previous simulation in its array sizing, m and n are varied in order to observe the difference between GWGB and GWFB that always generate similar result when $m = n$. Result from the data shows that when $m \gg n$, GWFB is a better read scheme than GWGB as GWFB is not affected by the increased $m-1$ number

of memristors affected by sneak-path current. Similarly, when $m \ll n$, GWGB becomes the better option as this scheme is immuned from sneak-paths from the high number of $n-1$ cells. The FWFB and FWGB read schemes show the usual deteriorating trend as they are both affected negatively by changes in both the dimensions of the crossbar array.

4.5 Power Analysis of Crossbar Array Read Schemes

The average power consumed by the read operation can be computed using the estimation from [103]. We estimated the total resistance R_{total} in each circuit during the read operation. The average power can be calculated as shown below:

$$P = \frac{V_{read}^2}{R_{total}} \tag{4.36}$$

R_{total} for each scheme was calculated as shown in Eqn. (4.24c), (4.25c), (4.26c) and (4.27c). Fig. 4.11 shows the power consumption metrics of the four schemes. The first observation from this simulation is that a read-out of '1' consumes more power than a read-out of '0' simply because more current flows through the circuit if the resistance of the selected device is low. Preliminary simulation also shows that the worst case power consumption during the read-out of '1' and '0' occurs when the neighbouring cells are in the low resistance state. Higher concentration of low resistive cells in the crossbar array heightens current leakages which also leads to excessive power consumption. The result also shows that FWFB saves more power than any of the other schemes. Although the three schemes that involve grounding the lines minimise sneak-paths, they however consume power incrementally as the crossbar size increases.

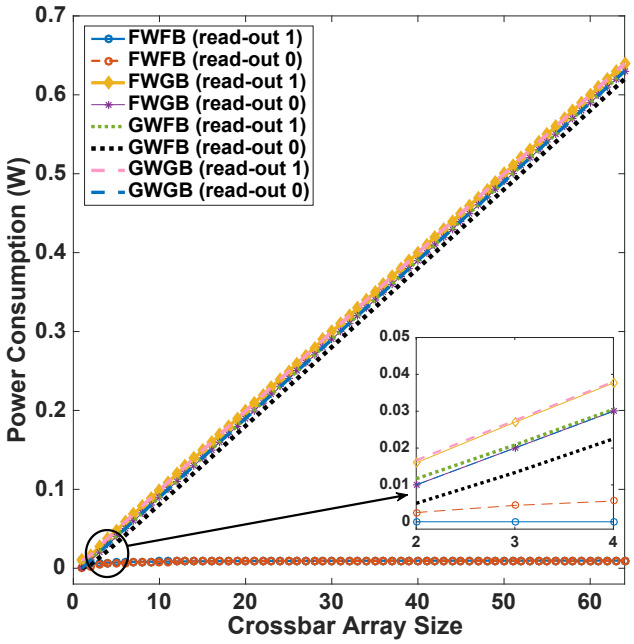


Figure 4.11: Power comparison of the four read schemes over a range of crossbar size. Data for plot was generated using the power values from the analytic tools listed in Appendix B.1, B.2, B.3 and B.4

4.6 Multiple Cells Read in Crossbar Arrays

Analysis of read operation in crossbar arrays has so far been focussed on reading a single cell at a time from the crossbar array. However, in practical and large scale applications, multiple cells are accessed in each read cycle. This section focusses on multiple cells read-out in crossbar array which invariably leads to the minimisation of the sneak-path effects in crossbar array. The concept of multiple cells read-out in resistive array was first introduced in [68] with limited analysis. A detailed analysis of the read method, challenges and simulation results are still missing. This section provides a comprehensive analysis of reading multiple cells in a crossbar array simultaneously alongside its effects on read margin and power consumption. Multiple cells can be selected for reading by connecting the target row to the read voltage and the columns with the target cell are connected to the sensing circuit, while other unselected lines are left floating. If k cells are selected for reading such that $1 < k < n$, the array can be regrouped into four blocks as shown in Fig. 4.12(a) with an instance of $k = 2$. Fig. 4.12(b) shows the case where $k = n$ cells are selected for read. The effectiveness of the multiple cells read technique in minimising sneak-paths is dependent on the data distribution in the selected cells and the number of cells selected, these will be further explained in subsequent sections.

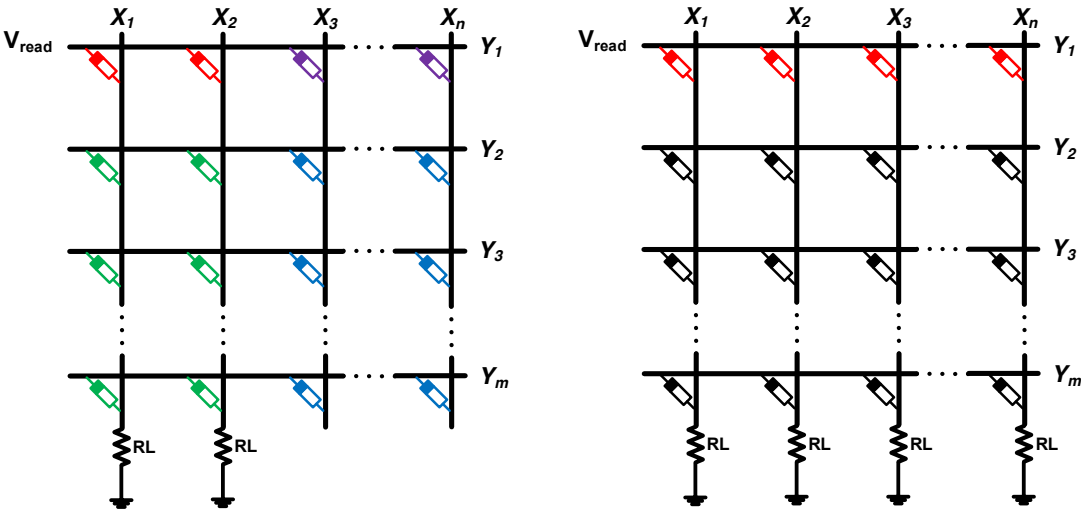


Figure 4.12: Description of read operation in cases where some of the cells in row are selected. (a) Schematic of an $m \times n$ array with $k = 2$ cells selected for reading. The red cells are the selected cells. Cells in purple and green are the partially-selected cells across the selected row and column respectively. (b) Schematic of an $m \times n$ memristor array with $k = n$ cells selected for simultaneous read out.

4.6.1 Multiple Cells Read with Similar Data

Multiple cells read operation can either be a selection of some of the cells $1 < k < n$ or the selection of all the cells $k = n$ in a row for simultaneous read. In order to understand the circuit behaviour during multiple cells read out in a $m \times n$ memory array, we start with a partial selection of some of the cells in the row. Let us consider the crossbar array of Fig. 4.12(a), where two cells are selected and all the k selected cells store the same data such that $R_{1,1} = R_{1,2}$. A crossbar with this selection can be represented by the equivalent circuit model of Fig. 4.13(a) where R_{sel_k}, R_{m_k}, R_{n_k} and R_{mn_k} are computed according to Eqn. 4.37, 4.38, 4.39 and 4.40 respectively. These equations are the more generalised forms of Eqn. 2.14 - 2.16. Note that these equations are still applicable when $k = 1$. The read-out voltage V_{out} from each selected cell can be computed by solving for V_{out} in the equivalent circuit model. Similarly, when all the cells in the row are selected for read $k = n$ such that $R_{1,1} = R_{1,2} = ... = R_{1,n}$, Eqn. 4.37, 4.38, 4.39 and 4.40 still stand but R_{n_k} and R_{mn_k} approach infinity because of their zero denominator. R_{m_k} will be shunted out of the equivalent circuit to result in the reduced circuit of Fig. 4.13(b). V_{out} from the circuits in Fig. 4.13(a) and Fig. 4.13(b) is the average read out voltage from all the selected memristor in the row, hence each selected memristor have a read-out voltage of exactly V_{out} because they have same resistance values.

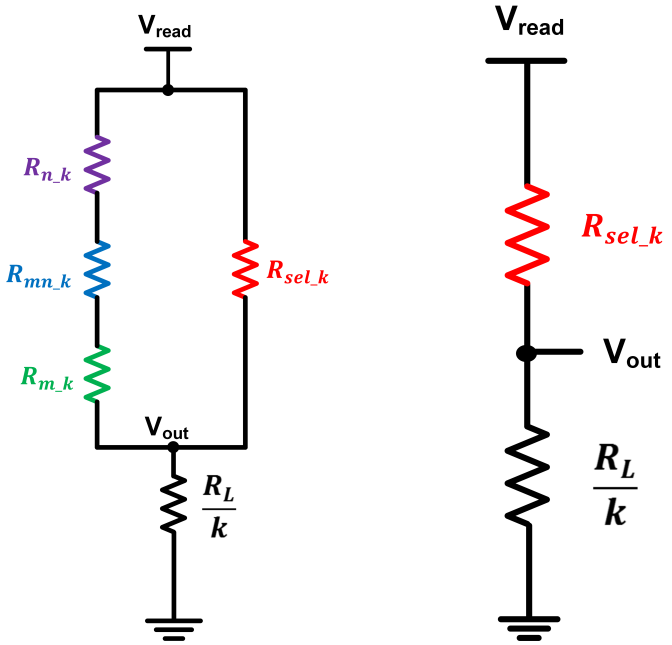


Figure 4.13: Corresponding generalised equivalent circuits for multiple cells selection where all the selected cells contain the same data. (a) Resistance model of the crossbar array selection in Fig. 4.12(a) for any $1 < k < n$ (b) Reduced equivalent resistance model of the crossbar array selection in Fig. 4.12(b) where all the cells in the row are selected for read, $k = n$. V_{out} is the read-out voltage of each of the k selected cells.

$$R_{sel\_k} = \frac{R}{k} \tag{4.37}$$

$$R_{m\_k} = \frac{R}{(m-1)(k)} \tag{4.38}$$

$$R_{n\_k} = \frac{R}{(n-k)} \tag{4.39}$$

$$R_{mn\_k} = \frac{R}{(m-1)(n-k)} \tag{4.40}$$

Note that Eqn. 4.37 - 4.40 only hold on the assumption that all $k > 1$ selected cells contain the same logic value. V_{out} for the equivalent circuit of Fig. 4.13(a) can be calculated using the equations below:

$$R_{sel\_sneak} = R_{m\_k} + R_{n\_k} + R_{mn\_k} \tag{4.41}$$

$$R_{sel\_eff} = R_{sel\_k} \,||\, R_{sel\_sneak} \tag{4.42}$$

$$V_{out} = V_{read} \times \frac{R_L/k}{R_L/k + R_{sel\_eff}} \tag{4.43}$$

Fig. 4.14 shows the gradual improvement in read margin as the number of cells k selected for reading increases in a 8×8 memristor array. The read margin is at its best when $k = n$ because the sneak current resistance R_{m_k}, R_{n_k} and R_{mn_k} are shunted out of the model at this stage as shown in Fig. 4.13(b). Sneak-path effects are completely eliminated from the array when $k = n$ based on the assumption that the selected cells store the same logic value of either 0 or 1, each of the selected cells thus behaves as if it is being accessed without the presence of the neighbouring cells. The worst case read margin is at its worst point when $k = 1$ as expected because of its impact on R_{m_k}, R_{n_k} and R_{mn_k}. As previously defined, the read margin is the difference between the output voltages when the selected cell(s)

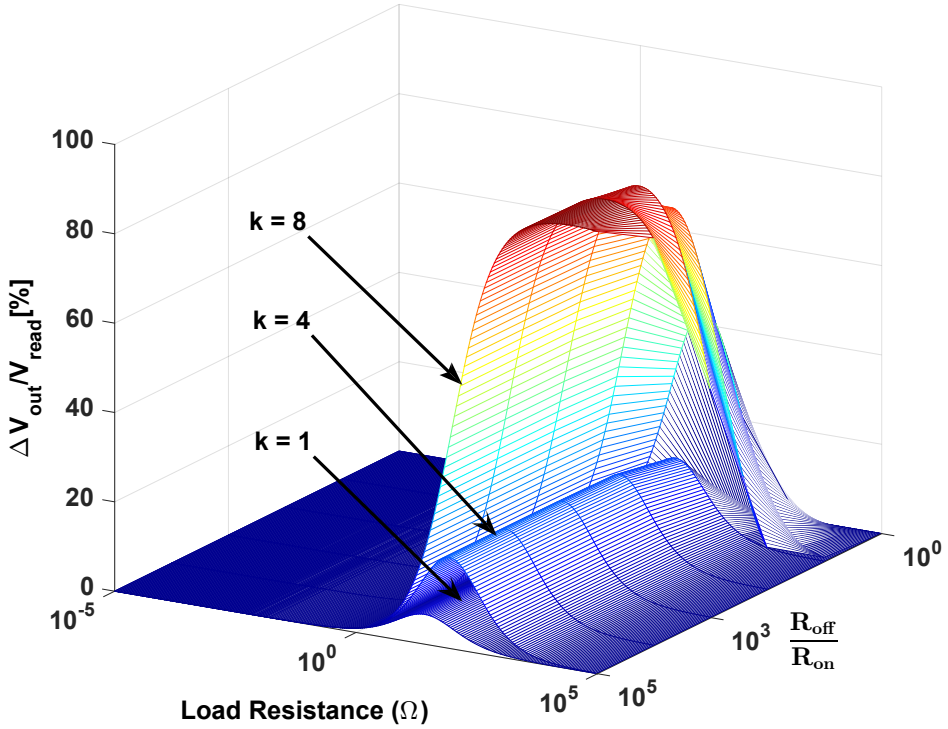


Figure 4.14: Read margin for each of the k selected cells in an 8×8 memristor array when $k = 1$, $k = 4$ (50% cells selected in selected row) and $k = 8$ (100% cells selected in selected row). sneak-path is completely eliminated when $k = 8$ because the sneak current resistances are shunted out of the circuit. $V_{read} = 1$, $R_{on} = 100\Omega$, R_{off} is varied between 10^2 and $10^7\Omega$. Load resistance is optimum when it is similar to R_{on}.

Fig. 4.14 also shows that the load resistance is optimum when it is approximately the same value as R_{on}. The R_{off}/R_{on} ratio is also optimum when its greater than 10^2.

4.6.2 Multiple Cells Read with Non Similar Data

In the case where the k selected cells do not contain the same value, V_{out} for each of the selected cells cannot be solved using the equivalent circuits of Section 4.6.1. This is

because the selected cells cannot be combined in parallel into a single factored resistance as previously done. Since the data in the selected memristors differ, it is impossible to correctly allocate output value to each memristor from the total average read-out as was done in Section 4.6.1. The read-out voltage from each cell would have to be solved independently using the equivalent circuit described by Fig. 4.15(a) when all of the cells are selected for simultaneous read as depicted by the crossbar of Fig. 4.12(b). Here, R_m, R_n and R_{mn} are previously defined by Eqn. 4.12, 4.13 and 4.14 respectively. The analysis of the output of each cell works by considering each selected cell with its load resistance independently. The other selected $n-1$ ($k=n$) cells and their load resistances still form a part of the parasitic resistance that negatively impacts the circuit.

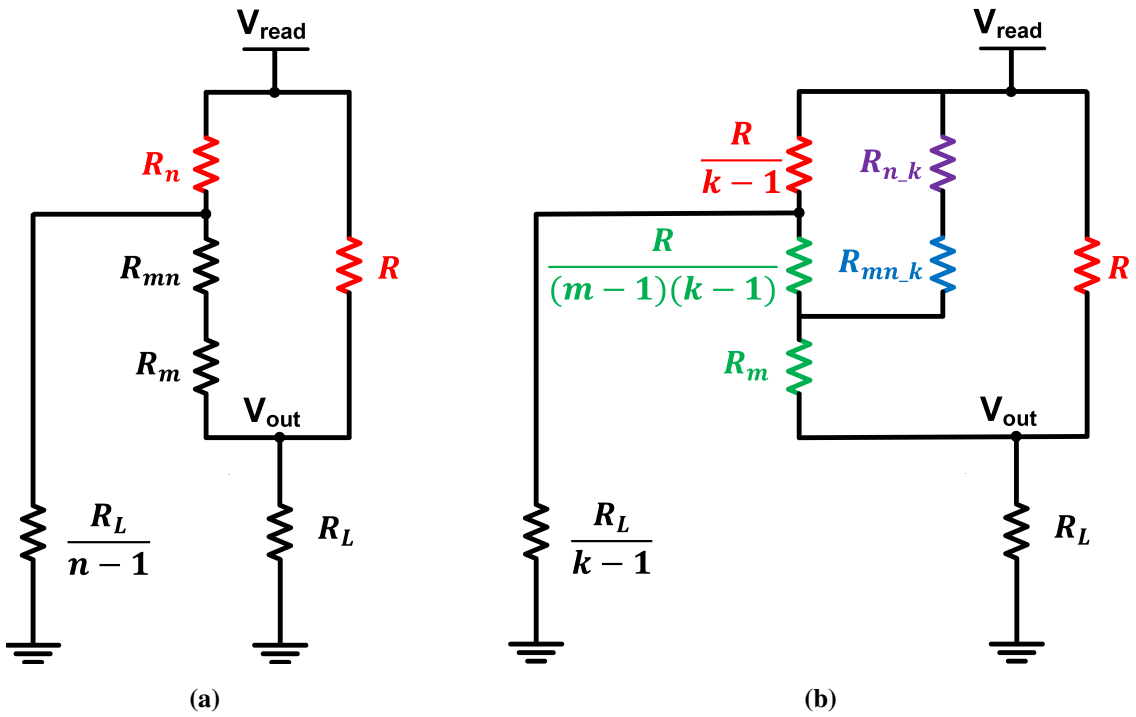


Figure 4.15: Corresponding generalised equivalent circuits for multiple cells selection where all the selected cells does not contain the same data. (a) Equivalent resistance model of the crossbar array selection in Fig. 4.12(b) where all the cells in the rows are selected for read, $k=n$. R is the resistance of any one of the selected n cells, R_n represents the resistance of other $n-1$ selected cells (b) Resistance model of the crossbar array selection in Fig. 4.12(a), for any selected number of k between 2 and n, R is the resistance of any one of the selected n cells, $R/k-1$ represents the resistance of other $k-1$ selected cells. V_{out} is the read-out voltage of each of the k selected cells.

The readout voltage V_{out} in the equivalent circuit of Fig. 4.15(a) can be solved using the $Y-\Delta$ transformation technique. R_{mn} combines in series with R_m to form a Y circuit with R_{Ln} ($R_{Ln}=R_L/(n-1)$) and R_n as shown in Fig. 4.16.

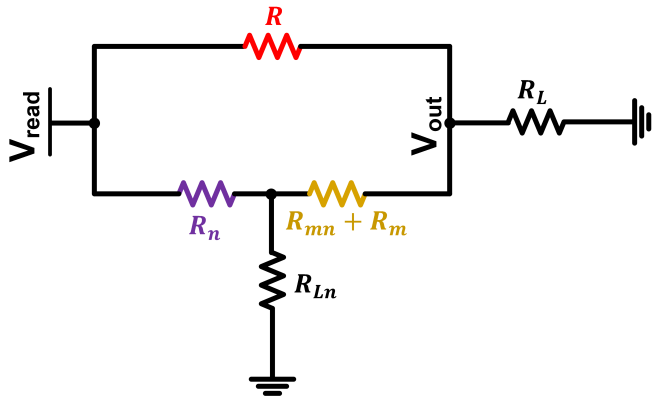


Figure 4.16: Step 1: Solving for V_{out} in the crossbar array of Fig. 4.12(b) from the equivalent circuit of Fig. 4.15(a) using $Y-\Delta$ transformation technique.

The Y circuit is transformed into a Δ circuit as shown in Fig. 4.17. $R_a(R_{sel\_sneak})$, R_b and $R_c(R_{L\_sneak})$ are computed according to Eqn 4.44, 4.45 and 4.46 respectively.

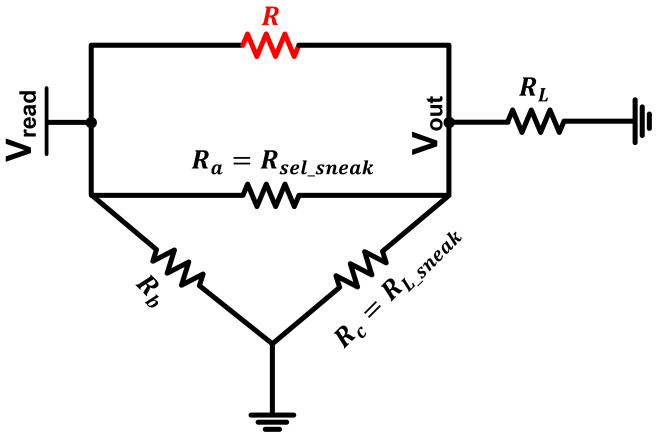


Figure 4.17: Step 2: Solving for V_{out} in the crossbar array of Fig. 4.12(b).

$$R_{sel\_sneak} = \frac{R_n(R_{mn}+R_m)+\dfrac{R_L}{n-1}(R_{mn}+R_m)+R_n\dfrac{R_L}{n-1}}{\dfrac{R_L}{n-1}} \tag{4.44}$$

$$R_b = \frac{R_n(R_{mn}+R_m)+R_{Ln}(R_{mn}+R_m)+R_nR_{Ln}}{R_{mn}+R_m} \tag{4.45}$$

$$R_{L\_sneak} = \frac{R_n(R_{mn}+R_m)+\dfrac{R_L}{n-1}(R_{mn}+R_m)+R_n\dfrac{R_L}{n-1}}{R_n} \tag{4.46}$$

R_{L_sneak} forms a parallel resistance with the load resistor R_L as depicted by Fig 4.18 while R_{sel_sneak} combines in parallel with the selected cell R_{sel} to result in the simple circuit shown in Fig 4.19. R_b is shunted out of the circuit and does not contribute to V_{out}. This

reduced circuit allows easy computation of the output voltage for each of the selected cell in crossbar array of Fig. 4.12(b).

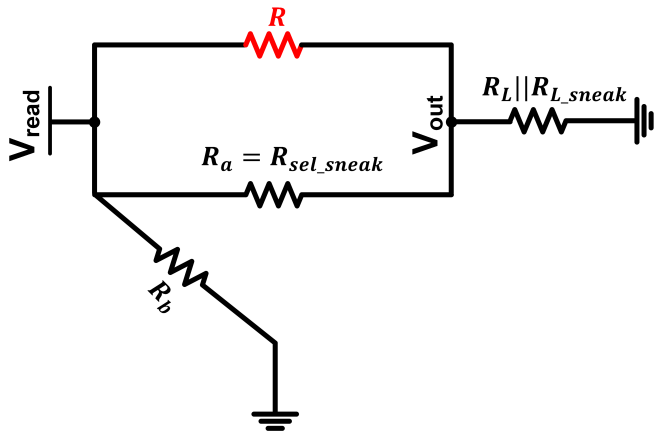


Figure 4.18: Step 3: Solving for V_{out} in the crossbar array of Fig. 4.12(b).

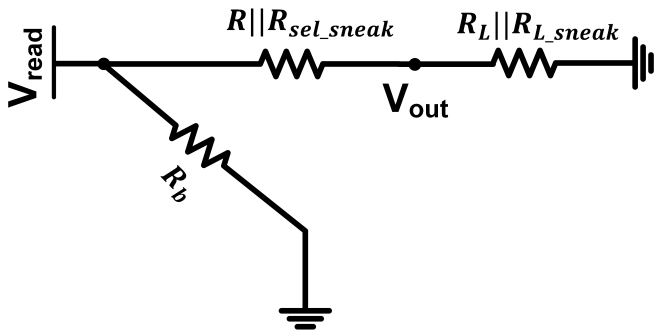


Figure 4.19: Step 4: Solving for V_{out} in the crossbar array of Fig. 4.12(b).

In the case where $k < n$, that is, only some of the cells in the row are selected for simultaneous read as in the crossbar array of Fig. 4.12(a), the read-out voltage of each memristor can also be computed independently using the crossbar's equivalent circuit model depicted by Fig. 4.15(b). For the sake of simplicity, it is assumed that the other groups of cells in green, purple and blue contain the same resistance value R. R_{n_k} and R_{mn_k} in Fig. 4.15(b) are as defined by Eqn. 4.39 and 4.40.

To solve V_{out} in the circuit of Fig. 4.15(b), we make the following representation for brevity, $R_k = R/(k-1)$, $R_{mk} = R/(m-1)(k-1)$ and $R_{Lk} = R_L/(k-1)$. Using $Y-\Delta$ transformation, R_k, R_{mk} and R_{Lk} forms a Y circuit that can be transformed into a Δ circuit as shown in Fig. 4.20. R_a, R_b and R_c can be represented as Eqn. 4.47, 4.48 and 4.47 respectively:

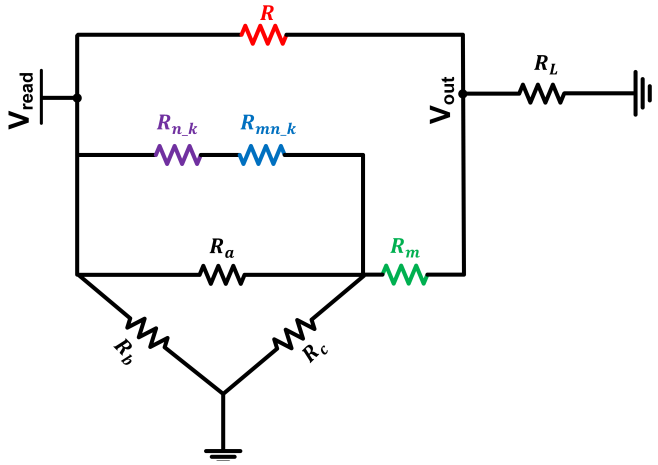


Figure 4.20: Step 1: Solving for V_{out} in the crossbar array of Fig. 4.12(a) from the equivalent circuit of Fig. 4.15(b) using $Y-\Delta$ transformation technique.

$$R_a = \frac{R_k \times R_{mk} + R_{Lk} \times R_{mk} + R_{Lk} \times R_k}{R_{Lk}} \tag{4.47}$$

$$R_b = \frac{R_k \times R_{mk} + R_{Lk} \times R_{mk} + R_{Lk} \times R_k}{R_{mk}} \tag{4.48}$$

$$R_c = \frac{R_k \times R_{mk} + R_{Lk} \times R_{mk} + R_{Lk} \times R_k}{R_k} \tag{4.49}$$

R_a forms a parallel resistance with the series combination of R_{n_k} and R_{mn_k} to result in R_{a_new} (Eqn. 4.50) shown in the circuit of Fig. 4.21.

$$R_{a\_new} = \frac{R_a \times (R_{mnk} + R_{nk})}{R_a + (R_{mnk} + R_{nk})} \tag{4.50}$$

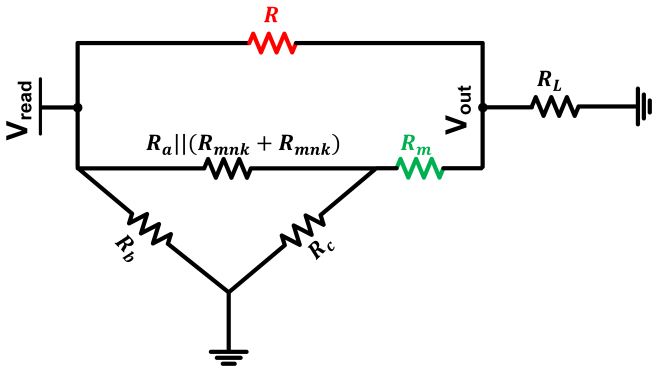


Figure 4.21: Step 2: Solving for V_{out} in the crossbar array of Fig. 4.12(a).

The Δ circuit in Fig. 4.21 can be transformed back into a Y circuit so as to combine R_m in series with one of the branches as show in Fig. 4.22. R_1, R_2 and R_3 are computed

according to Eqn. 4.51, 4.52 and 4.53 respectively:

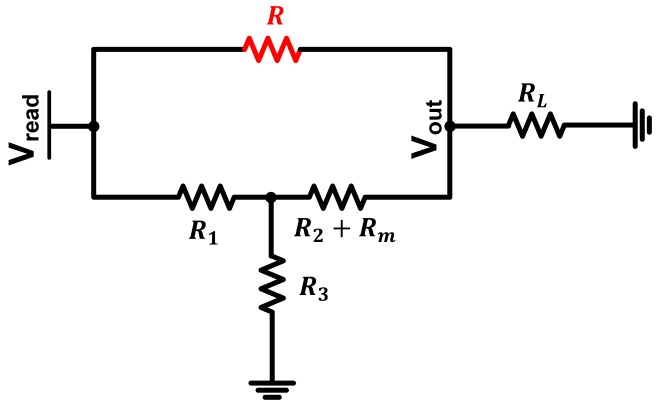


Figure 4.22: Step 3: Solving for V_{out} in the crossbar array of Fig. 4.12(a).

$$R_1 = \frac{R_{a\_new} \times R_b}{R_{a\_new} \times R_b + R_c} \tag{4.51}$$

$$R_2 = \frac{R_{a\_new} \times R_c}{R_{a\_new} \times R_b + R_c} \tag{4.52}$$

$$R_3 = \frac{R_b \times R_c}{R_{a\_new} \times R_b + R_c} \tag{4.53}$$

R_m combine in series with R_2 to result in R_{2_new}:

$$R_{2\_new} = R_2 + R_m \tag{4.54}$$

The current circuit can be further simplified by transforming the Y circuit to a Δ circuit as shown in Fig. 4.23.

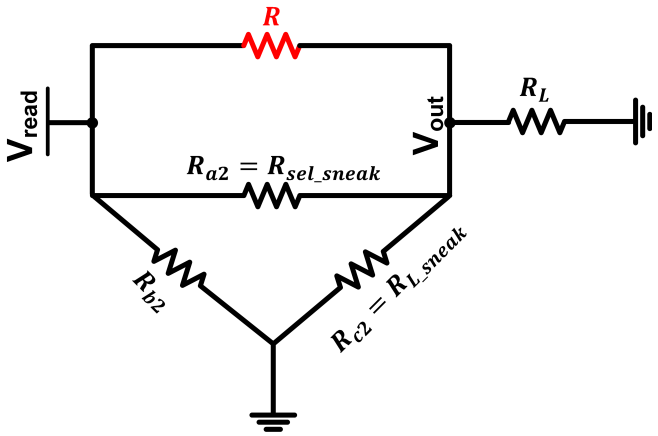


Figure 4.23: Step 4: Solving for V_{out} in the crossbar array of Fig. 4.12(a).

R_{L_sneak}, R_{b2} and R_{sel_sneak} are calculated as:

$$R_{sel\_sneak} = \frac{R_1 \times R_{2\_new} + R_{2\_new} \times R_3 + R_1 \times R_3}{R_3} \tag{4.55}$$

$$R_{b2} = \frac{R_1 \times R_{2\_new} + R_{2\_new} \times R_3 + R_1 \times R_3}{R_{2\_new}} \tag{4.56}$$

$$R_{L\_sneak} = \frac{R_1 \times R_{2\_new} + R_{2\_new} \times R_3 + R_1 \times R_3}{R_1} \tag{4.57}$$

R_{L_sneak} and R_{sel_sneak} form a parallel circuit with the load resistance R_L and selected resistance R respectively as portrayed in Fig. 4.24 and 4.25 respectively.

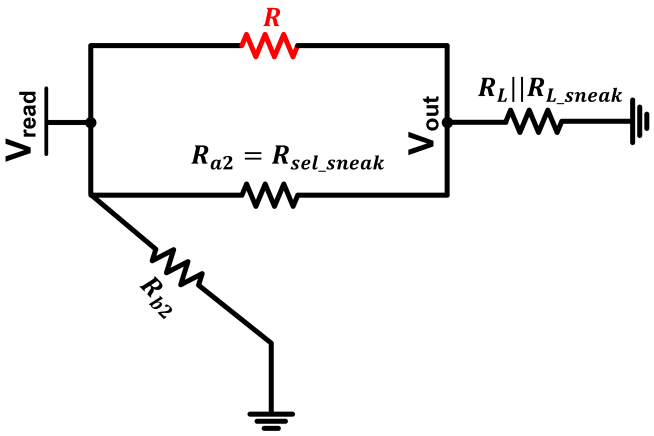


Figure 4.24: Step 5: Solving for V_{out} in the crossbar array of Fig. 4.12(a).

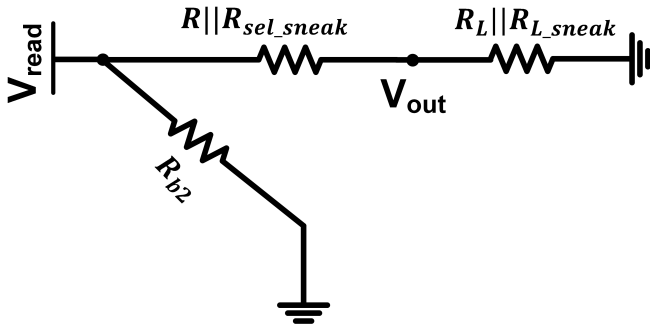


Figure 4.25: Step 6: Solving for V_{out} in the crossbar array of Fig. 4.12(a).

V_{out} for the circuit in Fig. 4.19 and 4.25 can thus be calculated as the voltage divider between $R||R_{sel\_sneak}$ and $R_L||R_{L\_sneak}$ while R_{b2} and R_b are shunted to ground without affecting V_{out}. Note that the equations for R_{sel_sneak} and R_{L_sneak} differ for each of the two circuits.

The effective load resistance (R_{L_eff}) thus becomes:

$$R_{L\_eff} = R_L \,||\, R_{L\_sneak} \tag{4.58}$$

and the effective resistance of each of the selected memristor from the memory array can be computed as:

$$R_{sel\_eff} = R \,||\, R_{sel\_sneak} \tag{4.59}$$

Therefore:

$$V_{out} = V_{read} \times \frac{R_{L\_eff}}{R_{L\_eff} + R_{sel\_eff}} \tag{4.60}$$

Having worked out the read-out voltages from each of the selected cells, further analysis was carried out to establish the effect of multiple cells selection on the read margin of the crossbar array when the selected cells have non-similar values. It is important to note that the worst and best case read of '1' in the multiple cells read operation differs from the previous case of reading from a cell.

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \end{bmatrix}$$

(a) **(b)** **(c)** **(d)**

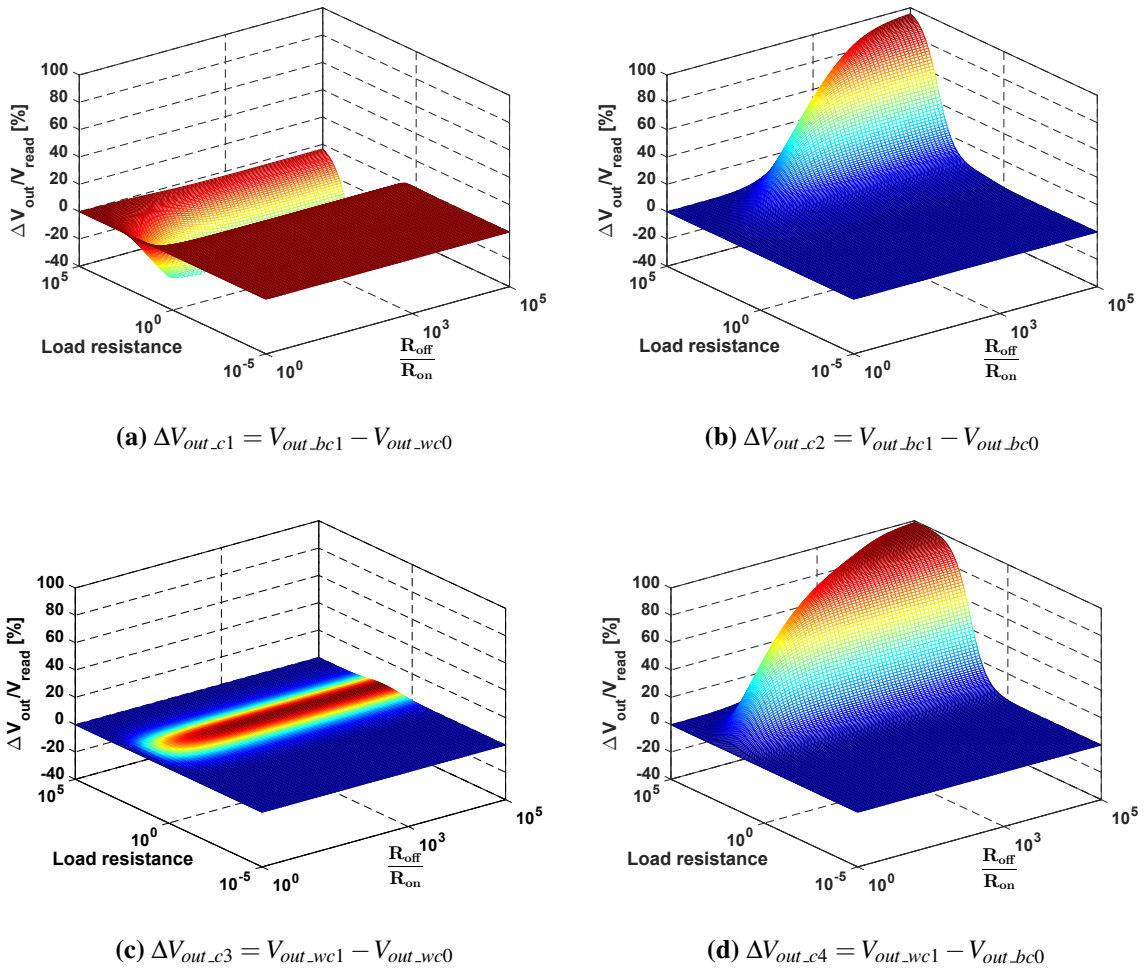
$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(e) **(f)**

Figure 4.26: Sample data distribution in a 4×4 crossbar array during multiple cells read where all the cells in the first row are selected for read. (a) Worst case read-out of zero (V_{out_wc0}) (b) Worst case read-out of one (V_{out_wc1}) (c) Best case read-out of zero (V_{out_bc0}) (d) Best case read-out of one (V_{out_bc1}) (e) Worst case read-out of one and zero with equal distribution of 1s and 0s (f) Best cases of read-out of one and zero with equal distribution of 1s and 0s.

When reading a single cell, the worst case read of '1' occurs when all the other unselected cells store a '1' and the best case when all other cells store a '0'. With multiple cells read, the worst case read of '1' occurs when the other selected cells in the selected row store a '0' and other unselected cells have a logic '1'. The best case read of '1' occurs when the other selected cells contain a '1' irrespective of the contents of the unselected cells. Fig. 4.26 gives a pictorial illustration of the various cases of reading a '1' and a '0' when all the cells in the first row are being read. The figures in Fig. 4.27 show the read margin results when varying the R_{off}/R_{on} ratio and load resistance value in a 32×32

crossbar array using the four cases of read margin explained in Section 4.4. Two of the read margin cases (ΔV_{out_c2} and ΔV_{out_c4}) show optimum results when the load resistance is approximately equal to R_{on} and the R_{off}/R_{on} ratio is at the maximum. ΔV_{out_c2} and ΔV_{out_c4} are similar because V_{out_wc1} is close to V_{out_bc1} but expected to widen as array size increases. Sneak-path is eliminated when all the cells in the row are selected and they store the same LRS value. The read operation will fail if the crossbar stores data that results in any of two other read margin cases (ΔV_{out_c1} and ΔV_{out_c3}). ΔV_{out_c3} in particular gives a negative read margin because the expected low read-out voltage of the OFF state is higher than the high read-out voltage of the ON state. The dynamics is expected to change as the array size grows.



(a) $\Delta V_{out\_c1} = V_{out\_bc1} - V_{out\_wc0}$

(b) $\Delta V_{out\_c2} = V_{out\_bc1} - V_{out\_bc0}$

(c) $\Delta V_{out\_c3} = V_{out\_wc1} - V_{out\_wc0}$

(d) $\Delta V_{out\_c4} = V_{out\_wc1} - V_{out\_bc0}$

Figure 4.27: Read margins of each cell in the multiple read scheme where all cells in the row are selected for read operation in a 32×32 crossbar array.

4.6.3 Application of Multiple Cells Read Technique

In order to test an application of the multiple bits read scheme especially for power consumption, a new inverted-hamming coding architecture that is able to correct single bit write and read error is proposed. Hamming codes are Single Error Correcting (SEC) binary linear block codes [104]. Normally represented as $H(n,l)$, where n is the length of the encoded word and l is the length of the data word. $n-l$ parity bits are added to the l bits of data. We started by adding extra parity bits to each row of codeword as computed using hamming code. We then inverted the entire codeword depending on the number of '1s' to be written into the memory, in order to mitigate the effect of the extra parity bits. In a case where we have more '1s' than '0s' in the write buffer, we complement and write the inverted data into the memory then the inversion flag is registered to show the codeword is an inverted data. The basic concept of Inverted-Hamming code architecture is depicted in Fig. 4.28c, the detailed peripheral and logic circuits are not shown. Appendix B.9 and B.10 shows the analytical implementation of this architecture in C++. The simulation result for the power consumption analysis is described in Section 4.6.4.

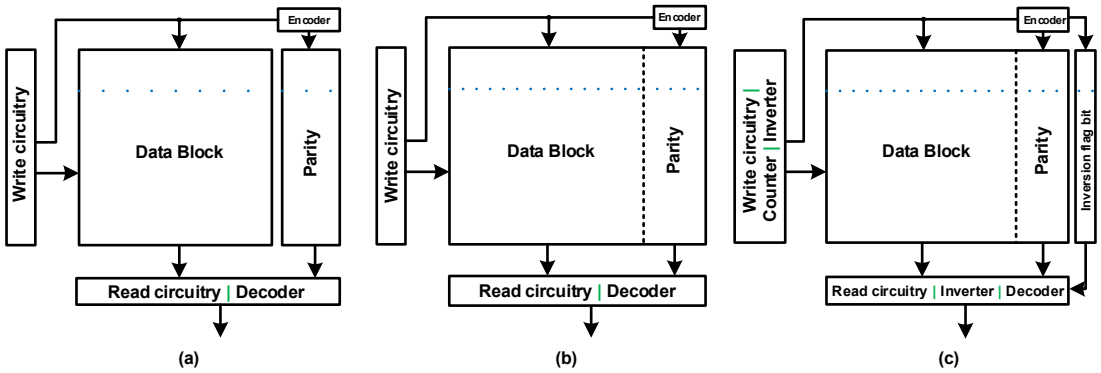


Figure 4.28: Block schematic representation of (a) Hamming code with parity block separated from the primary data block (b) Hamming code structure, parity bits implemented with data block (c) proposed Inverted-Hamming code circuit.

4.6.4 Experimental Results and Discussions

In order to simulate larger arrays, we have modelled the behaviour of the crossbar array in C++. This allows for easy simulation of larger array compared to the Cadence tool. Majority of the EDA tools are mostly suitable for CMOS device. They often require adaption in order for them to work effectively with emerging devices like the memristor and often struggles with larger circuits. To verify the correctness of the analytic tool that was used for previous experimental results, we modelled the read margin of the crossbar

arrays with multiple cells selected for read using both the analytic tool and the Cadence EDA tool. The output of our tool matched closely with the output of the simulation tool with improved accuracy as the array size grows. For all simulations in this chapter, parameters used includes a R_{off}/R_{on} ratio of 10^3 similar to the one used in [82], a load resistance of 100Ω and a read voltage of $1V$ unless otherwise stated.

The impact of sneak-path on the read margin of each of the selected cell k is individually evaluated as shown in Fig. 4.15(a). This is because the remaining $k-1$ ($k=n$) selected cells and their corresponding load resistances contribute to the sneak-paths resistance.

In order to read the output of any one of the selected cells, the resistances of the other selected cells R_{sel_sneak} (sneak-path contribution from other $k-1$ selected cells) was computed using Eqn. 4.44 as they form a part of the overall sneak resistance. The overall sneak-path resistance effect on each selected cell can thus be calculated using an adaptation of the equation described in [38] with Eqn. 4.44 as a vital component.

In order to understand the sneak-path effect when the selected cells contain non-similar data, we have simulated the architecture described in Fig. 4.15(a). With all the cells in the first row selected for reading, the read margin of three different data patterns were considered, when: 1) only one cell differs (Fig. 4.26(a) and 4.26(b)), 2) equal numbers of cells store '0' and '1' (Fig. 4.26(e) and 4.26(f)), and 3) all selected cells store same data - all 1s or 0s (Fig. 4.26(c) and 4.26(d)). The normalised read margin computed in this case is for one of the selected cells in the row, $\Delta V_{out\_array} = V_{out\_wc1} - V_{out\_wc0}$ (crossbar of size greater than 1) and $\Delta V_{out\_single} = V_{out\_1} - V_{out\_0}$ (a crossbar of size of 1). It is worthy of mention that the worst case read-out voltage of 1 in the multiple cells read occurs when other neighbouring cells stores 0 unlike the usual single read where the worst case occurs when the neighbouring cells stores 1. This is because when reading a 1 in multiple cells selection and other neighbours contain a 1, sneak- path is eliminated as discussed in Section 4.6.1 and thus make a best case read of 1.

Fig. 4.29 shows the result of our analytic tool discussed in Section 4.6 against the output of the simulation tool for up to an array size of 128×128. The output of both tools are approximately similar and improves as the array size grows. Fig. 4.30 shows the outcome of multiple cells read scheme ($k=n$) over larger array sizes. The read margin remains optimal when the cells store uniform data even as the array size increases. This is not the case for the two other data patterns. The read margin remains at an acceptable level of over 30% with checkered pattern for an array size of 1024×1024. The read margin

in Fig. 4.30 was computed using the worst case read scenario. For a cell storing a 0, the worst case occurs when all other neighbouring cells contain a 1 (Fig. 4.26(a)) and for a cell storing a 1, the worst case occurs when all other unselected cells contain a 1 (Fig. 4.26(b)).

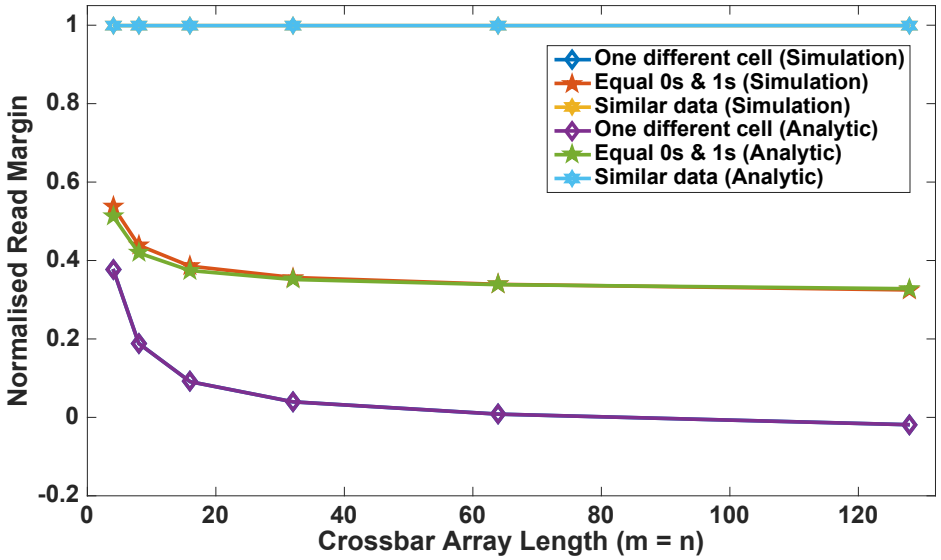


Figure 4.29: Comparison of normalised read margin result from analytic tool against simulation tool.

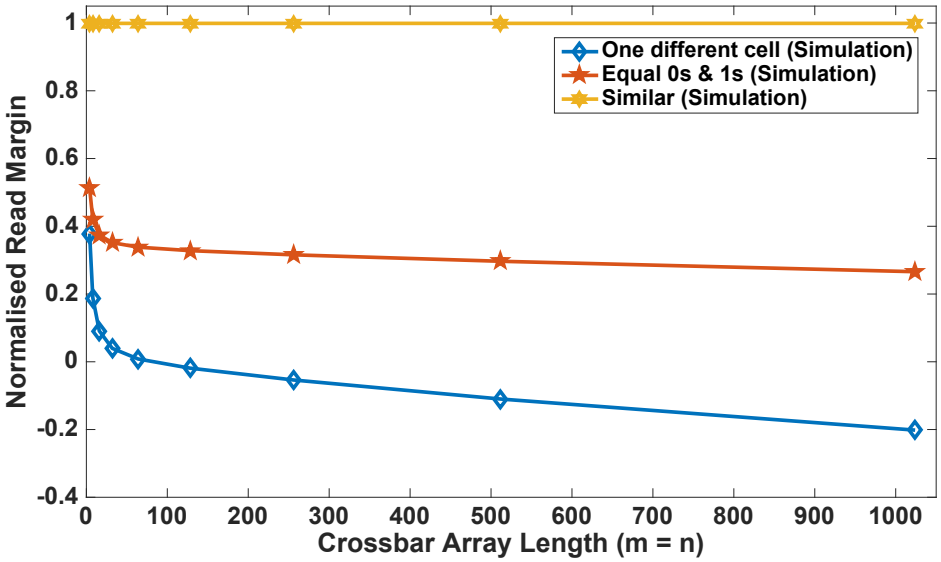


Figure 4.30: Normalised read margin result over larger array. Read margin remains maximum when all the cells in the selected row stores uniform data and at minimum (goes below minimum as the array size increase) when the row is composed principally of low resistance cells.

The average power consumed was computed based on Eqn. 4.61:

$$P_{read} = I_{array} V_{array} \tag{4.61}$$

where I_{array} and V_{array} is the current and voltage across the crossbar array as well as the

load resistances. We simulated various sizes of the memristor based array, at various values of k over a range of array sizes. We read all k cells simultaneously and also read each k cell individually over k cycles. The average power consumed during multiple cells ($k > 1$) read operation is higher compared to reading a single cell at a time. The average power consumption after reading k cells is however much lower in multiple read when compared with the total energy required to read one bit at a time over k cycles. Fig. 4.31 shows the power measurement of reading k cells over a cycle and single bits over k cycles. The power savings when reading multiple cells reduces as k increases from 25% to 100%. At 100% (when all cells in the row are selected), the power consumption of the multiple read out method is approximately zero assuming all cells store the same value. This is because sneak-path is non-existent at this stage.

Also simulated and compared is the power consumptions of the three memristor memory structures described in Section 4.6.3: with the Hamming code based error detection architecture (data and parity bits in different memory blocks then data and parity bits in the same memory block) and the Inverted-Hamming architecture in Fig. 4.28a, 4.28b and 4.28c respectively. We used the worst case scenario of the selected cell(s) storing a 0 and all unselected cells storing a 1. We found the average of the power consumed when the selected row stores a 0 up until when the entire row contains $n-1$ zeros. For instance, in a 4×4 array, the average power consumed when reading all $k = n$ selected cells is the average of the power consumed when the selected row stores 0001, 0011 and 0111 . Fig. 4.32 shows the power consumption trend across the three structures as the array size increases.

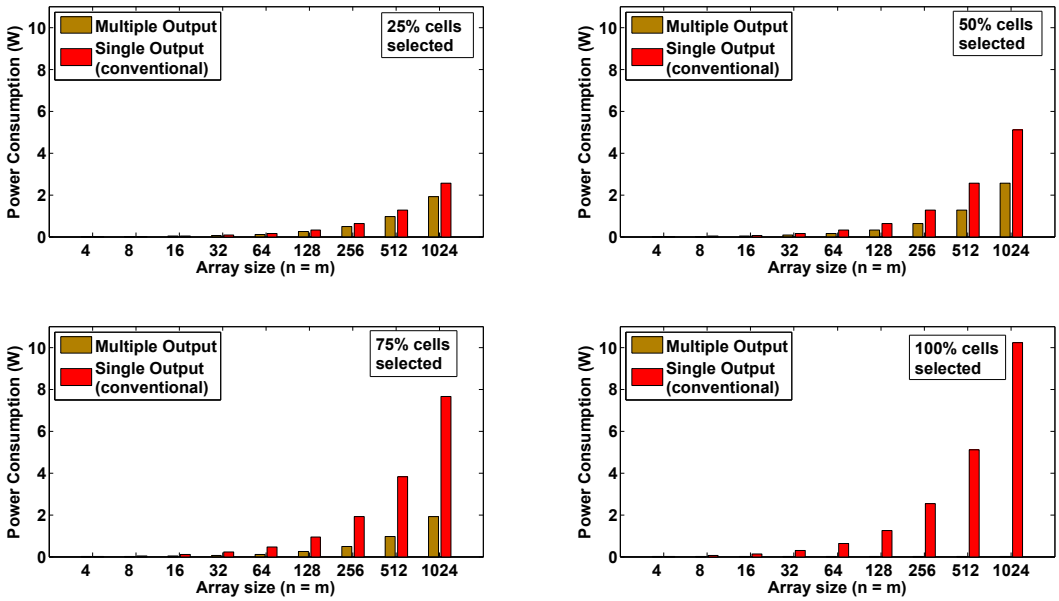


Figure 4.31: Trend of power consumption as the number of selected cells (k) increase over a range of array sizes. The red bars represent the power consumed by the array when reading a single bit over k cycles while the brown bars represent reading multiple cells in a single cycle.

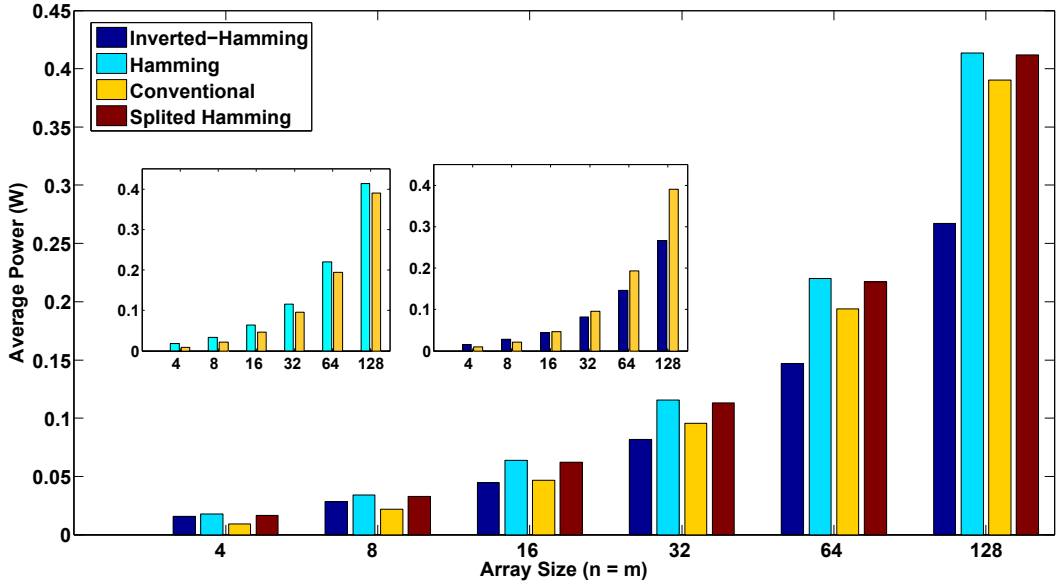


Figure 4.32: Power consumption trend when $k = n$ with non-uniform data as array size increases. Inset on the right is the power consumed in the array using inversion on hamming coding compared against the default memristor array without coding, at smaller array size (n = 4 and 8) inversion on hamming coding consumed more power but reduces drastically as the array size grows. On the left inset is the comparison of the default memory array against an array with Hamming coding.

The inset figure on the left of Fig. 4.32 shows the comparison between the conventional memory structure and a structure with Hamming code. The Hamming code structure consumes more power as the array size increases because of the extra parity bits. Sneak-path leakages are usually more severe when there are more cells with a logic 1 (low resistance) in the selected row than cells with logic 0 (high resistance) in the selected row. The inversion on hamming coding was implemented to neutralise the effect of the added

extra parity bits by inverting the contents of the entire rows if there are more low resistance cells. The figure on the left inset of Fig. 4.32 shows the Inverted-Hamming structure consuming less power than the usual memory structure because of the less number of cells with low resistance value in the selected row.

4.7 Conclusions

A better understanding of the crossbar's read operation is essential in order for the memristor to be effective in other application areas such as gas sensing (Chapter 6). This chapter starts by re-examining the existing modelling of the crossbar architecture. The current model is based on the assumption that all the memristors in a group are in the same resistance state, resulting in a total resistance based on the ratio of the resistance value and the number of devices involved. Since this is not always the case, in reality, each group will usually contain both low and high resistance states. This led to the derivation of model solutions that catered for this real case where each group is free to include both high and low resistance state cells simultaneously. The analytic models for the crossbar array and its read schemes presented in this chapter were verified by simulating the real crossbar array (not equivalent circuit) using the Cadence Spectre simulation tool. A further analysis and modelling of four existing crossbar read schemes was also carried out. Analysis proves that the floating wordline and floating bitline scheme has a failure rate of 75% for small arrays up to 300×300 and a 100% failure rate on larger arrays due to the effect of sneak-path currents. Other schemes that involve grounding of the line(s) have a much lesser fail rate but become highly unreliable at array size greater than 3 orders of magnitude. The grounded schemes also have the disadvantage of consuming enormous power.

This chapter also implemented and analysed the behaviour of the multiple bits read scheme in memristor-based crossbar array. Results shows that the multiple bits read scheme eliminates sneak-paths when all the cells in the row are selected and they store similar data. For a $m \times n$ array, where n cells are selected for reading, the sneak-path effect is eliminated when all n cells store similar logic value. Sneak-path cannot be eliminated when they store different data but can be minimised by selecting more cells for read. This scheme also saves energy when compared with the conventional schemes of read single bit over multiple cycles.

Chapter 5

Improved Techniques for Crossbar Array Write Operation

5.1 Introduction

Having considered the read operation in crossbar architectures, write operation deserves similar considerations because of its reliability issues. Write operation in passive crossbar memory structures could introduce some complexity into the crossbar structure beyond the target cells. Write error in crossbar memories may occur when the voltage reaching the target cell(s) is insufficient to switch it to the desired state and/or the state of unselected cells are perturbed. Write voltage degradation and sneak-path are some of the challenges with crossbar array during the write operation. Reliability of write techniques is critical to the full commercialisation of emerging crossbar memories. This chapter begins with an in-depth performance analysis and analytic modelling of three existing write schemes in Section 5.2: floating lines, $V/2$ and $V/3$ in order to identify their best and worst cases with and without line resistance. These mathematical models will facilitate quick simulation of crossbar write operation with improved accuracy compared to existing EDA tools. The models are then used to develop simulation environments in C++ where required crossbar parameters can be easily modified and circuit simulated faster than with EDA tools. With these models, any $m \times n$ crossbar array can be solved analytically with fewer restrictions. Analysis shows that floating lines scheme could also be considered reliable in arrays with aspect ratio of 1:1 and negligible line resistance just like the latter two schemes. Further analysis also shows that high density crossbar structures cannot be designed using any of the three schemes with worst case line resistance and data distribution within the array.

To ease this problem, a voltage compensating technique is proposed for the write voltage degradation caused by line resistance during write operation in crossbar arrays in Section 5.3. This technique is able to enhance write voltage in the presence of worst case line resistance and thus enable the design of higher density and reliable crossbar arrays. A novel multiple cells write technique is also presented in Section 5.4, multiple cells write helps to conserve power consumption over time as well as a faster write time.

5.2 Analysis of Crossbar Array Write Schemes

Write operation in resistive-based memories suffers from leakage current through undesired paths in the crossbar. In traditional memories, selectors such as diodes and transistors are used to prevent flow of unwanted current in the circuit but this usually introduces extra area overhead. Write operation is carried out on cells in the crossbar array by passing a write voltage across the selected cell while biasing other lines in the crossbar. A write operation can be termed successful if the following conditions are satisfied: a) the selected cell switches to the desired state and b) the state of unselected cells are preserved. For the first condition to hold, the applied write voltage $|V_w|$ must be ideally greater than the SET (RESET) threshold voltages of the selected memristors. This will ensure the cell switches to the desired new state. To preserve the state of unselected cells, the maximum voltage reaching them must be sufficiently less than their threshold voltages. A write failure occurs if one or both conditions are not satisfied. The state of unselected cells can be disturbed if the unselected lines are not biased properly. Aside process variation which may cause fluctuation to the device's threshold voltages, write failure could also be triggered by line resistances in the crossbar arrays. Line resistance weakens the magnitude of the write voltage reaching the selected cell(s) depending on its distance from the voltage driver. Line resistance can be neglected in smaller arrays but becomes more prominent as array size increases. High line resistance prevents the selected cell(s) from receiving sufficient voltage across it in large crossbar arrays. An important figure of merit for write operation is the "write voltage window" which is the difference between the voltage reaching the selected cell and the maximum voltage reaching any of the unselected cells. Line resistance and the choice of biasing scheme are the two major factors that affect the "write voltage window". These factors will be further discussed in subsequent sections.

5.2.1 Crossbar Array Write Operation without Line Resistance

Write operation in crossbar arrays can be modelled without the effect of line resistance for the purpose of architectures with low density or cases where the value of the line resistances are negligible. As earlier mentioned, there are three write schemes commonly used in crossbar arrays - the floating lines, $V/2$ and $V/3$ write schemes. This section presents a novel analysis of the floating lines write schemes' architecture alongside its operation and performances without considering the effect of line resistance. The $V/2$ and $V/3$ write schemes are included in this section for the sake of completion and comparison with the floating lines scheme. Section 5.2.2 addresses the effect of line resistance while using these schemes for crossbar write operations.

5.2.1.1 Floating Lines Write Scheme

In the floating lines write scheme shown in Fig. 5.1, a write voltage of $|V_w|$ and zero are applied to the wordline and the bitline of the target cell respectively while other lines are unbiased. This scheme is the most complicated of the three write schemes and its exact behaviour is often misunderstood. The selected cell in this scheme usually switches successfully, however, the state of the partially-selected cells on the selected wordline (purple) and bitline (green) might not be preserved depending on the structure of the crossbar array.

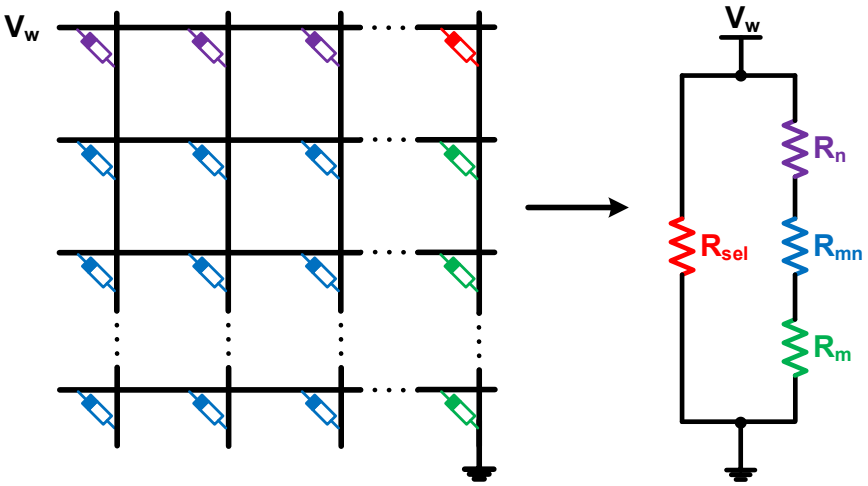


Figure 5.1: Structure of floating lines write scheme and its equivalent circuit level model. Group II cells are in green, Group III cells in purple and Group IV are in blue.

In this scheme, no voltage is applied directly to the unselected wordlines and bitlines. The voltages on the unselected lines, V_{word} and V_{bit} are dependent on the values of m and n

hence $V_{word} = V_{bit} = f(m,n)$. The voltage drop on the unselected groups of cells (R_n, R_m and R_{mn}) can thus be computed by solving the equivalent circuit on the right of Fig. 5.1.

Assuming a fixed V_w is applied to the circuit in Fig. 5.1, a current I_{sneak} represents the sneak-path current flowing through the resistance of the unselected memristors R_m, R_n and R_{mn}. Another current I_{sel} flows through R_{sel} which is the resistance of the selected memristor in the equivalent circuit of Fig. 5.1. V_m, V_n, V_{mn} and V_{sel} below represent the voltages across each resistor respectively:

$$V_m = I_{sneak}.R_m \tag{5.1}$$

$$V_n = I_{sneak}.R_n \tag{5.2}$$

$$V_{mn} = I_{sneak}.R_{mn} \tag{5.3}$$

$$V_{sel} = V_w = I_{sel}.R_{sel} \tag{5.4}$$

In order to compute these voltages, we will need to find the constant current I_{sneak} flowing through R_n, R_{mn} and R_m. If I_w is the current supplied by the voltage driver V_w and I_{sel} is the current flowing through the selected memristor R_{sel}, the following equations can be formed:

$$I_w = I_{sel} + I_{sneak}$$

$$I_{sneak} = I_w - I_{sel}$$

$$\text{Since } I_w = V_w \left(\frac{R_{sel} + R_n + R_{mn} + R_m}{R_{sel}\,(R_n + R_{mn} + R_m)} \right)$$

$$\text{and } I_{sel} = \frac{V_w}{R_{sel}}, I_{sneak} = \frac{V_w}{R_n + R_{mn} + R_m}$$

Substituting the equation of I_{sneak} and those of $R_m = R/(m-1)$, $R_n = R/(n-1)$ and $R_{mn} = R/(n-1)(m-1)$ defined from the basic crossbar modelling in Chapter 2.3 into Eqn. 5.1, 5.2 and 5.3, V_m, V_n and V_{mn} can be solved as below

$$V_m = \frac{V_w}{R_n + R_{mn} + R_m} . \frac{R}{m-1}$$

$$\text{Since } R_n + R_{mn} + R_m = \frac{R(m+n-1)}{(m-1)(n-1)}$$

$$V_m = \frac{V_w(m-1)(n-1)}{R(m+n-1)} \cdot \frac{R}{m-1} \tag{5.5}$$

$$V_m = \frac{V_w(n-1)}{m+n-1} \tag{5.6}$$

V_n and V_{mn} can be derived in similar way:

$$V_n = \frac{V_w(m-1)}{m+n-1} \tag{5.7}$$

$$V_{mn} = \frac{V_w}{m+n-1} \tag{5.8}$$

Based on the solutions of V_m, V_n, V_{mn} and V_{sel}, we can properly explore the behaviour of the floating lines scheme under different array structures. Fig. 5.2 shows the simulation results of write operation using this scheme over a range of square array sizes. The selected cell has a desired voltage drop of V_w while the voltage drop across other cells does not exceed $V_w/2$. It is also worthy of mention that a voltage of $V_w/2$ can cause a partial change of memristance value if applied for too long [37]. This analysis assumes that the voltage is only applied for the duration required to switch the selected cell to the desired state.

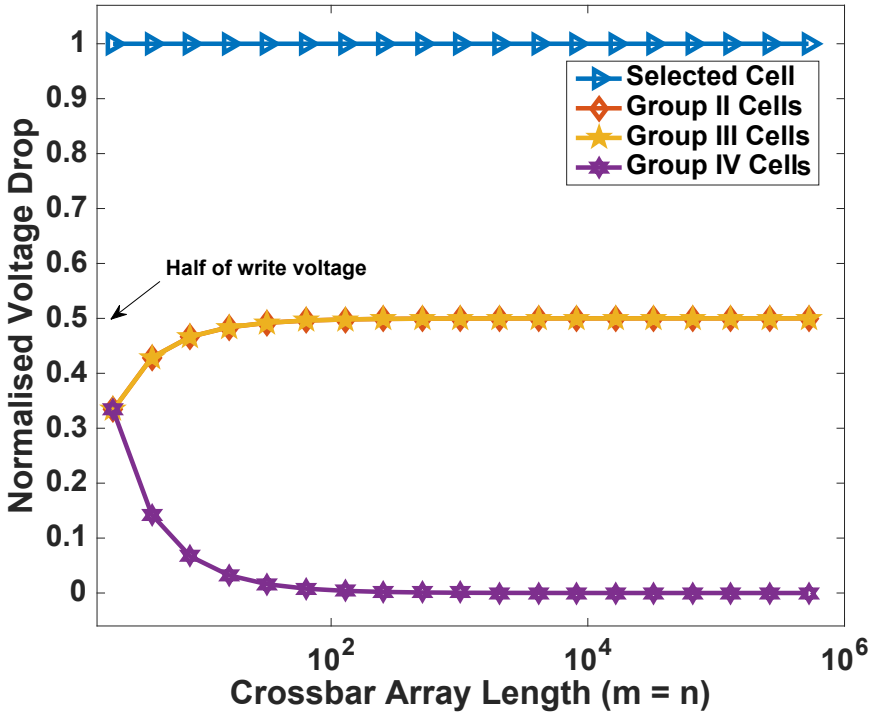


Figure 5.2: Simulation result of the floating lines write scheme. The voltage drop on the unselected cells are guaranteed not to exceed $V_w/2$ irrespective of the array size if $m = n$.

Fig. 5.3 shows a simulation result of voltage drop on the three groups of unselected cells for different array sizes including cases of $m \neq n$, the worst case write disturbance in

the floating scheme occurs in arrays with non-square aspect ratio ($m \neq n$). Partially-selected cells in Group II (green) and Group III (purple) have an undesired voltage of approximately V_w across them when $n \gg m$ and $m \gg n$ respectively. The disturbance impact increases as the gap between m and n widens. However, Group IV (blue) cells receive less voltage as the array size increases irrespective of the array shape. In summary, the success of the floating scheme depends heavily on the size and aspect ratio of the crossbar array. It is also worthy of mention that the floating scheme cannot be used for multiple cells write as will be discussed in Section 5.4. Table 5.1 shows a summary of voltage drop and current across each group of cells in the crossbar array during write operation with floating lines scheme. V_{sel_word} and V_{sel_bit} are the voltages applied to the selected wordline and bitline respectively. V_{word} and V_{bit} are the voltages applied to unselected wordlines and bitlines respectively.

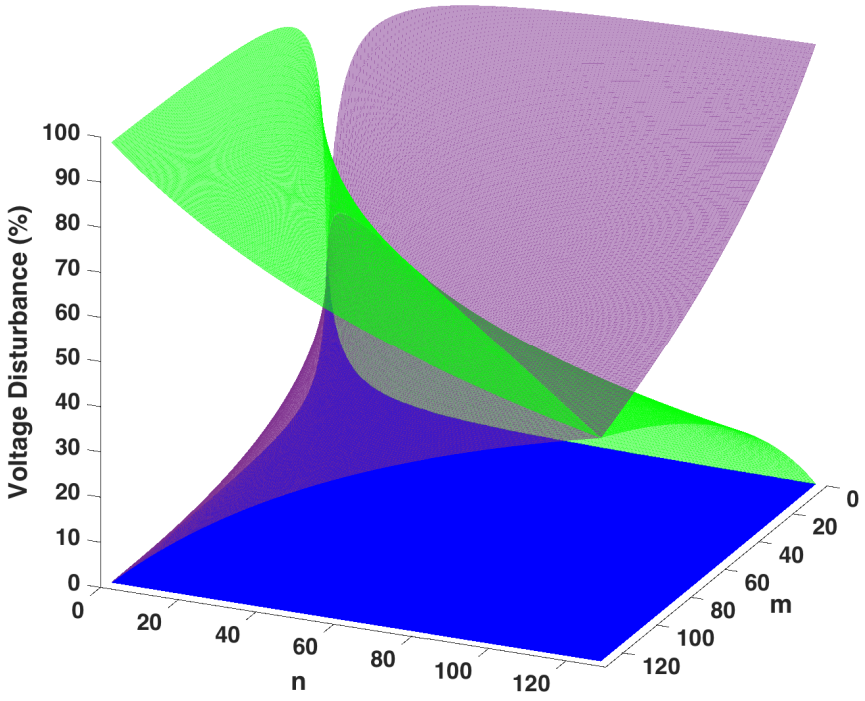


Figure 5.3: Voltage drop on unselected cells as array size varies when unselected lines are left floating. Voltage on partially-selected cells are depicted in purple and green, while unselected cells are represented in blue. Partially-selected cells might reach up to V_w in non-square array structures but guaranteed not to exceed $V_w/2$ only if $m = n$.

In the floating lines scheme, maximum voltage to the unselected cells depends on the aspect ratio of the array. Voltage disturbance in this scheme does not exceed $V_w/2$ when the aspect ratio is 1:1 ($m = n$) irrespective of the array size. This invariably implies that the write operation will always succeed with this scheme when the crossbar array is square shaped. On the other hand, a partially-selected cell could have a voltage of almost V_w

across it in an array where $m \neq n$; a definite write failure.

Table 5.1: Voltage drop across the various groups of cells using the floating lines write scheme assuming all the cells have the same resistance R.

	Voltage Drop Formula	Voltage drop	Current
Selected cell(s)	$V_{sel\_word} - V_{sel\_bit}$	$V_w - 0 = V_w$	V_w/R
Group II cell(s)	$V_{word} - V_{sel\_bit}$	$f(m,n) - 0 = \frac{V_w(m-1)}{m+n-1}$	$\frac{V_w(m-1)(n-1)}{R(m+n-1)}$
Group III cell(s)	$V_{sel\_word} - V_{bit}$	$V_w - f(m,n) = \frac{V_w(n-1)}{m+n-1}$	$\frac{V_w(m-1)(n-1)}{R(m+n-1)}$
Group IV cell(s)	$V_{word} - V_{bit}$	$f(m,n) - f(m,n) = \frac{V_w}{m+n-1}$	$\frac{V_w(m-1)(n-1)}{R(m+n-1)}$

5.2.1.2 V/2 Write Scheme

The $V/2$ write scheme is well understood and commonly used as the choice scheme for write operation in crossbar arrays [70]. The structure of the $V/2$ scheme is shown in Fig. 5.4. A voltage of V_w and zero are applied to the wordline and the bitline of the target cell respectively and all other lines are biased with a voltage of $V_w/2$. The voltage bias helps to minimise current leakages to some of the unselected cells. In this scheme, the majority of the unselected cells are totally protected against disturbance. To be specific, $((m-1)(n-1))$ cells are protected, these are the cells in Group III (blue). The remaining $m+n-2$ cells in Group I (green) and II (purple) in the $m \times n$ array are exposed to a voltage drop of $V_w/2$, which will ideally keep the cells safe. The number of cells $(m+n-2)$ susceptible to voltage disturbance in this scheme is less than 50% of the total cells in the array (mn) (percentage reduces as the array size grows). However, the probability of write error is slightly high in the $V/2$ scheme because the partially-selected cells $(m+n-2)$ have a voltage of $V_w/2$ across them constantly. Their state could be perturbed if the voltage is applied long enough [37]. Table 5.2 shows a summary of voltage drop and current across each group of cells in the crossbar array during write operation with the $V/2$ scheme.

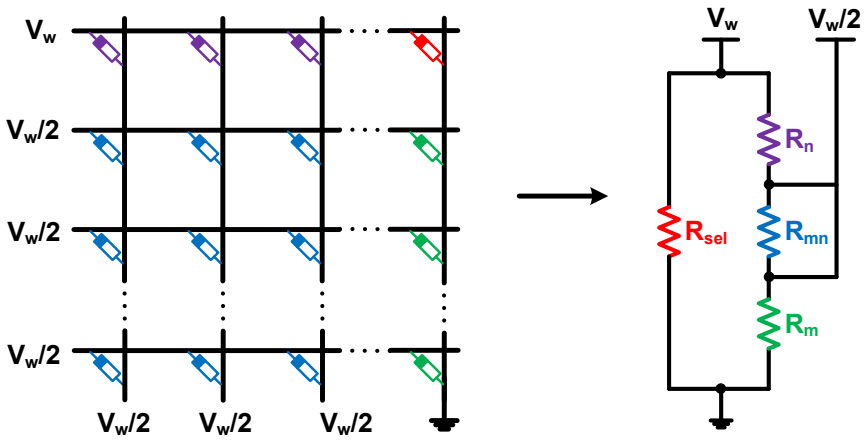


Figure 5.4: Structure of $V/2$ write scheme and its equivalent circuit level model. The selected cell in red has a voltage of V_w across it while the purple and green partially-selected cells have a voltage drop of $V_w/2$ on them. The unselected blue cells have zero voltage across them.

Table 5.2: Voltage drop across the various groups of cells using the $V/2$ write scheme assuming all the cells have the same resistance R.

	Voltage Drop Formula	Voltage drop	Current
Selected cell(s)	$V_{sel\_word} - V_{sel\_bit}$	$V_w - 0 = V_w$	V_w/R
Group II cell(s)	$V_{word} - V_{sel\_bit}$	$V_w/2 - 0 = V_w/2$	$V_w/2R$
Group III cell(s)	$V_{sel\_word} - V_{bit}$	$V_w - V_w/2 = V_w/2$	$V_w/2R$
Group IV cell(s)	$V_{word} - V_{bit}$	$V_w/2 - V_w/2 = 0$	≈ 0

5.2.1.3 V/3 Write Scheme

In the $V/3$ scheme shown in Fig. 5.5, the lines of the target cell are biased with the usual V_w and zero voltages but the other cells have a voltage of $V_w/3$ and $2V_w/3$ applied to their wordlines and bitlines respectively [70]. The $V/3$ scheme offers the best protection against voltage disturbance to unselected cells. However, all the groups of unselected cells shown in Fig. 5.5 are affected by a voltage of $V/3$ which is less than their threshold voltages. The maximum amount of voltage applied to any of the unselected cell is reduced to $V_w/3$ in this scheme as against $V_w/2$ in the floating lines and the $V/2$ scheme. Although more cells $(mn-1)$ are affected by leakage in the $V/3$ scheme but the probability of their state being disturbed is low. Each of the unselected cells in the $V/3$ scheme has a current of $V_w/3R$ sneaking through them. Table 5.3 shows a summary of voltage drop and

current across each group of cells in the crossbar array during write operation with the $V/3$ scheme.

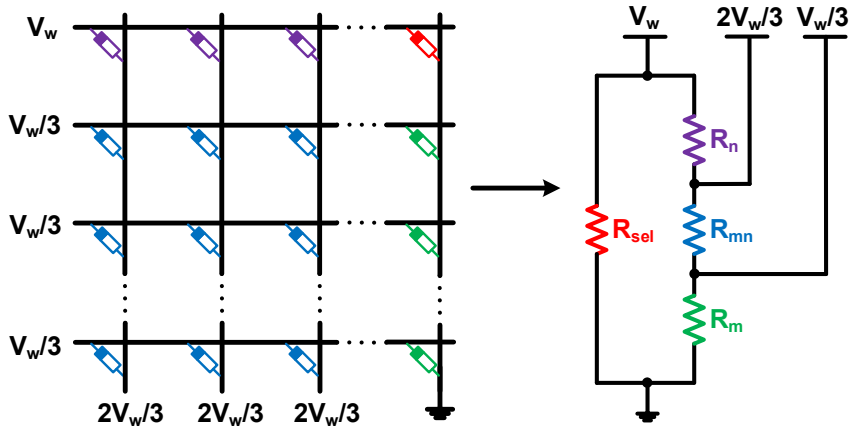


Figure 5.5: Structure of $V/3$ write scheme and its equivalent circuit level model. The selected cell in red has a voltage of V_w across it while the other cells (purple, green and blue) has a voltage drop of $V_w/3$.

Table 5.3: Voltage drop across the various groups of cells using the $V/3$ write scheme assuming all the cells have the same resistance R.

	Voltage Drop Formula	Voltage drop	Current
Selected cell(s)	$V_{sel\_word} - V_{sel\_bit}$	$V_w - 0 = V_w$	V_w/R
Group II cell(s)	$V_{word} - V_{sel\_bit}$	$V_w/3 - 0 = V_w/3$	$V_w/3R$
Group III cell(s)	$V_{sel\_word} - V_{bit}$	$V_w - 2V_w/3 = V_w/3$	$V_w/3R$
Group IV cell(s)	$V_{word} - V_{bit}$	$V_w/3 - 2V_w/3 = V_w/3$	$V_w/3R$

5.2.1.4 Experimental Results and Discussions

In summary, all the three aforementioned write schemes have their pros and cons without the effect of line resistance. The floating scheme is only considered reliable when the array is square shaped; otherwise state of unselected cells are perturbed. The $V/2$ scheme protects more cells than any other scheme but the state of the $m+n-2$ cells constantly exposed to a voltage of $V_w/2$ can be overwritten if the write voltage is applied for a longer time. The $V/3$ scheme on the other hand, exposes all the groups of unselected cells to a safe low voltage of $V_w/3$. Analysis of these schemes will be incomplete without considering their power consumptions.

The average power consumption of the floating lines scheme can be accurately computed by summing up the individual power consumption of the four groups of cells in Fig. 5.1 as in:

$$P_{float} = (V_w \times I_{sel}) + (V_n \times I_{sneak}) + (V_{mn} \times I_{sneak}) + (V_m \times I_{sneak}) \tag{5.9}$$

The power consumption of the $V/2$ and $V/3$ can be calculated using Eqn. 5.10 and 5.11 respectively:

$$P_{V/2} = (V_w \times I_{sel}) + (V_w/2 \times I_{sneak}) \tag{5.10}$$

$$P_{V/3} = (V_w \times I_{sel}) + (V_w/3 \times I_{sneak}) \tag{5.11}$$

We can also show that the power consumptions of the floating line and $V_w/2$ scheme is approximately similar: If we consider $n \approx m$:

$$I_{float\_sneak} = \frac{V_w}{R}\frac{(n-1)^2}{2n-1} \tag{5.12}$$

$$P_{float\_sneak} = V_w \times I_{float\_sneak} \tag{5.13}$$

$$P_{float\_sneak} = \frac{{V_w}^2}{R}\frac{(n-1)^2}{2n-1} \tag{5.14}$$

If $V_w/2$ is considered in similar way:

$$I_{V/2\_sneak} = \frac{V_w}{2R}(n-1) + \frac{V_w}{2R}(n-1) \tag{5.15}$$

$$I_{V/2\_sneak} = \frac{V_w}{R}(n-1) \tag{5.16}$$

$$P_{V/2\_sneak} = \frac{V_w}{2} \times I_{V/2\_sneak} \tag{5.17}$$

$$P_{V/2\_sneak} = \frac{V_w}{2} \times \frac{V_w}{R}(n-1) \tag{5.18}$$

$$P_{V/2\_sneak} = \frac{{V_w}^2}{R}\frac{n-1}{2} \tag{5.19}$$

for large n:

$$P_{float\_sneak} = \frac{{V_w}^2}{R}\frac{n-1}{2} \tag{5.20}$$

Therefore: (5.21)

$$P_{float\_sneak} \approx P_{V/2\_sneak} \tag{5.22}$$

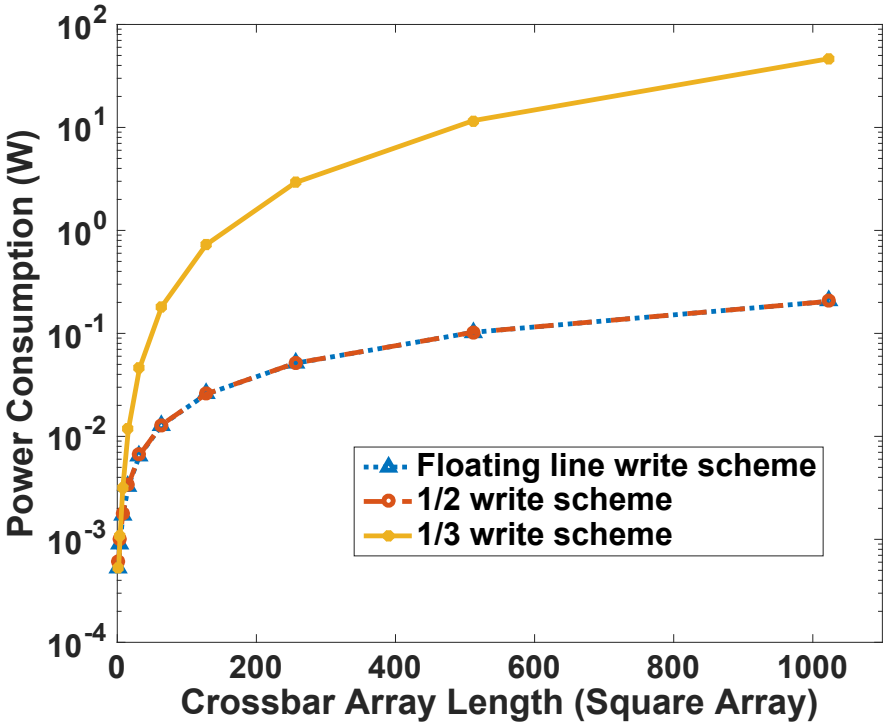
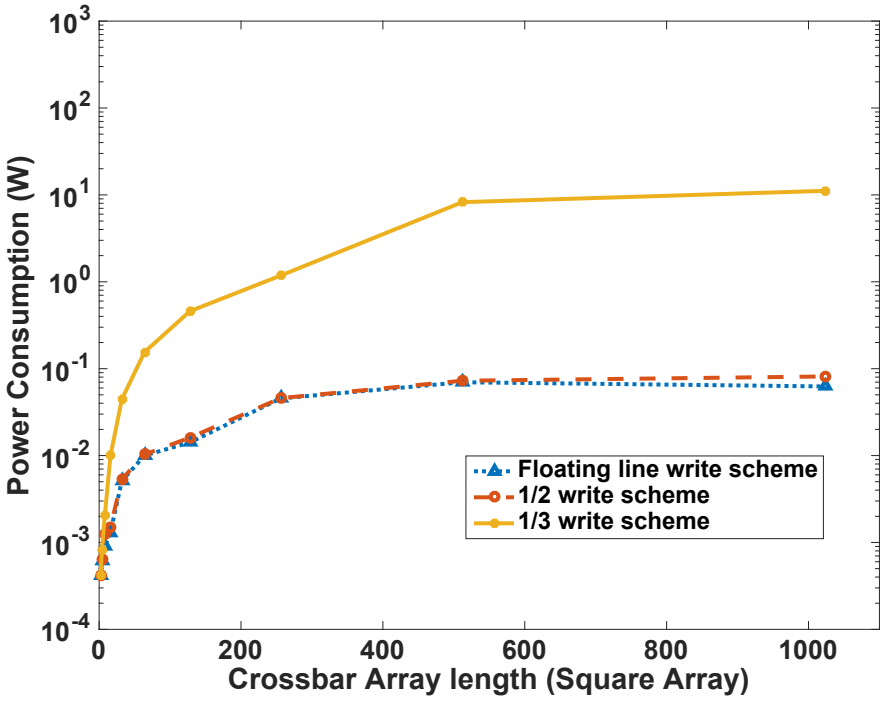
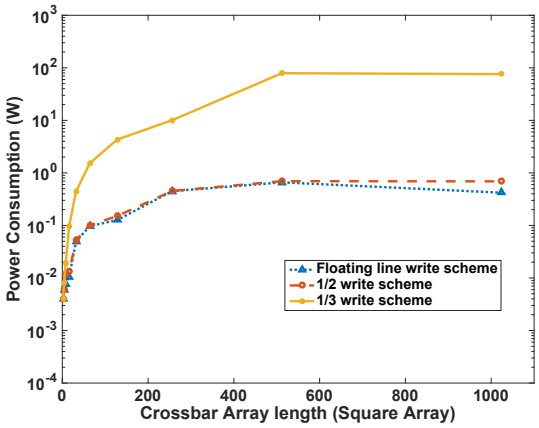


Figure 5.6: Comparison of power consumptions in the three write schemes in the worst case scenario (all unselected cells are initialised to R_{on}). $R_{on} = 10k\Omega$, $R_{off} = 100K\Omega$, $R_{off}/R_{on} = 10^1$. Power value is scaled up equally for the three schemes in order to show the difference between the schemes.

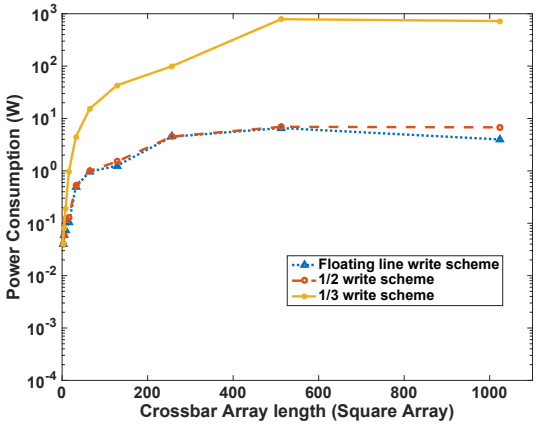
Fig. 5.6 shows an ideal power consumption comparison between the three schemes. The $V/3$ scheme which is the most reliable write scheme consumes enormous power. The huge power consumption of the $V/3$ scheme is as a result of frequent switching between the $V_w/3$ and $2V_w/3$ power rail as well as the fact that all the cells in the array experience some degree of sneak-path current. Fig. 5.7 shows the power consumption comparison between the three scheme in a less severe situation, here the crossbar array is designed such that the number of R_{on} and R_{off} cells are random. This is different from the simulation of Fig. 5.6 where the crossbar is written in the worst case situation - all unselected cells are set to R_{on}.



(a)



(b)



(c)

Figure 5.7: Comparison of power consumptions in the three write schemes with random array resistance pattern, this is a less severe case to the worst case scenario presented in Fig. 5.6. R_{off}/R_{on} ratio set to: (a) 10^1 (b) 10^2 (c) 10^3. Power value is scaled up equally for the three schemes in order to show the difference between the schemes.

Fig. 5.8 shows the power consumptions of the floating lines scheme compared against the $V/2$ write scheme so as to show cases where the dimensions of the array are non-square ($m \neq n$). As m deviates from n, there is more power savings with the floating lines than the $V/2$ scheme but the inequality between m and n causes further disturbance to the unselected cells which could lead to write failure as explained in Section 5.2.1.1. Table 5.4 shows a performance summary of the three schemes with their corresponding analytical models for their performance metrics.

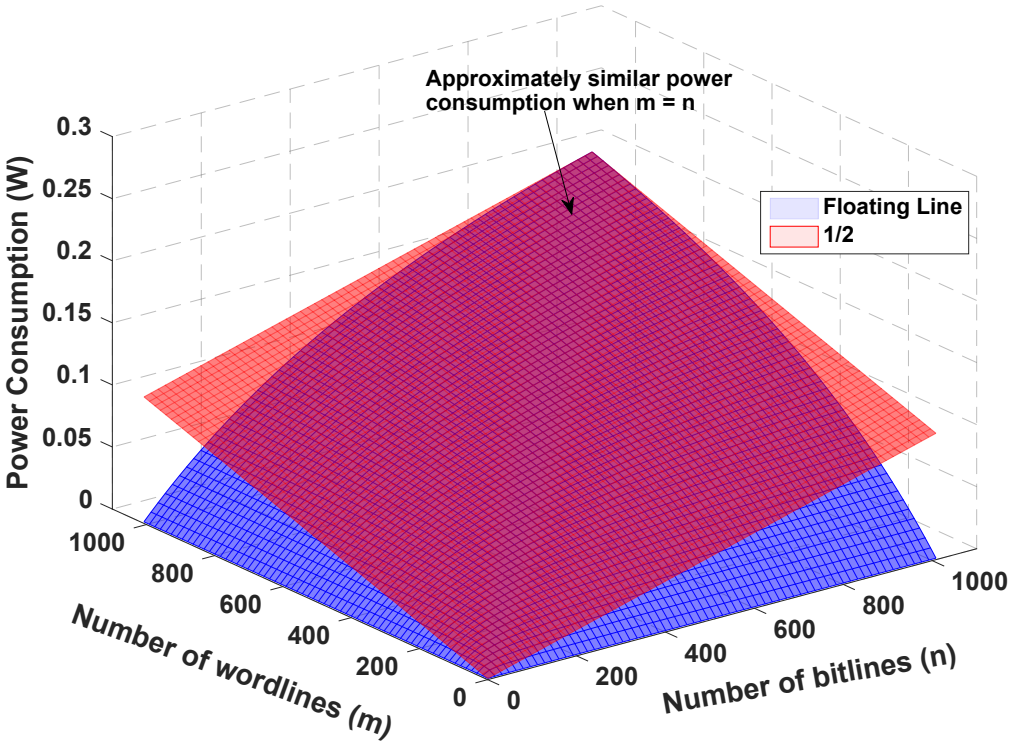


Figure 5.8: Comparison of power consumptions between the floating lines and $V/2$ write schemes over varying range of array sizes and structures. Power value is scaled up equally for the two schemes in order to show the difference between the schemes.

Table 5.4: Comparative summary of write schemes' performances without the effect of line resistance

	Voltage on selected cell(s)	Voltage on Unselected Cells		Total Current Leakage	No. of disturbed cells	Dependence on Array Size/ Aspect Ratio	Probability of unselected cells Switching
		Min. Voltage	Max. Voltage				
Floating Lines	V_w	$\frac{V_w}{m+n-1}$	$\max\left(\frac{V_w(m-1)}{m+n-1}, \frac{V_w(n-1)}{m+n-1}\right)$	$\frac{V_w(m-1)(n-1)}{R(m+n-1)}$	$mn-1$	Yes	≤ 0.5 (if $m=n$)
V/2	V_w	0	$V_w/2$	$\frac{V_w(m+n-2)}{2R}$	$m+n-2$	No	0.5
V/3	V_w	$V_w/3$	$V_w/3$	$\frac{V_w(mn-1)}{3R}$	$mn-1$	No	0.33

5.2.2 Crossbar Array Write Operation with Line Resistance

In smaller arrays, line resistance could be negligible but as the array size increases and integration complexity grows, the effects of line resistance becomes more prominent. Therefore, it is important to factor in line resistances for an accurate analysis of write operation in crossbar architectures. Fig. 5.9 shows a resistance model of a $m \times n$ crossbar array. Each memristor in the array has a wire resistance adjacent to either side of its bitline and wordline. In the presence of line resistance, the voltage reaching the farthest

cell from the voltage source might not be sufficient to switch the cell to the desired state, thereby leading to a write error. Similarly, unselected cells closer to the voltage source could be written in error as a result of voltage deflected to/from them. Current analyses of crossbar write schemes in the presence of line resistance are limited and difficult to adapt into simulation tools.

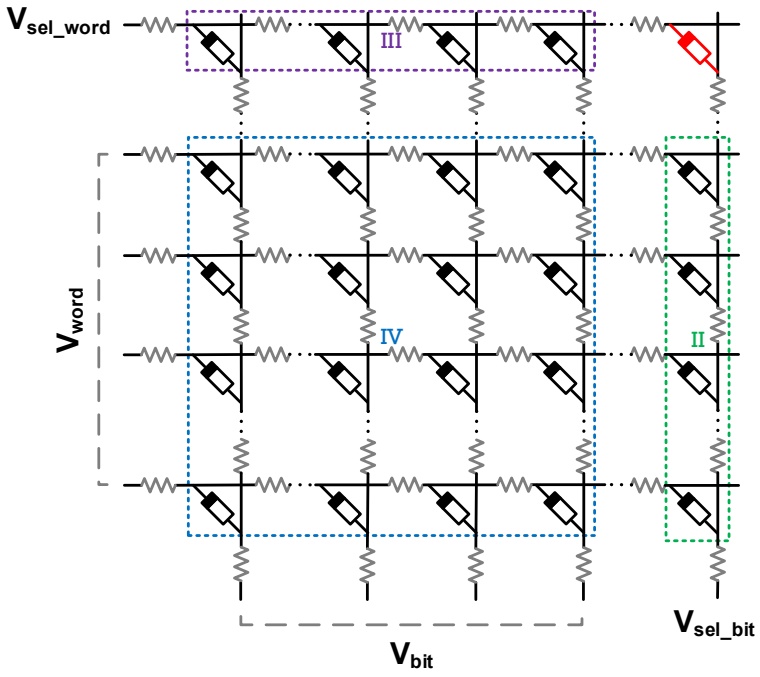


Figure 5.9: Resistance model of the crossbar array showing resistance of nanowires.

There is no single closed form formula that can accurately describe the voltage drop on each of the cells in the crossbar array of the resistance model depicted by Fig. 5.9. In order to determine the voltage drop on any memristor of resistance $R^m_{i,j}$ in any $m \times n$ array, where $1 \leq i \leq m$ and $1 \leq j \leq n$, the voltage on its wordline (row) and bitline (column) denoted by $V^w_{i,j}$ and $V^b_{i,j}$ must be solved. For the entire array, there are $2 \times m \times n$ unknown voltages as each memristor has two unknown voltages. A set of $2mn$ linear equations are thus required in order to determine the voltage drop on the memristors [72]. The current flowing through each memristor can be described by two Kirchhoff's current law equations - each representing the current flowing through the wordline and bitline. Eqn. 5.23 shows the three possible wordline equations of each memristor depending on its location on the wordline (first column ($j = 1$) or last column ($j = n$) or mid-column ($1 < j < n$)). Current flowing through the bitline of a memristor will be described by one of Eqn. 5.24 depending on the cell's location on the bitline (first row ($i = 1$) or last row ($i = m$) or mid-row ($1 < i < m$))

Kirchhoff's Equations formed based on the voltages on the wordline junction of each

memristor:

$$\frac{V^b_{i,j} - V^w_{i,j}}{R^m_{i,j}} = \begin{cases} \frac{V^w_{i,j} - V^w_i}{R_{wi}} + \frac{V^w_{i,j} - V^w_{i,j+1}}{R_{wi,j}}, & \text{if } j = 1 \\ \frac{V^w_{i,j} - V^w_{i,j-1}}{R_{wi,j-1}} + \frac{V^w_{i,j} - V^w_i}{R_{wi}}, & \text{if } j = n \\ \frac{V^w_{i,j} - V^b_{i,j+1}}{R_{wi,j}} + \frac{V^w_{i,j} - V^w_{i,j-1}}{R_{wi,j-1}}, & \text{otherwise} \end{cases} \tag{5.23}$$

Kirchhoff's Equations formed based on the voltages on the bitline junction of each memristor:

$$\frac{V^w_{i,j} - V^b_{i,j}}{R^m_{i,j}} = \begin{cases} \frac{V^b_{i,j} - V^b_j}{R_{bj}} + \frac{V^b_{i,j} - V^b_{i+1,j}}{R_{bi,j}}, & \text{if } i = 1 \\ \frac{V^b_{i,j} - V^b_{i-1,j}}{R_{bi-1,j}} + \frac{V^b_{i,j} - V^b_j}{R_{bj}}, & \text{if } i = m \\ \frac{V^b_{i,j} - V^w_{i+1,j}}{R_{bi,j}} + \frac{V^b_{i,j} - V^b_{i-1,j}}{R_{bi-1,j}}, & \text{otherwise} \end{cases} \tag{5.24}$$

Here, $V^w_i = V_{sel\_word}$ and $V^b_j = V_{sel\_bit}$ are the write voltage values on the selected wordline i and bitline j respectively. $V^w_{i,j}$ and $V^b_{i,j}$ are the voltage values on the wordline and bitline of the selected memristor respectively. R_{wi} and R_{bj} are the line resistance values at source of the voltage to wordline i and bitline j respectively. $R_{wi,j}$ and $R_{bi,j}$ are the line resistance values at the wordline i and bitline j respectively.

With line resistance, the worst case write scenario occurs when all the cells are in LRS (R_{on}) and the selected cell is farthest away from the voltage source. In addition to the weakening of the voltage drop on the selected cell, maximum disturbance to unselected memristors also increases as array size grows in all the three schemes described in Section 5.2.1. Fig. 5.10 shows the simulation results of the three schemes in the presence of different line resistance values across an increasing array size. The line resistance values chosen are similar to those in [105]. For this simulation, all cells in the array are kept at a worst case value of R_{on}. Maximum disturbance to the unselected cells using both floating lines and $V/2$ schemes are kept at a maximum of $V_w/2$ as array size increases. Unlike other schemes, the $V/3$ write scheme causes voltage of up to $2V_w/3$ to reach the unselected cells nearest to the bitline and farthest from the wordline as shown in the simulation results. Crossbar array density could be highly limited depending on the resistivity of nanowire used in the crossbar design. A worst case R_L/R_{on} value of 10^{-2} leads to a negative write voltage window (difference between voltage reaching selected device and maximum voltage to unselected cell) in array with over 256 devices across all the three

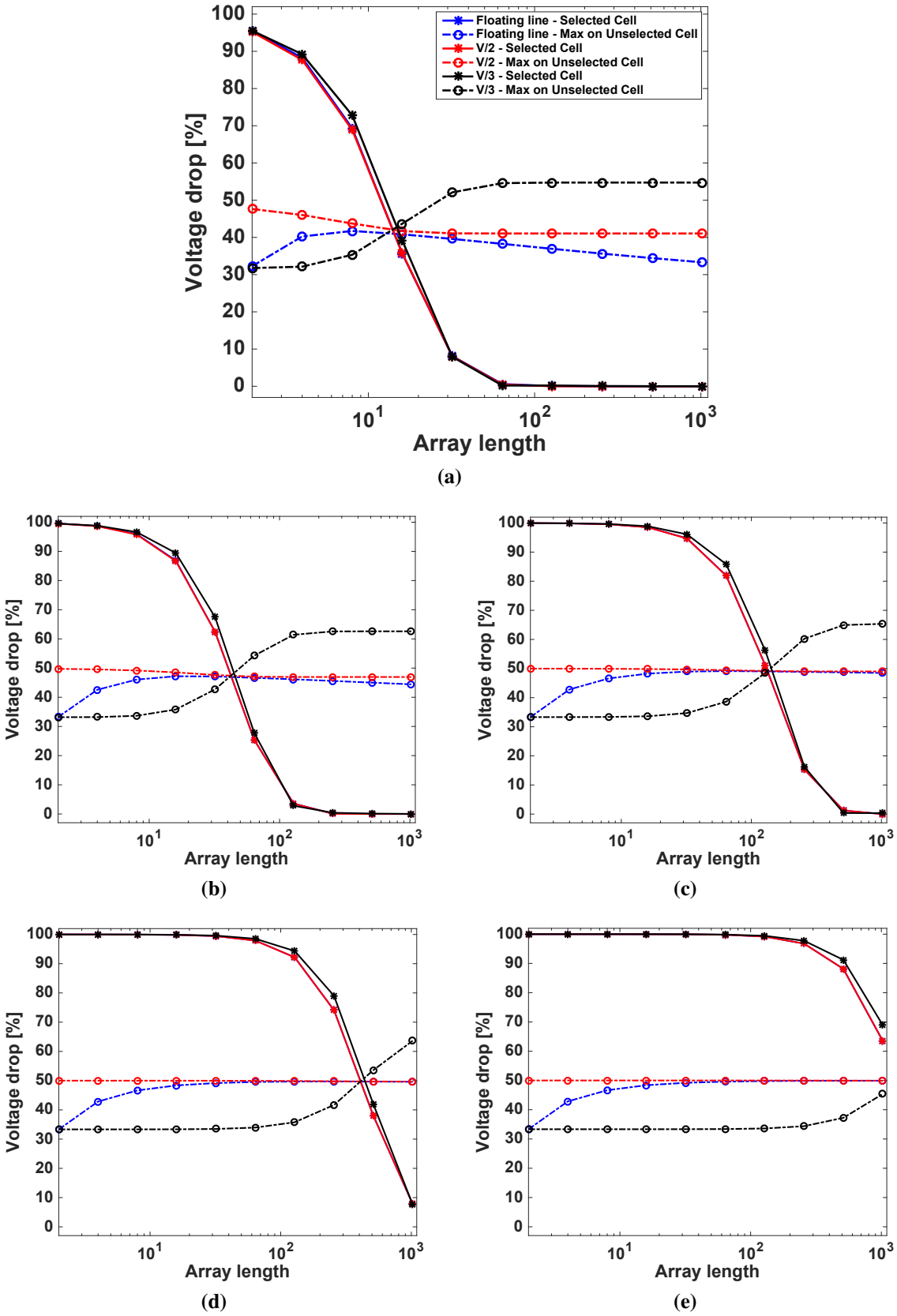


Figure 5.10: Voltage drop across selected and unselected cells with all memristor in the crossbar set to R_{on} and R_L/R_{on} set to: (a) 10^{-2} (b) 10^{-3} (c) 10^{-4} (d) 10^{-5} (e) 10^{-6}.

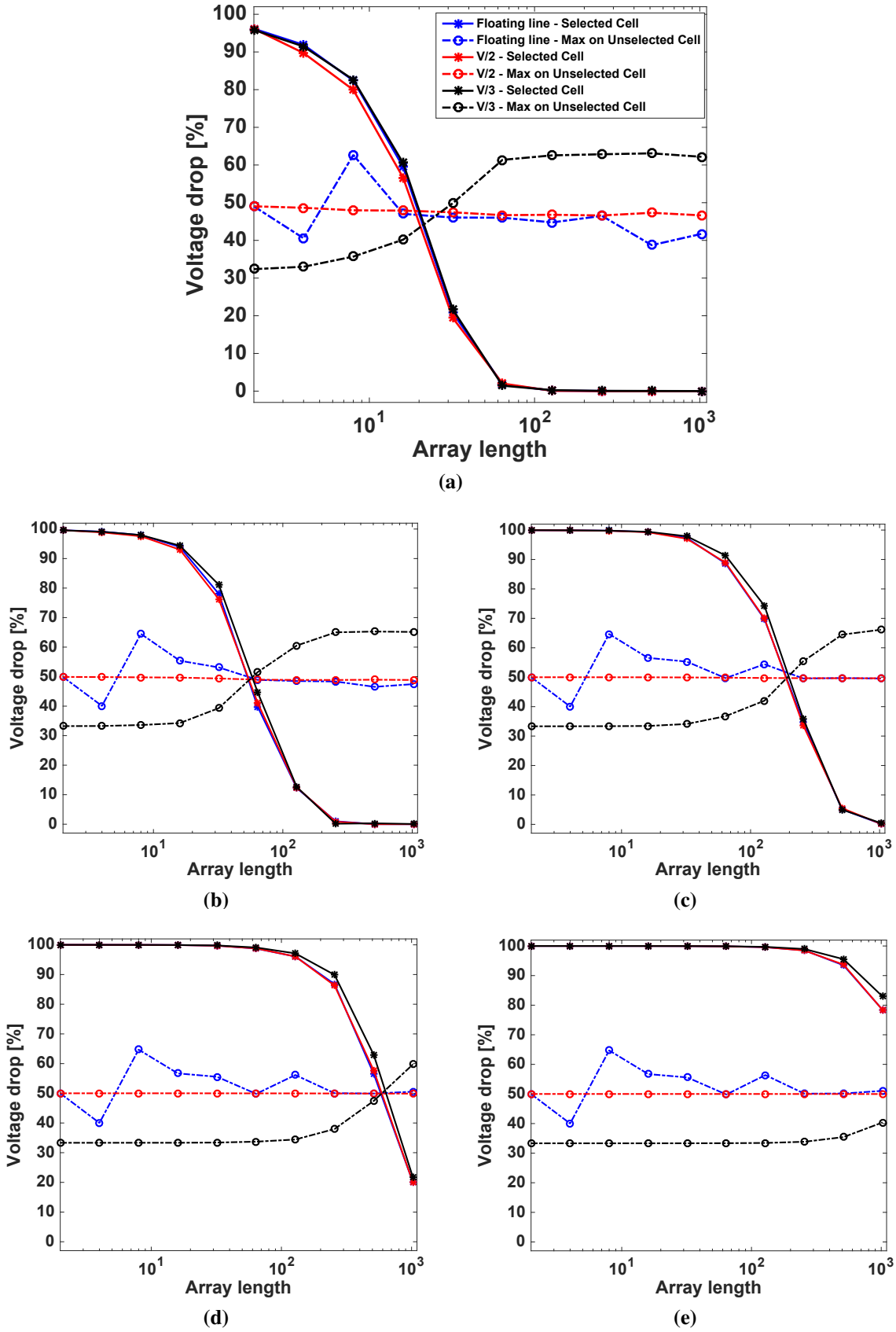


Figure 5.11: Voltage drop across selected and unselected cells with random resistance pattern in the crossbar with R_L/R_{on} set to: (a) 10^{-2} (b) 10^{-3} (c) 10^{-4} (d) 10^{-5} (e) 10^{-6}.

schemes (see Fig. 5.11(a)). The memory is not useful with a write technique and worst case nanowire resistivity that leads to a negative write window. High density and reliable crossbar memory can be realised by keeping line resistance to the barest minimum. Result with a generous line resistance of $R_L/R_{on} = 10^{-6}$ is quite promising as shown in Fig. 5.11(e) even with a worst case data pattern where all memristors are set to R_{on}. Fig. 5.11 shows similar simulation results as in Fig. 5.10 but with a random resistance pattern. The result shows an improved write voltage window as the voltage drop across the selected cell degrades at a slower rate as the array size increases.

One approach to solving this problem is the double-sided ground biasing (DSGB) approach where both sides of the selected wordline array are grounded and the selected bitline is connected to the write voltage [73]. Another solution uses *dual voltage source* design where voltage are delivered via both sides of the selected wordlines [71]. Both described techniques however incur additional chip area overhead thereby reducing the array efficiency without offering much in terms of preventing voltage degradation.

5.3 Compensated Write Voltage Technique

In this section, a novel write technique that is able to compensate for the voltage loss during the write operation on memristor-based crossbar using the $V/2$ write scheme is proposed. The $V/2$ scheme was chosen primarily because of the balanced voltage levels it offer on the wordlines and bitlines. Reliability can be enforced by ensuring voltage is only applied for a short duration as discussed in Section 5.2. Existing solutions such as the 'double-sided ground biasing' and 'dual voltage source' incurs additional chip area overhead without offering much in terms of preventing voltage degradation. The technique proposed here can also be extended to crossbar arrays with crosspoint cells other than memristors.

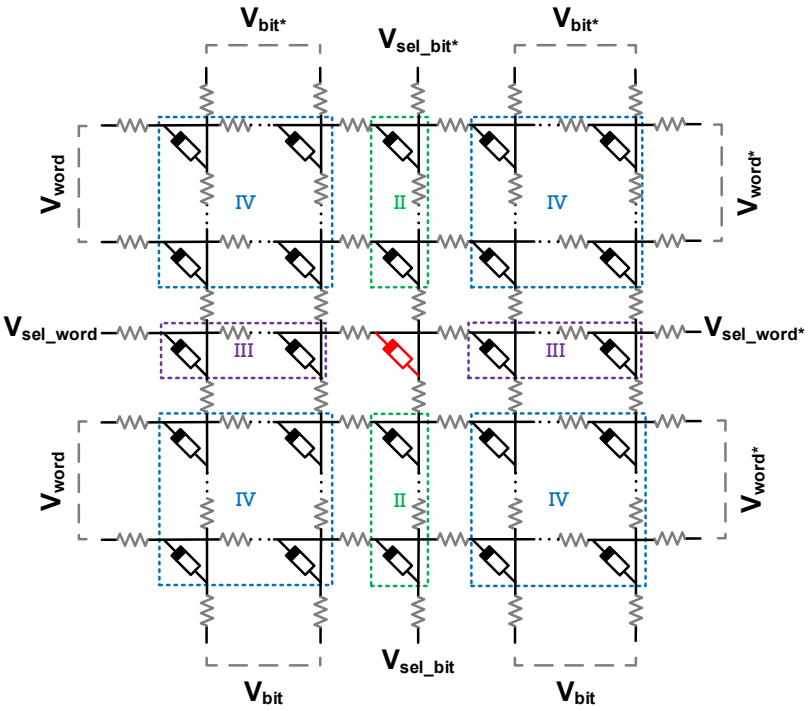


Figure 5.12: Resistance model of the crossbar array using the dual voltage source technique. The starred voltages will have the same value as their unstarred counterpart.

As mentioned earlier, the value of the line resistance, data distribution in the array and choice of write scheme are the three major parameters that causes voltage loss during write operation. The compensated voltage technique was used alongside the dual voltage source technique. The schematic is depicted in Fig. 5.12. In the 'dual voltage source' technique, the worst case selected cell will move to the middle of the array [71]. The starred voltage sources will have the same magnitude as their unstarred counterparts. The starred voltage sources are optional and are used to strengthen the voltage reaching cells further away from the main voltage driver. The compensated technique was implemented by adjusting the value of the voltages applied to the unselected wordlines and the selected bitline. Usually, voltage drop on the selected cell is determined by V_{sel_word} and V_{sel_bit} as explained in previous sections. In this proposed technique, as opposed to the usual grounding of V_{sel_bit}, a negative (positive) voltage is applied instead to supplement the positive (negative) write voltage applied at V_{sel_word}, thereby increasing the overall voltage drop on the selected cell or compensating for voltage degradation due to line resistance effects. Applying a voltage source to V_{sel_bit} also leads to an increase in the voltage drop on cells in Group II. The effect of this modification can be balanced by reducing the voltage applied to V_{word} such that the summation of voltages applied to both V_{word} and V_{sel_bit} still results in $V_w/2$ therefore ensuring cells in Group II are kept safe from disturbances. Group IV cells experience an increase in voltage drop because of the change to V_{word}

but the voltage drop is guaranteed to be below $V_w/2$ depending on the sharing ratio of V_{word} and V_{sel_bit}. Cells in Group III are not affected by these modifications. In order to keep the voltage drop on Group II cells at or below $V_w/2$, a percentage of V_{word}'s voltage ($V_w/2$) is extracted and applied to V_{sel_bit}. Table 5.5 shows a summary of the new voltage drop on the various groups of cells in the new compensated and dual voltage techniques. The selected cells can only benefit by this compensation as it is tolerable for the selected cell(s) to have a voltage drop in excess of its threshold voltage.

We simulated three sharing ratios between V_{word} and V_{sel_bit} (K:1-K) namely 0.8:0.2, 0.5:0.5 and 0.2:0.8 over a range of R_L/R_{on} values. We have used a switching voltage requirement of a minimum of $75\%V_w$ for the selected cell(s). Simulation results depicted in Fig. 5.13(a) - 5.13(d) show over 3× improvements in the voltage reaching the worst case selected cells with the 0.2:0.8 compensation technique compared to the conventional method. This technique simply helps to increase the voltage delivered to the worst case selected cell(s) without endangering the unselected cells beyond their preset thresholds.

Table 5.5: Voltage drop across the various groups of cells using the dual and compensated voltage technique on the $V/2$ write scheme in the presence of line resistance. K is the percentage of voltage to be extracted from V_{word} for onward application to V_{sel_bit} to supplement the write voltage V_w.

	Voltage Drop Formula	**Voltage drop**
Selected cell(s)	$V_{sel\_word} - V_{sel\_bit}$	$V_w - (-K.V_w)/2 = (V_w(2+K))/2$
Group II cell(s)	$V_{word} - V_{sel\_bit}$	$((1-K)V_w)/2 - (-K.V_w)/2 = V_w/2$
Group III cell(s)	$V_{sel\_word} - V_{bit}$	$V_w - V_w/2 = V_w/2$
Group IV cell(s)	$V_{word} - V_{bit}$	$((1-K)V_w)/2 - V_w/2 = -K.V_w/2$

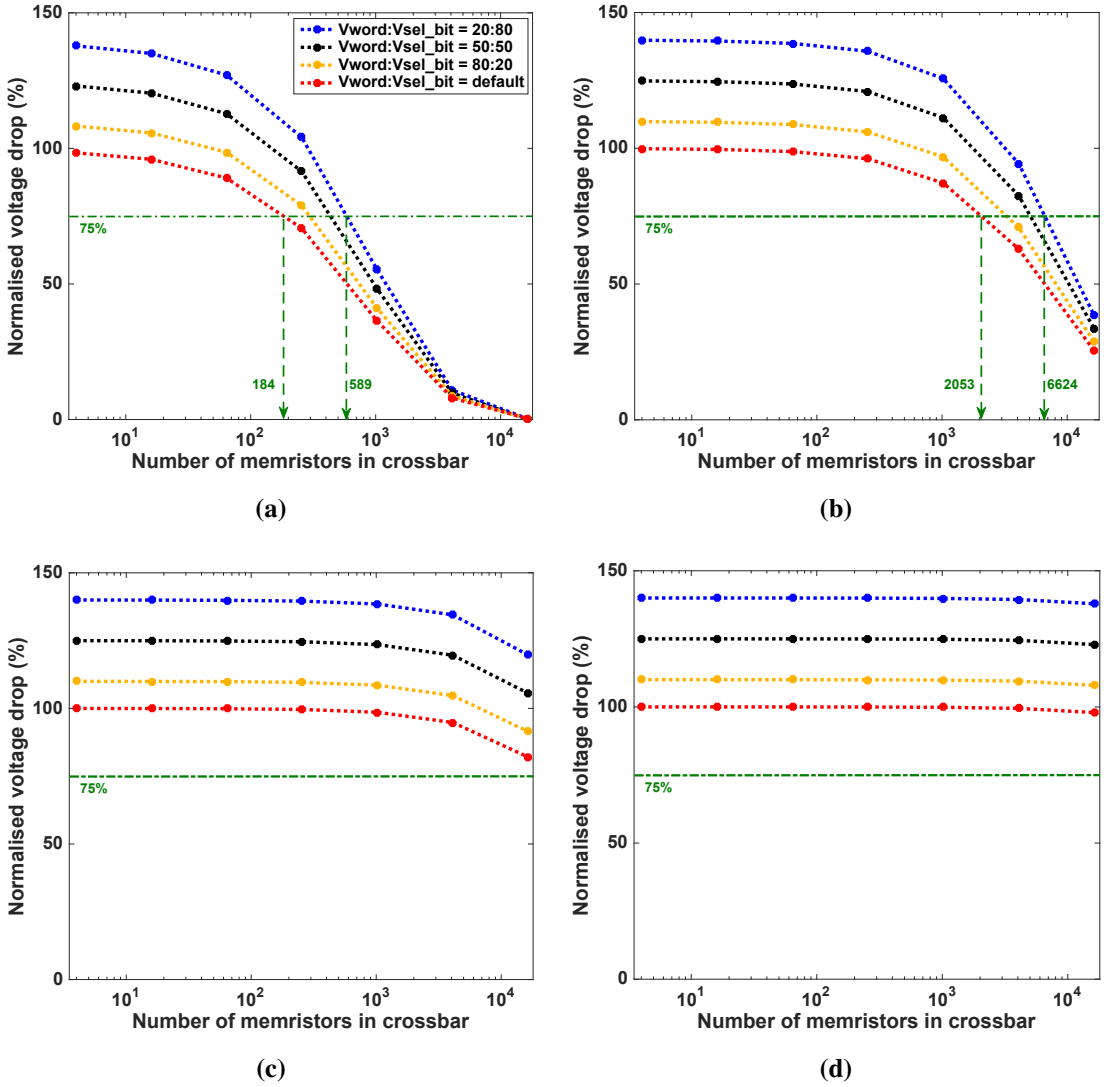


Figure 5.13: Simulation results showing voltage drop on the worst case selected cell over a range of crossbar sizes using the proposed compensated voltage technique with R_L/R_{on} values set to: (a) 10^{-2} **(b)** 10^{-3} **(c)** 10^{-4} **(d)** 10^{-5}**.**

It is expected that the power consumption of this technique will be more than the conventional $V/2$ scheme. This can however be managed by applying this technique to the conventional single side write method which will further reduce power consumption. A much smaller V_{word}:V_{sel_bit} ratio can also be used to drive down power consumptions.

5.4 Multiple Cells Write Operation

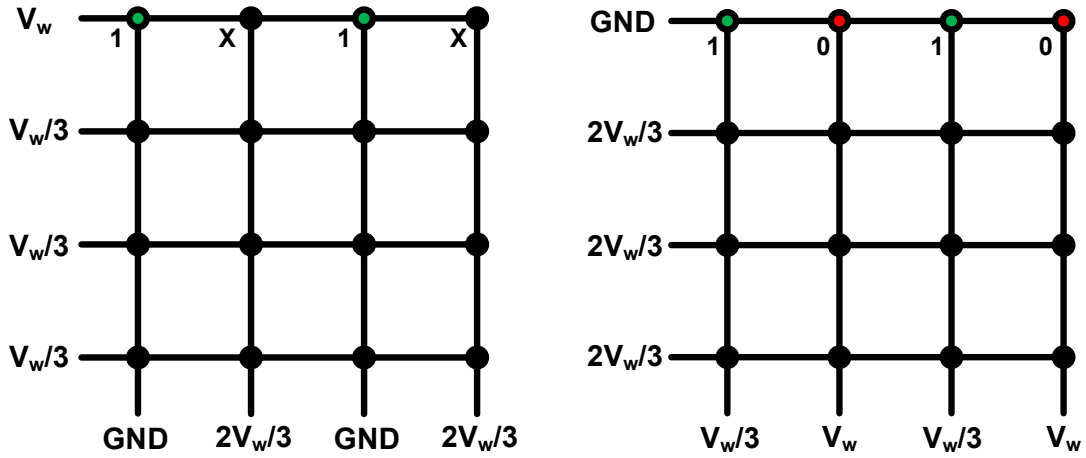
As in the case of read operation, it is also possible to write multiple cells in a crossbar row simultaneously by applying a write voltage to the selected rows and biasing the columns that has the target cells. This simple technique works well if only all the targets cells are to be programmed to the same value. However, write operation becomes more complex

when the target cells are to be programmed with different values. We assume the memristors are in the bipolar mode of operation (can only be switched ON (OFF) by a positive voltage and switched OFF (ON) only by a negative voltage). Writing non-similar data in multiple cells require the SET operation to write a logic 1 (ON) and the CLEAR (RESET) operation to write a logic 0 (OFF). Two write schemes for writing non-similar data in multiple cells in memristor-based array were proposed in [32] using the V/2 write scheme: SET-before-RESET (S-b-R) and ERASE-before-RESET (E-b-R). These schemes are well described in Section 3. Designing the multiple bits write scheme using the $V/2$ scheme means that the unselected cells have a 0.5 probability of accidentally switching unlike the $V/3$ scheme that guarantees a 0.7 probability rate for the preservation for the state of the unselected cells. The following section presents a first implementation of the multiple bits write technique using the $V/3$ write scheme.

5.4.1 Low Power $V/3$ Write Scheme for Multiple Crossbar Cells

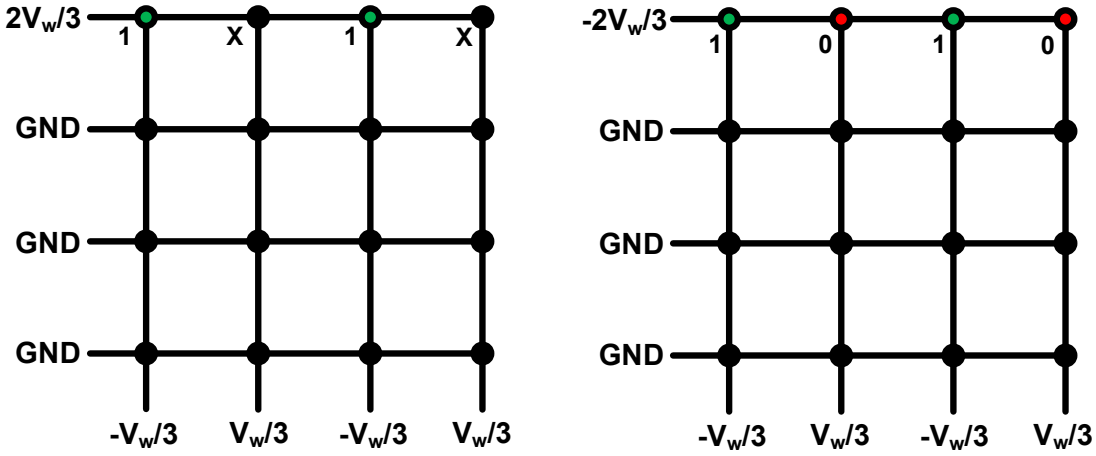
The S-b-R and E-b-R multiple cells write techniques proposed in [32] and described in Section 3 were designed using the $V/2$ write scheme earlier described. During write operation on a single cell, when compared with the $V/3$ write scheme, the $V/2$ scheme has the advantage of consuming less power and also reducing the number of cells that can be perturbed. One major downside of the $V/2$ scheme during single cell write is the high probability of the unselected cells being disturbed. Although reliability in $V/2$ scheme can be improved by ensuring the voltage is only applied for a short duration, this can be difficult for multiple cells write due to threshold voltage variation. The few cells $(m+n-2)$ exposed to voltage disturbance in $V/2$ have a 0.5 probability of being affected while the exposed $(mn-1)$ cells in the $V/3$ scheme have a 0.33 chance of being disturbed. In this section, we propose for the first time to apply the S-b-R and E-b-R multiple cells write techniques to crossbar arrays using the $V/3$ write scheme (Fig. 5.14, 5.15, 5.16 and 5.17). An additional technique, ERASE-before-SET (E-b-S) will also be implemented (Fig. 5.18 and 5.19). E-b-S was mentioned as an alternative to E-b-R in [32] but was not implemented. Two configurations of each of these techniques are explored in this section, these configurations are based on the position of the voltage levels. Configuration I (Fig. 5.14, 5.16 and 5.18) follows similar pattern as the $V/2$ scheme proposed in [32] but for the $V/3$ write scheme used in this case. Configuration II (Fig. 5.14, 5.17 and 5.18) for multiple cells write operation using the $V/3$ is based on asymmetric voltage application as

done with $V/2$ scheme in [106]. In the second configuration, a voltage of $2V_w/3$ ($-2V_w/3$) is applied to the selected wordline and the selected bitlines will have a voltage of $-V_w/3$ ($V_w/3$) applied to result in a voltage of V_w drop across the target cells. The unselected wordlines are grounded and the unselected bitlines are exposed to $V_w/3$ ($-V_w/3$) voltage so that the unselected cells have a maximum safe voltage of $V_w/3$ or $-V_w/3$ across them. Both configurations consume similar power but Configuration II requires less switching of voltage levels when the opposite data needs to be written after the first writing phase. Table 5.6 shows a detailed comparison between the two configurations of the new scheme and the conventional $V/2$ scheme.



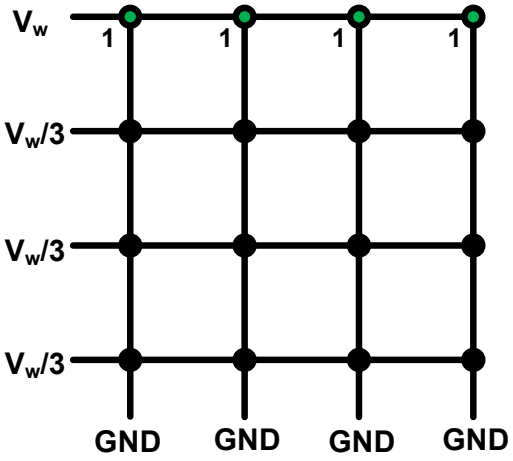
(a) SET Phase - Write of $1\,X\,1\,X$ **(b) RESET Phase - Write of $X\,0\,X\,0$**

Figure 5.14: Configuration I: Proposed SET-before-RESET technique using the $V/3$ Write Scheme.

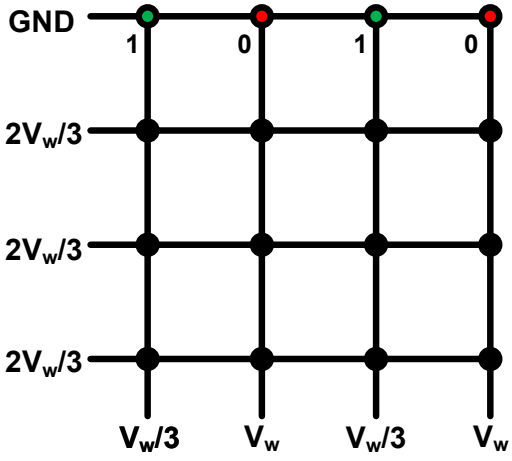


(a) SET Phase - Write of $1\,X\,1\,X$ **(b) RESET Phase - Write of $X\,0\,X\,0$**

Figure 5.15: Configuration II: Proposed SET-before-RESET technique using the $V/3$ Write Scheme.

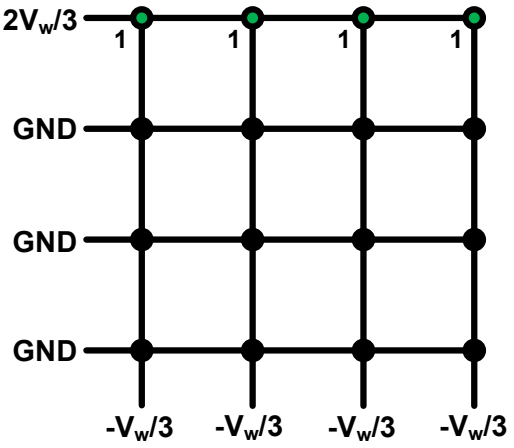


(a) ERASE Phase - Write of 1 1 1 1

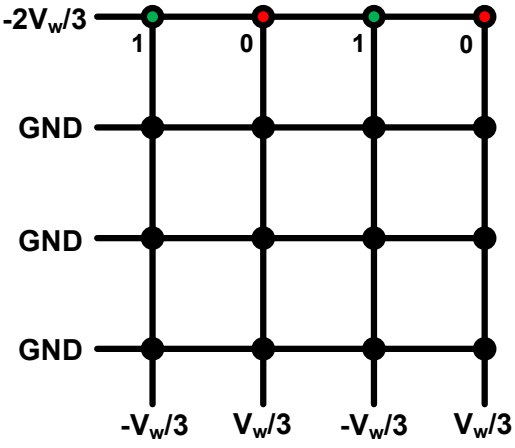


(b) RESET Phase - Write of $X\,0\,X\,0$

Figure 5.16: Configuration I: Proposed ERASE-before-RESET technique using the $V/3$ Write Scheme.

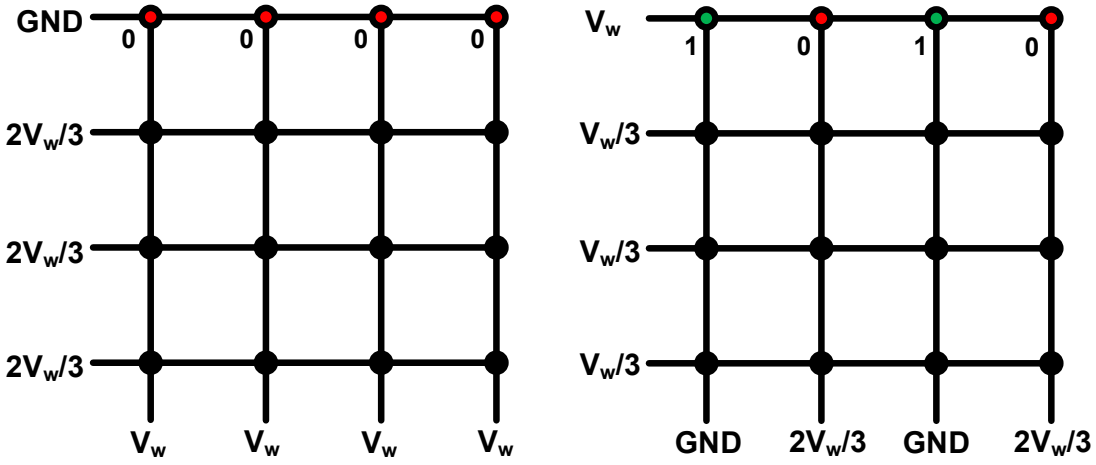


(a) ERASE Phase - Write of 1 1 1 1



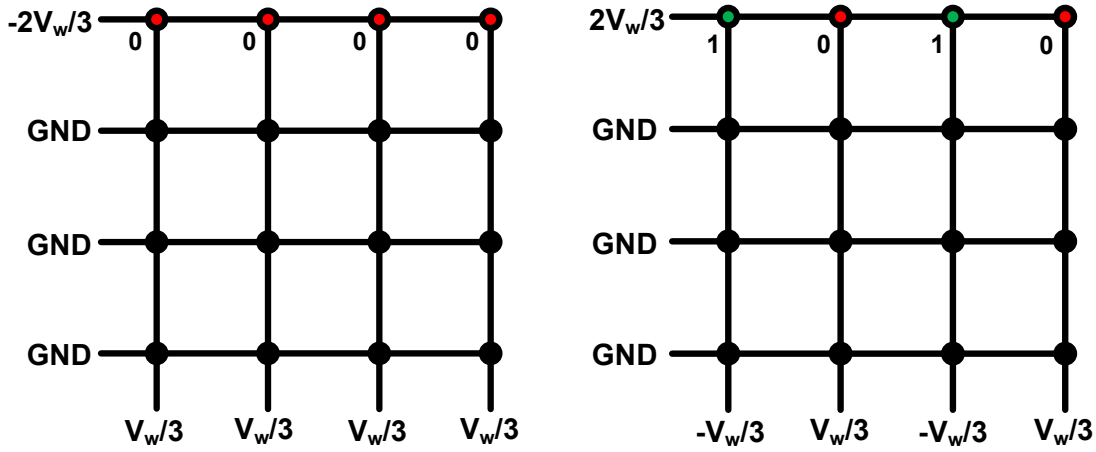
(b) RESET Phase - Write of $X\,0\,X\,0$

Figure 5.17: Configuration II: Proposed ERASE-before-RESET technique using the $V/3$ Write Scheme.



(a) ERASE Phase - Write of 0000 **(b) SET Phase - Write of** $1X1X$

Figure 5.18: Configuration I: Proposed ERASE-before-SET technique using the $V/3$ **Write Scheme.**



(a) ERASE Phase - Write of 0000 **(b) SET Phase - Write of** $1X1X$

Figure 5.19: Configuration II: Proposed SET-before-RESET technique using the $V/3$ **Write Scheme.**

Table 5.6: A comparison multiple cells write techniques. Values in braces applies to the second phase of the write operation if required.

Write Scheme	**V/2** [32]		**V/3** (Proposed)					
Write Technique	**S-b-R**	**E-b-R**	**S-b-R**		**E-b-R**		**E-b-S**	
Configuration	**I**	**I**	**I**	**II**	**I**	**II**	**I**	**II**
Selected Wordline	V_w (GND)	V_w (GND)	V_w (GND)	$2V_w/3$ ($-2V_w/3$)	V_w (GND)	$2V_w/3$ ($-2V_w/3$)	GND (V_w)	$-2V_w/3$ ($2V_w/3$)
Selected Bitlines	GND (V_w)	GND (V_w)	GND (V_w)	$-V_w/3$ ($V_w/3$)	GND (V_w)	$-V_w/3$ ($V_w/3$)	V_w (GND)	$V_w/3$ ($-V_w/3$)
Unselected Wordlines	$V_w/2$ ($V_w/2$)	$V_w/2$ ($V_w/2$)	$V_w/3$ ($2V_w/3$)	GND (GND)	$V_w/3$ ($2V_w/3$)	GND (GND)	$2V_w/3$ ($V_w/3$)	GND (GND)
Unselected Bitlines	$V_w/2$ ($V_w/2$)	$V_w/2$ ($V_w/2$)	$2V_w/3$ ($V_w/3$)	$V_w/3$ ($-V_w/3$)	NA ($V_w/3$)	NA ($-V_w/3$)	NA ($2V_w/3$)	NA ($V_w/3$)
Number of Voltage Levels	3 (3)	3 (3)	4 (4)	4 (4)	3 (4)	3 (4)	3 (4)	3 (4)
Number of Voltage Level Switch	2	2	4	3	3	2	3	2

5.4.2 Experimental Results and Discussions

In order to fairly simulate the power consumption of the multiple cells write schemes. We simulate a single phase of writing a logic one into all the selected cells while all other cells are kept at a low resistance value R_{on} where their power consumption is at its maximum. In Fig. 5.20, the conventional multiple cells write with the $V/2$ write scheme and the newly proposed multiple cells write techniques with the $V/3$ write scheme are compared in terms of power consumptions. Passive crossbar arrays ranging from 4×4 to 2048×2048 were simulated with different number of cells selected to be written into. V_w is $2V$, $R_{on} = 100\Omega$ and $R_{off} = 200K\Omega$. It is worth mentioning again that only the unselected cells on the selected wordline and bitlines are exposed to $V_w/2$ in the $V/2$ scheme, while all other unselected cells have *zero* voltage across them. In the $V/3$ scheme, all unselected cells have a fixed voltage of $V_w/3$ or $-V_w/3$. Fig. 5.20(a) shows the power consumption result of writing a logic 1 into $n/4$ number of cells in the selected wordline of the crossbar arrays. The $V/3$ scheme consumes more power in this case. In the simulation result of Fig. 5.20(b), $n/2$ number of cells are written into. Here, the $V/2$ scheme now consumes more power than the $V/3$ scheme across all the crossbar arrays. The increased power consumption of the $V/2$ scheme in Fig. 5.20(b) is due to more cells experiencing the high $V_w/2$ voltage. As the number of selected cells increases, the number of half-selected cells that experiences the $V_w/2$ voltage increases thereby increasing the overall average power consumption. Fig. 5.20(c) and 5.20(d) show the power consumption for $3n/4$ and n selected cells respectively. It is thus easy to conclude that $V/3$ write scheme conserves more power than the $V/2$ during multiple cells write operation. The worst case power consumption for both schemes occur when all the cells in the selected wordline are selected for simultaneous write of logic 1 and the crossbar array size is at the maximum. Only one wordline can be selected for write at a time. In Fig. 5.21, 64×64 and 512×512 passive crossbar arrays was simulated with the number of selected cells gradually varied from 1 to 64 (512). The power consumed by the $V/3$ was enormous at first but it remained steady as the number of cells being written into increases. This constant trend is due to the fact that all unselected cells have a voltage drop of $V_w/3$ across them irrespective of the number of cells selected. Only the power from the selected cells accumulates as their number increases which is not massive in an array dominated by unselected cells. The $V/2$ scheme started well but rises steadily as the number of selected cells increase (more cells experience $V_w/2$ voltage across them and less cells have a voltage drop of zero) and

thus consumes much more power than the $V/3$ scheme as the number of cells being written into approaches maximum. The $V/3$ multiple cells write scheme saves up to 55% in power consumption compared to the conventional $V/2$ scheme when all the cells in the wordline of a typical 512×512 crossbar array are written simultaneously.

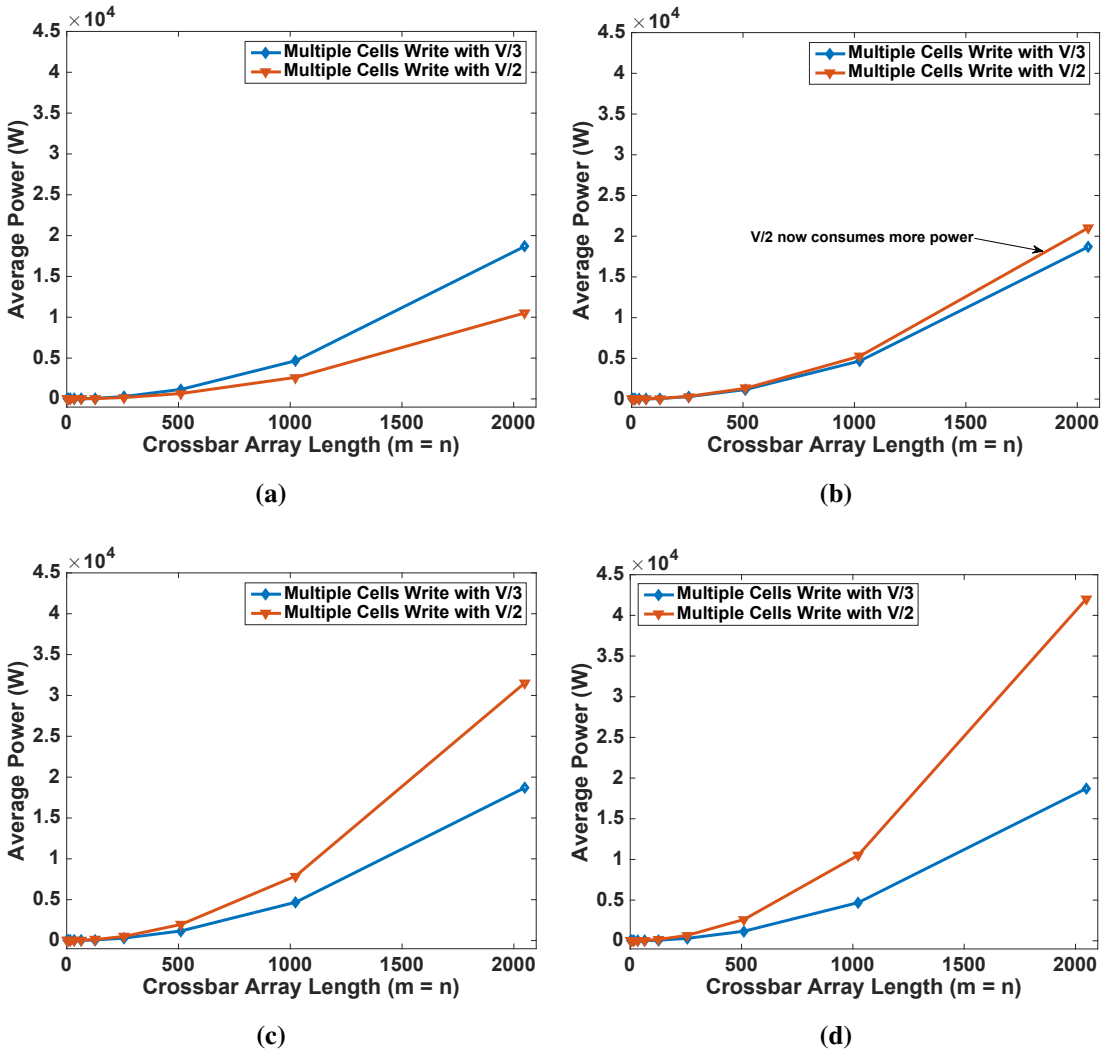
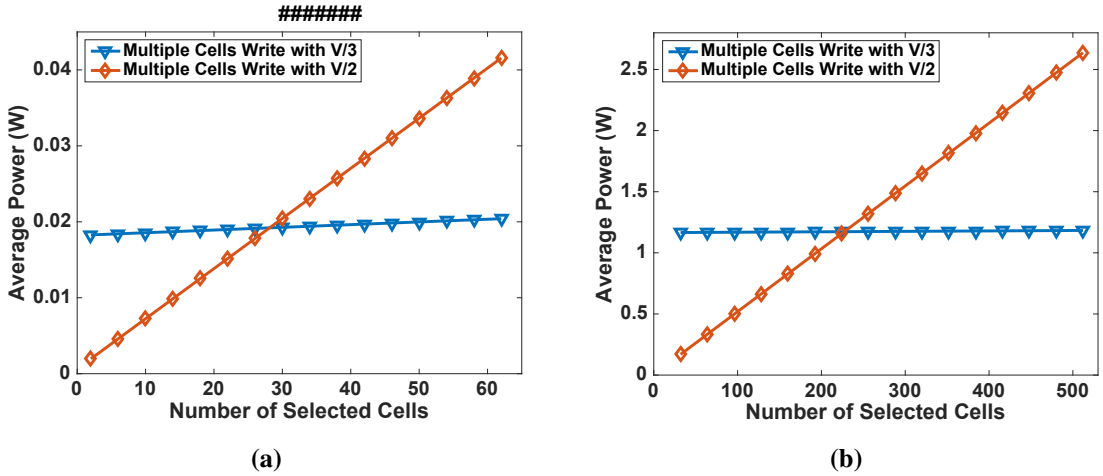


Figure 5.20: Power consumptions of the $V/3$ and $V/2$ schemes over a range of crossbar array size during multiple cells write operation. Number of cells selected for write: (a) $n/4$ cells selected (b) $n/2$ cells selected (c) $3n/4$ cells selected (d) n cells selected. Power values were scaled up to enhance the visibility of the difference between the schemes.



(a) (b)

Figure 5.21: Comparison of power consumptions between the conventional $V/2$ and the proposed $V/3$ write scheme during multiple cells with an array of (a) 64×64 and (b) 512×512. Power is scaled and computed based on the simulation of writing a one in the selected cells.

5.5 Conclusions

In an effort to improve the reliability of write operations in crossbar memories, an accurate analytical modelling of the crossbar write operation was developed especially for denser arrays. The crossbar analytic models derived in this chapter caters for the real case where each group of cells in the crossbar is free to include both high and low resistance state cells simultaneously. With this model, analysis was carried out on the performances of three existing crossbar write schemes (floating lines, $V/2$ and $V/3$) with and without line resistance consideration. The floating scheme is only considered reliable only when the array is square shaped. The $V/2$ scheme protects more cells than any other scheme but the state of the $m+n-2$ cells constantly exposed to a voltage of $V_w/2$ can be overwritten if exposed to voltage for a long time. The $V/3$ scheme on the other hand, exposes all the groups of unselected cells to a more reasonable low voltage of $V_w/3$ but at the expense of power conservation. Based on these initial analyses, a compensated voltage technique was designed for the low-power $V/2$ scheme using the dual voltage method to increase the maximum array size that can be designed and in turn reduce the possibility of write failure. With this compensation technique, the maximum array size can be increased up to three times depending on selected parameters. This chapter also proposed a new multiple cells write technique using the $V/3$ write scheme with two different configurations. A comparison of the conventional scheme and the new technique shows up to a 55% reduction in power consumption when all the cells in the wordline of a typical 512×512 crossbar array are written simultaneously.

Chapter 6

Gas Sensing with Memristor-based Crossbar Array

6.1 Introduction

Memristive devices are traditionally used in memory, logic and neuromorphic systems. Gas sensing is one of the recently discovered application fields of the memristor. This work is the first piece that considers memristor arrays for use as a gas sensor. Gas sensing with memristor could lead to unprecedented sensor density and ubiquity in electronic systems. Gas sensing with memristor offers several advantages as previously mentioned in Chapter 1 and 2. Metal oxide gas sensors operate based on internal resistance change in the presence of the target gases. Memristors consists primarily of metal oxide which changes its resistance based on the polarity and magnitude of an applied voltage. In this chapter, a framework for efficient gas detection using memristor based crossbar array is proposed and analysed using information and techniques developed from the crossbar read and write operation detailed in Chapter 4 and 5 respectively. A novel Verilog-A based memristor model that emulates the gas sensing behaviour of doped metal oxides is developed for simulation and integration with design automation tools. Using this model, we propose and analyse three different gas detection architectures based on array of memristor-based sensors. Gas presence, together with some of its properties, can be detected using resistance changes and spatial information from one or a group of memristive sensors. Our simulation results show that depending on the organisation of the memristive elements and the sensing method, the response of the sensor varies providing a broader design space for future designers. For instance, with a 8×8 memristor sensor

Table 6.1: Effect of increasing gas concentration on the resistance of semiconductor metal oxide [107].

Classification		**Gas Type**	
		Reducing	Oxidising
Semiconductor	p-type	↑	↓
	n-type	↓	↑

array, there is a ten times improvement in the accuracy of the sensor's response when compared with a single memristor sensor.

This chapter explores memristor-based sensor structures in a crossbar architecture with emphasis on the sensor's response to gas presence. A fundamental question that this work poses and attempts to answer is, given a fixed array of memristors what is the best arrangement to achieve better sensitivity. In particular, we investigate if it is better to have: 1) fewer sensor elements with a low accuracy; 2) large array of sensors with a high response rate and reliability; or 3) a trade off between the previous two extremes. The critical argument is that when the size and number of sensors in the crossbar increases (decreases), the performance of the overall sensor structure improves (degrades) in a similar fashion. In achieving this aim, a suitable Verilog-A memristor model is developed and three different crossbar sensing structures are analysed with this model.

The rest of the chapter is organized as follows: Section 6.2 describes the adaptation of memristor for use as a gas sensor. Section 6.3 presents a description of the proposed Verilog-A memristor model. In Section 6.4, we show a more detailed analysis of the crossbar architecture and some key sensing problems are discussed. Section 6.5 describes the proposed structures for gas sensing using memristor arrays. Further analysis of simulation results were presented in Section 6.5.4 and Section 6.6 concludes the chapter.

6.2 Memristor as Gas Sensor

The basic mode of operation of a metal oxide gas sensor is the control of the surface potential barrier by the adsorbed surface-charge. The interaction of a target gas with the surface of a metal oxide sensor results in a change in the concentration of charge carriers in the material. The change in the concentration causes resistivity alteration in the metal oxide. In order for a memristor to be effective as a gas sensor, a slight modification will be incorporated into devices fabricated for sensing purpose. The physical memristor will

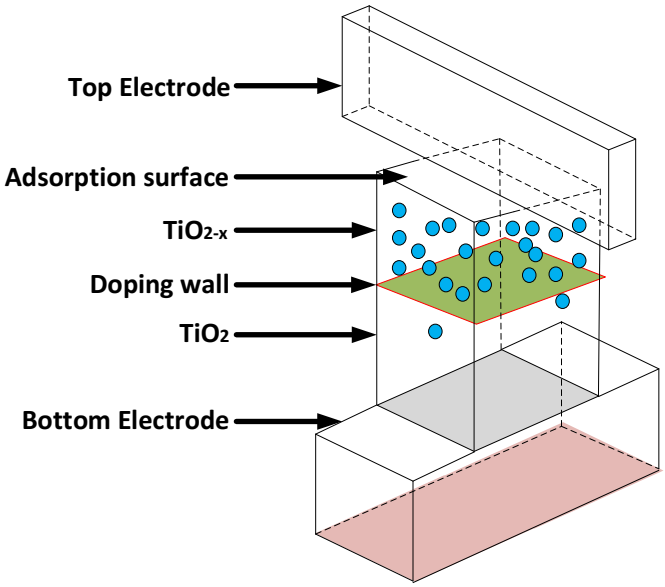


Figure 6.1: Structure of memristor for sensing applications. Both sides of the top contact is left uncovered to enhance interaction of target gas with the titanium oxide layer.

feature a partially covered top terminal as shown in Fig. 6.1 to allow gas interaction with the semiconductor layer. The exposed part of the TiO_{2-x} thin film allows for easy interaction between the gas and the surface of the memristor via both sides of the top contact. The degree and direction of resistance change depends on the metal oxide semiconductor material in the sensor and the type and concentration of the subject gas. Table 6.1 shows a summary of how the resistance of metal oxides react to the presence of different gases. The sensor design in this thesis focuses on TiO_2 based memristor. TiO_2 is regarded as a reliable chemical sensor because of its high sensitivity, low cost, fast response time and stability [90, 108]. Gas adsorbed by the surface of a TiO_2 film increases or decreases the resistance of the film as a result of interactions among electrons in the film. A typical structure of TiO_2 based memristor is the HP Labs memristor shown in Fig. 6.1 and described in Section 2.2.1 of Chapter 2. HP Labs' memristor consists of two layers; a less conductive TiO_2 layer and a conductive oxygen-deficient titanium dioxide TiO_{2-x} layer. We propose to use this memristor as a sensor by initialising the memristor to the R_{on} state by applying a high positive voltage to the positively charged TiO_{2-x} layer. This repels the oxygen vacancies in the TiO_{2-x} region further into the TiO_2 layer, thereby making the device almost completely filled with the more conductive TiO_{2-x} from the surface. The total resistance (memristance) of the memristor is determined by two variable resistors R_{off} and R_{on} representing the resistance of the undoped (TiO_2) and the doped (TiO_{2-x}) regions of the memristor respectively (see Section 2.2.1). TiO_{2-x} is p-type semiconductor and its resistivity (R_{on}) decreases (increases) in the presence of an oxidising (reducing)

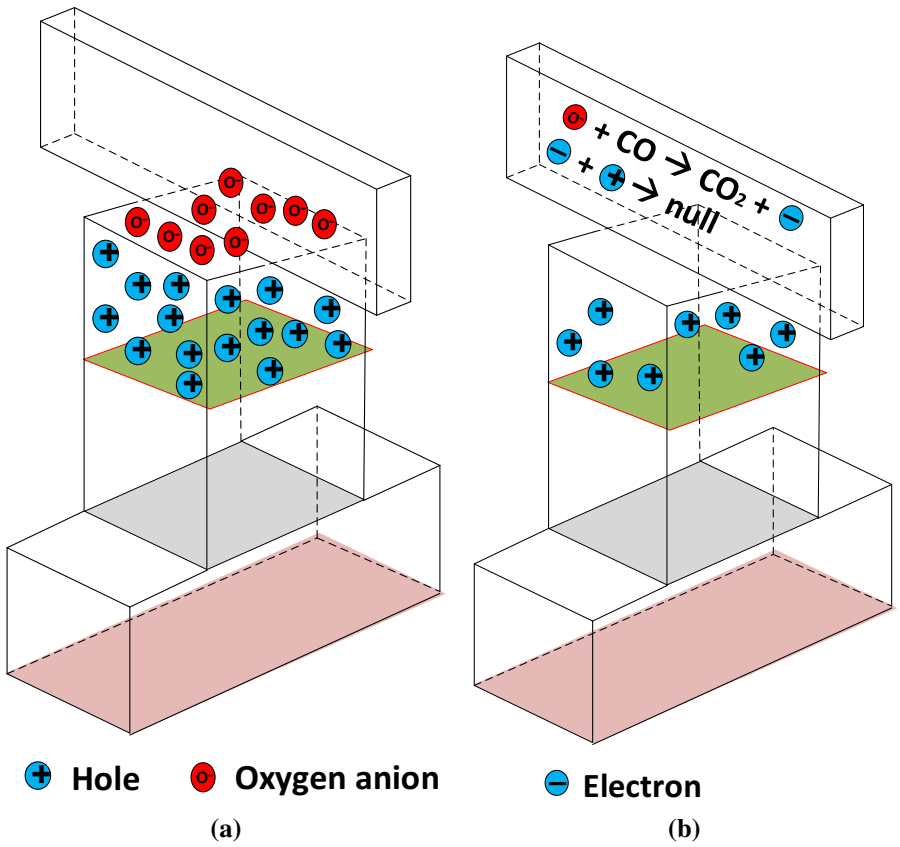


Figure 6.2: Interaction of reducing gas with the surface of TiO_2 based memristor. (a) Oxygen anion are adsorbed to the surface of the memristor on exposure to air (b) In the presence of a reducing gas such as *CO*, the oxygen anion react with the gas to release electron that are injected into the thin film.

gas.

The concentration of charge carrier in the TiO_2 thin film will increase (decrease) in the presence of an oxidising (reducing) gas since holes (electrons) are generated from the interaction of such gas with the oxygen anion (O^-) adsorbed at the surface of the device [86, 108, 109]. Fig. 6.2(a) shows oxygen ion adsorbed to the surface of the memristor on exposure to air. Whenever a reducing gas such as carbon monoxide (CO) is exposed to the surface of the memristor sensor, oxidation reaction takes place between the *CO* and the oxygen anions to emit electrons as shown in Fig. 6.2(b) according to Eqn. 6.1. The emitted electrons in turn nullifies the positively charged carriers in the TiO_{2-x} region (Eqn. 6.2) thereby increasing its resistance and the resistance of the entire device [109].

$$CO + O^- \xrightarrow{Oxidation} CO_2 + e^- \tag{6.1}$$

$$h^+ + e^- \longrightarrow \text{null} \tag{6.2}$$

6.3 Verilog-A Model for Memristive Gas Sensor

Prior to analysis of memristor-based structures suitable for gas sensing, there is the need to develop an effective memristor sensor model as there is not a known one in existence. This model accounts for gas concentration as an input to the memristor model as this forms the basis for memristor's application in gas detection. Generally, when semiconducting metal oxide are used as gas sensor, the change of the sensor's resistance is as a result of the loss (gain) of free charge carriers (electrons or holes) from (to) the semiconductor to (from) its surface [110].

```
1  `include "disciplines.vams"
2  `include "constants.h"
3  nature distance
4     access = Metr;
5     units = "m";
6     abstol = 0.01n;
7  endnature
8
9  discipline Distance
10    potential distance;
11 enddiscipline
12
13 module Memristor(p, n, c, x_position);
14 //in: p, c, out: n, x_position; define in & out port;
15 input p; //positive pin
16 input c; //concentration
17 output n;    //negative pin
18 output w_position;    // w-width pin
19 electrical p, n, gnd;
20 Distance w_position, c;
21 ...
22 analog function integer stp;  //Stp function
23    real arg; input arg;
24    stp = (arg >= 0 ? 1 : 0 );
25 endfunction
26 analog begin
27   I_ion= k *sinh(alpha*V(p,n));
28   f_drift = 1-pow(x/D-stp(-I_ion), 2*p_coeff);
29   Ron_eff = Ron*(1+A*pow(Metr(c), beta));//Reducing gas
30   //Ron_eff = Ron/(1+A*pow(Metr(c), beta));//Oxidising gas
31   Roff_eff = Roff //Roff_eff could also be set to change similar to Ron_eff
32   dxdt =(uv*Ron_eff/D)*I(p,n)*f_drift;
33   x = idt(dxdt, x_0);
34   ..// quantize x to [0, D]
35   R=Ron_eff*x/D+Roff_eff*(1-x/D);
```

```
36    I(p,n) <+  1/R*V(p,n);
37    Metr(x_position) <+ x;
38  end
39  endmodule
```

Listing 6.1: Verilog-A description of proposed memristor model for gas sensing

Any of the memristor models based on a gas sensitive metal oxide semiconductor can be adapted for the purpose of simulating a gas sensor. We present an adaption of the physical memristor fabricated by HP labs for use as a memristive gas sensor. As earlier mentioned, HP's memristor consists of two regions; the perfect and less conductive TiO_2 and the conductive TiO_{2-x} with oxygen vacancies (holes) as described in Section 2.2.1. While the practical viability of the HP model has been contested in memory design, it is sufficient for sensor simulations as the resistance changes depends majorly on the gas concentration. The positively charged oxygen vacancies makes the TiO_{2-x} material conductive [30]. Gas interaction will occur at the surface of the TiO_{2-x} layer where the number of positively charged holes will increase or decrease depending on the properties of the subject gas (see Table 6.1). We assume the ion mobility rate will remain at its average value.

Algorithm 6.1: Memristor gas sensor

Function `Memristor Sensor`
- **Input:** Conc (C), R_{on}, R_{off}
- **Output:** memristance
- $w(0)$ = initial state
- D = Memristor's size;
- **if** *reducing gas* **then**
  - $R_{on\_eff} = R_{on}(1 + A[C]^{\beta})$;
- **end**
- **if** *oxizidising gas* **then**
  - $R_{on\_eff} = R_{on}/(1 + A[C]^{\beta})$;
- **end**
- $R_{mem}^{final} = R_{on\_eff}\frac{w(t)}{D} + R_{off\_eff}\left(1 - \frac{w(t)}{D}\right)$;
- return R_{mem}^{final};

end

The Verilog-A model listed in listing. 6.1 and summarised in Alg. 6.1 will facilitate easy integration and simulation of memristor-based sensors with design automation tools. For brevity, only the main functions are shown. The model was built on the basic memris-

tor model presented in [111] and further expanded in [101]. The Biolek window function [57] to ensure state variable x remains within the 0 and D boundary is shown in line 28. The Butler-Volmer equation in line 27 is used to introduce some degree of non-linearity to the current-voltage (I-V) characteristics with programmable thresholds. k and α are fitting constants for characterising the memristor's state. The response (S) of sensors to gas presence is often defined in various forms such as $S = R_{init}/R_{final}$, $S = R_{final}/R_{init}$, $S = |R_{init} - R_{final}|/R_{init}$, $S = |R_{init} - R_{final}|/R_{final}$, where R_{final} and R_{init} are the final and initial resistance of the sensor after and before exposure to gas respectively. The value of R_{on} changes to R_{on_eff} using equation in line 29 from the adaptable response model equation in Eqn. 6.3 [107, 112, 113]. The adaptation of this equation depends on the type of gas interacting with the memristor. Therefore, for any oxidising gas of concentration C, the sensitivity of a gas sensor made of p-type semiconductor can be represented directly by Eqn. 6.3. Clearly, the sensitivity of a p-type material to a reducing gas can also be represented by the inverse of Eqn. 6.3.

$$S = \frac{R_{init}}{R_{final}} = 1 + A[C]^{\beta} \tag{6.3}$$

The response equation above is based on the concentration C (one of the three input ports in the model) of the target gas. Where A is the constant parameter or sensitivity coefficient for the material of the semiconductor and β is the response order for the subject gas. R_{on} (R_{off}) and R_{on_eff} (R_{off_eff}) in the Verilog-A model replaces R_{init} and R_{final} respectively in Eqn. 6.3. Depending on the type of gas, Eqn. 6.3 can be adapted for either an oxidising or reducing gas as summarised in Table 6.1. TiO_2 is a n-type semiconductor metal oxide but can be transformed into a p-type by doping [108]. HP Lab's memristor can be regarded as a p-type metal oxide semiconductor material when used as a gas sensor with the doped TiO_{2-x} region exposed to the subject gas.

The Verilog-A model is developed based on the generally accepted assumption that only the region of the memristor directly exposed to the target gas experiences a change in resistance [112, 109]. The model thus uses the new R_{on_eff} and R_{off_eff} to compute the final memristance (R_{mem}^{final}) of the device after exposure to the target gas using Eqn. 6.5. The initial memristance of the device is calculated with R_{on} as in Eqn. 6.4. Eqn. 6.4 and 6.5 are an adaptation of the original TiO_2 based memristor equations proposed by HP

Labs [30].

$$R_{mem}^{init} = R_{on}\frac{w(t)}{D} + R_{off}\left(1 - \frac{w(t)}{D}\right) \tag{6.4}$$

$$R_{mem}^{final} = R_{on\_eff}\frac{w(t)}{D} + R_{off\_eff}\left(1 - \frac{w(t)}{D}\right) \tag{6.5}$$

where $0 \leq w(t) \leq D$ is the time dependent state variable acting as a boundary between the doped and undoped regions. R_{off_eff} and R_{on_eff} are the effective new resistance values of the two regions after exposure to gas. The memristor is initialized to R_{on} ($w = D$) such that it constitutes mostly oxygen deficient TiO_{2-x}. Interaction of gases with the doped region changes R_{on} to R_{on_eff} without shifting the state variable. The extent of change to the resistance of the doped region depends on the carrier concentration and type of gas. As portrayed in the Verilog-A model, $R_{off\_eff} \approx R_{off}$ because the gas has negligible interaction with the undoped region of the memristor, the Verilog-A model could also be adjusted to accommodate changes in R_{off}.

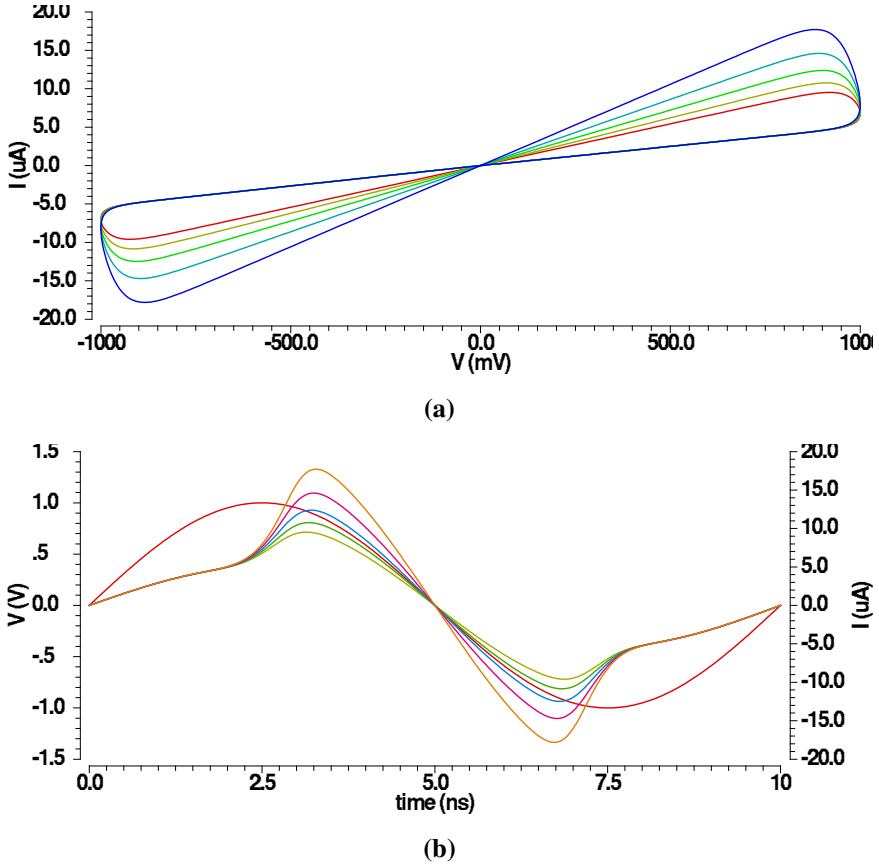


Figure 6.3: (a) I-V characteristics of the proposed Verilog-A memristor model over 5 steps of an oxidising gas concentration ($0\,ppm$ (red line) - $1000\,ppm$ (blue line)) (b) The applied voltage (red) and resulting currents (other colours) from varying the value of gas concentration ($0\,ppm$ (green line) - $1000\,ppm$ (yellow line)). Initial R_{on} and R_{off} are $100\,\Omega$ and $200K\,\Omega$ respectively, $\beta = 1$[112] and $A = 4.2 \times 10^{-4}$.

The I-V characteristics of the developed model was plotted over a range of gas concentration to ensure the device still retains its memristive behaviour. A sinusoidal voltage was applied to the device and the corresponding current responses shown in Fig. 6.3(b) are very similar to those reported in various experimental results in literature [57, 114]. Fig. 6.3(a) shows the I-V curve over a parametric range ($0\,ppm$ - $1000\,ppm$) of a reducing gas concentration. From the I-V characteristics, the R_{off} value remains almost constant because the gas interacts less with the undoped layer of the memristor as previously explained. On the other hand, R_{on} reduces to R_{on_eff} causing increased current as gas concentration increases as seen by the multiple coloured lines on the vertical plane of curve in Fig. 6.3(a). The I-V characteristics were obtained through simulation in *Cadence Virtuoso* with a sinusoidal input voltage to the sensor element.

The authors in [115] presented the effects of temperature on the I-V characteristics of Tantalum-oxide-based resistive memories. Their results show a 60°C temperature change triggering a negligible change in the hysteresis curve of the device. The technique of [99] also showed that the TiO_2 gas sensor can effectively respond to gas presence in the range 24-100°C. In addition, it has been demonstrated in [8] that a current change caused by parasitic effects only move the I-V curve's pinch-point marginally. As the overall current in the sensor is expected to be low, our model did not consider the effects of temperature changes. With the Verilog-A model firmly established, the sensing capability of a single isolated memristor was verified with an oxidising gas. Fig. 6.4 shows a single memristor's response as a function of gas concentrations ranging from $0\,ppm$ to $10^4\,ppm$. Expectedly, the response rate increases with increasing gas concentrations. For the development of the Verilog-A sensor model, we have made the following assumptions:

1. The memristor operates at a temperature sufficient for the resistance of the thin film to change in the presence of gases. Metal oxide sensors are able to operate at room temperature [90, 91].

2. The gas concentration causes an absolute change in the value of R_{on} alone which causes no significant change to the position of the state variable x.

3. The resistance value of the memristor could be read-out with or without the presence of the target gas using appropriate read circuitry.

4. Recovery time of memristor depends on the properties of the material in use.

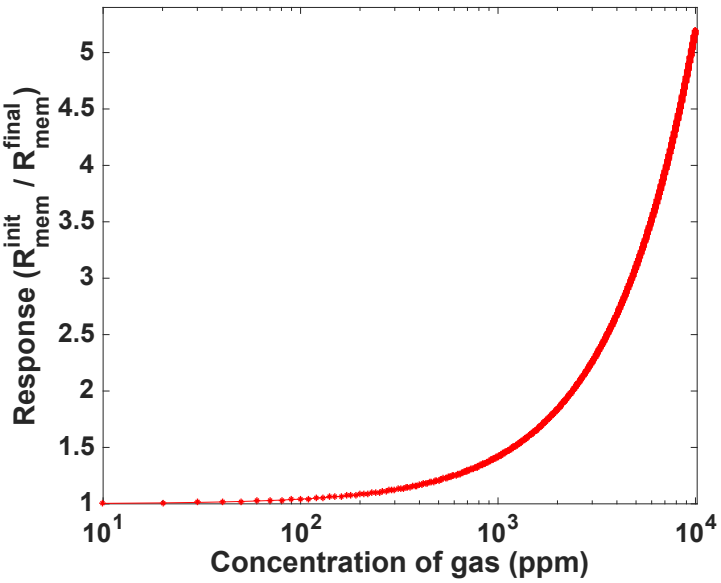


Figure 6.4: Sensor response to different concentrations of an oxidising gas. R_{mem}^{init} is calculated using Eqn. 6.4 with the initial values of R_{on} R_{off}. R_{mem}^{final} (Eqn. 6.5) is the memristance of the sensor after gas interaction, where R_{on} changes to $R_{on_{eff}}$.

6.4 Gas Sensing with Crossbar Array

The use of a single memristor cell as a sensor has been demonstrated experimentally in previous publications [52, 53]. Measurement errors in sensors could be reduced statistically by taking multiple measurements from the same sensor or from different independent sensors [116]. Array of sensors also open up the possibility of detecting multiple gases in real time. The use of MOSFET based sensors in an array to improve performance was demonstrated in [117], where pairs of Pd- and Pt-gate MOSFET were used to detect and analyse gas mixtures. Similarly, the use of commercially available SnO_2 sensors in an array to detect and differentiate organic compounds was also explored in [118]. These array of sensors make it possible to detect, quantify and differentiate multiple gases in real time. Fig. 6.5(a) shows a crossbar array with the cell circled in blue for sensing. The crossbar consists of a set of parallel nanowires perpendicularly placed on another set of parallel nanowires with a memristor cell inserted at every intersecting point of the wires. Each set of parallel wires represents the bitlines and wordlines. In order to sense the resistance of a memristor in a crossbar memory array, a read voltage V_{read} (less than the threshold voltage to switch the memristor) is applied to either the wordline or bitline and the other line is grounded through a load resistor. The other lines can be left floating. However, this simple sensing technique introduces the sneak-path problem that hinders independent sensing of each device in the array. Memristor crossbar architecture suffers

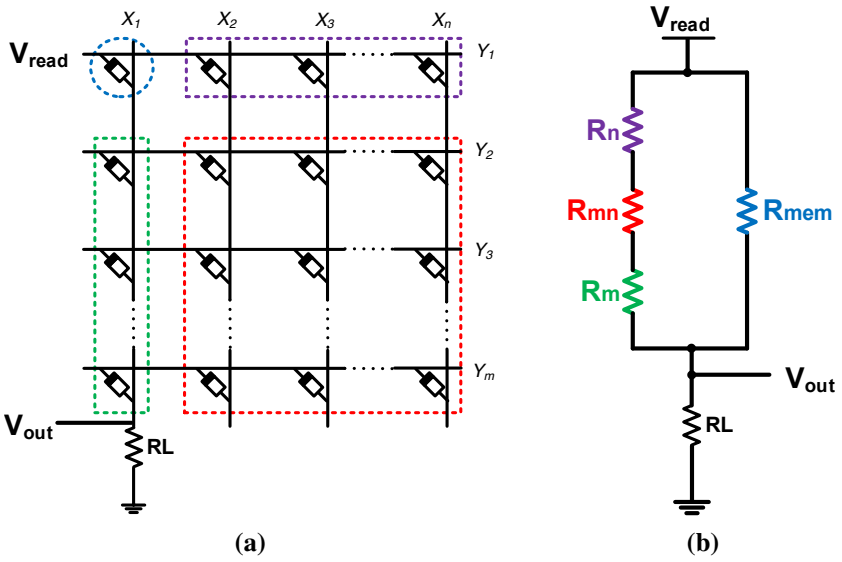


Figure 6.5: (a) A typical crossbar architecture conditioned for reading. The cell circled in blue (R_{mem}) is the target cell (b) Equivalent circuit of the ideal crossbar read operation showing the undesired parallel combination of neighbouring memristors with the target memristor to influence the read output.

from sneak-path because of the bidirectional flow of current in oxide based memristors. Sneak-paths are undesired current paths within the crossbar architecture that may lead to erroneous sensing of the resistance of a memristor cell in a crossbar array. Resistance of unselected cells combine to form a parallel resistance path with the resistance of the desired cell(s) as shown in the equivalent circuit in Fig. 6.5(a) [38]. Sneak-path effect makes it difficult to use each of the memristor in the crossbar array as an individual sensor. Apart from degrading the sensing margin, existence of sneak-paths also leads to increased power consumption. Sneak-paths also limit the maximum array size because sensing margin degrades severely as the array size increases. The sneak-path problem was previously explained in Chapter 2 and further analysed in Chapter 4.

6.5 Proposed Crossbar Gas Sensing Structures

Sensing each memristor from the basic crossbar structure without an isolating device or extra line biasing is usually plagued by sneak-path as described in Section 2.3.3. In the presence of sneak-path, each memristor in a basic crossbar array cannot be used as a sensor on its own. We propose two sneak-paths free structures without the use of an isolating device and one sneak-paths free structure that makes use of the transistor as an isolating device.

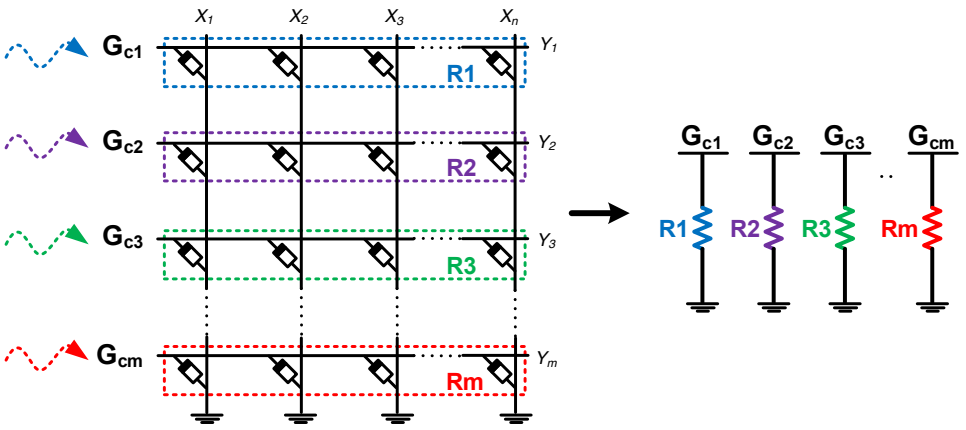


Figure 6.6: The proposed multi-sensing structure. A $m \times n$ array of multiple sensors able to detect m different gases and its equivalent resistance model.

6.5.1 Multi-Gas Sensing with m $(1 \times n)$ Sensor Array

This section proposes a crossbar structure suitable for sensing multiple gases. A matrix of memristor sensors are designed such that the memristors in the same row have identical initial properties that enables them to react in approximately similar pattern in the presence of any target gas. In this architecture, each row in the matrix acts as a sensor. The number of gases that can be sensed using this architecture depends on the number of rows in the crossbar as shown in Fig. 6.6. The number of sensing devices in each row is based on the level of redundancy or samples needed in order to accurately detect the properties of the target gas. From a physical point of view, every memristors in a target row should adsorb similar concentration of gas as they are connected by the same electrode. Fig. 6.7 shows the experimental set-up of this structure for a 4×4 sensor array using the Cadence Virtuoso simulation tool. The passage of gas concentration to the memristors were simulated using a Piece-wise Linear Voltage source (VPWL) .

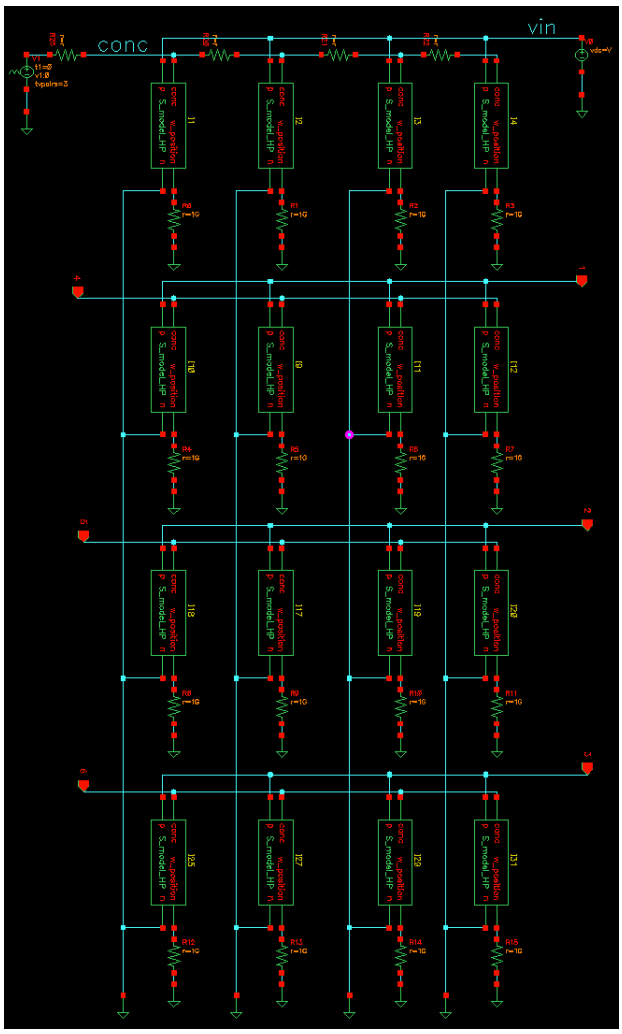


Figure 6.7: Experimental set-up of the detection schematic of a 4×4 memristor sensor array on Cadence Virtuoso simulation tool. Gas concentration is passed on to the memristors in the target row through the extra node created while their corresponding bitlines are grounded.

After exposing this architecture to a certain level of gas concentration, the response rate of the sensor can be determined using the voltage divider technique. This architecture benefits from a sneak-path free sense operation as memristors from other rows do not interfere with the sensing circuitry. The sneak-path effect in this structure is minimised by sensing all memristors in the target row simultaneously while the columns are shorted as shown in Fig. 6.8. Also, the presence of line resistances in untargeted rows does not introduce sneak-paths current into the sensing circuit but the negligible line resistances in the target row might be factored in to further improve the sensor's accuracy. The experimental set-up of this read technique is shown in Fig. 6.9 where a read voltage V_{read} is applied to the first row in the array and all the columns are shorted to the load resistance (R_L) that forms a voltage divider with the target row. All the memristors in the array are set to a resistance value of 100Ω. V_{out} is the output of the voltage divider which can be computed analytically according to Eqn. 6.6. If $R_L = 100\Omega$ and $V_{read} = 1V$, V_{out}

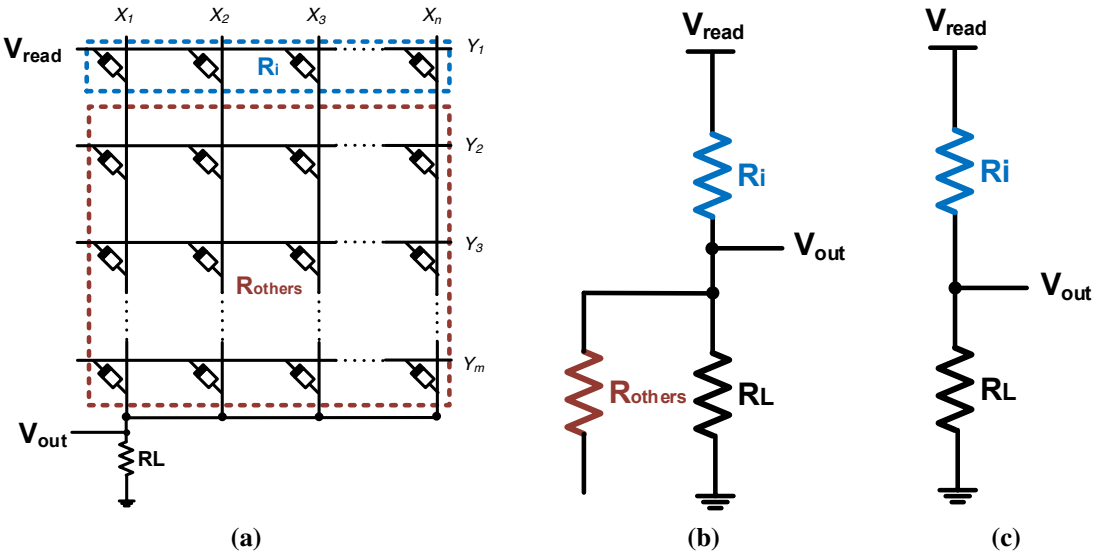


Figure 6.8: Read mechanism for the multi-sensing structure (a) Sensing one of the m sensors in the array (b) Corresponding equivalent circuit model showing the negligible impact of the other memristors in the array (c) Final equivalent circuit model without the effect of the other memristors in the array.

for this particular experiment is expected to be 0.8V and $R_i = 25\Omega$. If V_{out} is known, R_i, which is the resistance of the memristors acting as gas sensor, can be analytically calculated by solving the equivalent circuit in Fig. 6.8(c) using Eqn. 6.7. Fig. 6.10 shows the simulation result of reading the target (first row) sensor's resistance value without the effect of sneak-path currents from other memristors in the remaining rows of the array. The second waveform is the aggregate resistance of the sensor (four memristors) in the target row. This simulation method also applies to any of the m-row sensor in the crossbar array.

$$V_{out} = V_{read} \times \frac{R_L}{R_L + R_i} \tag{6.6}$$

$$R_i = \frac{V_{read} - V_{out}}{I_i} \tag{6.7}$$

Here, R_i is a parallel combination of all the resistances of the memristors in the target row ($1 \leq i \leq m$) and I_i is the current flowing through the memristors in the target row and the load resistor. No current flows through the other memristors in the array as demonstrated by the simulation result of Fig. 6.11. The three zero-current are from one randomly chosen memristor in each of the untargeted rows. This effectively demonstrates that the read technique is free of current leakages.

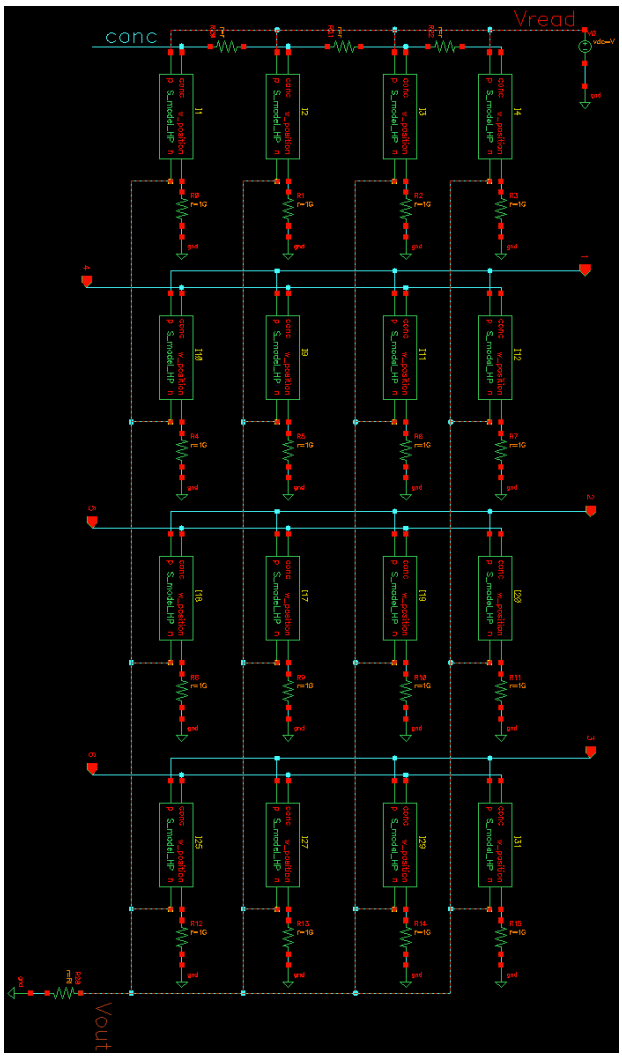


Figure 6.9: Experimental set-up of the sensing schematic of a 4×4 memristor sensor array on Cadence Virtuoso simulation tool. A voltage of V_{read} is applied to the first row while the columns are shorted and connected to the sensing circuit.

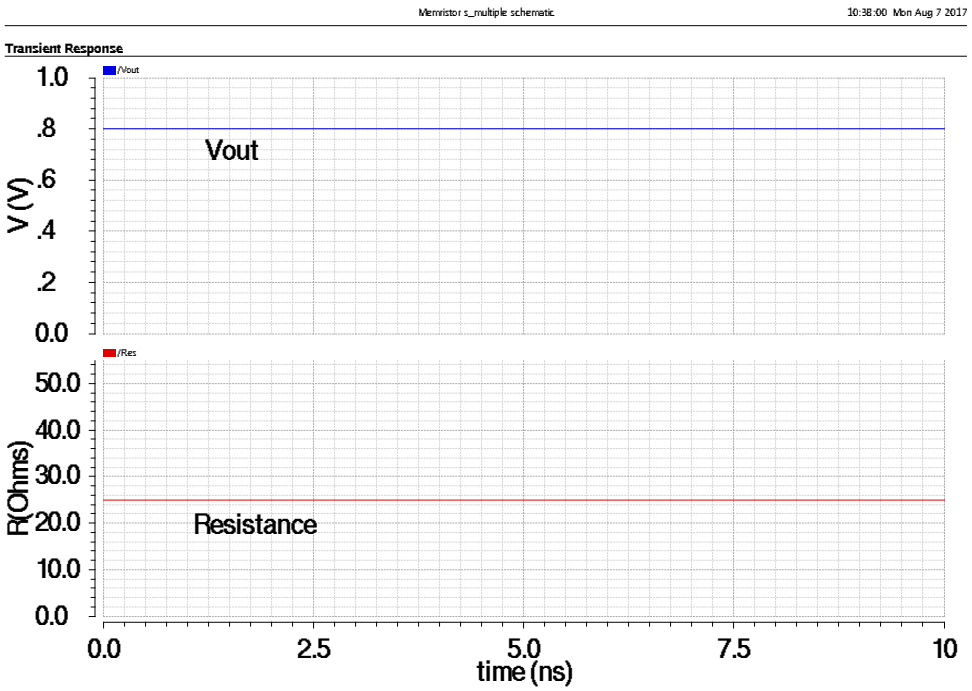


Figure 6.10: Simulation result of sensing the first row (sensor) of the 4×4 crossbar array. A V_{read} of 1V was used. V_{out} is the output of the voltage divider which can in turn be used to compute the resistance value R_i of the four parallel memristors in the first row. In this case, each of the memristors have a resistance value of 100Ω, therefore the four selected memristors have a total resistance value of 25Ω as shown in the waveform.

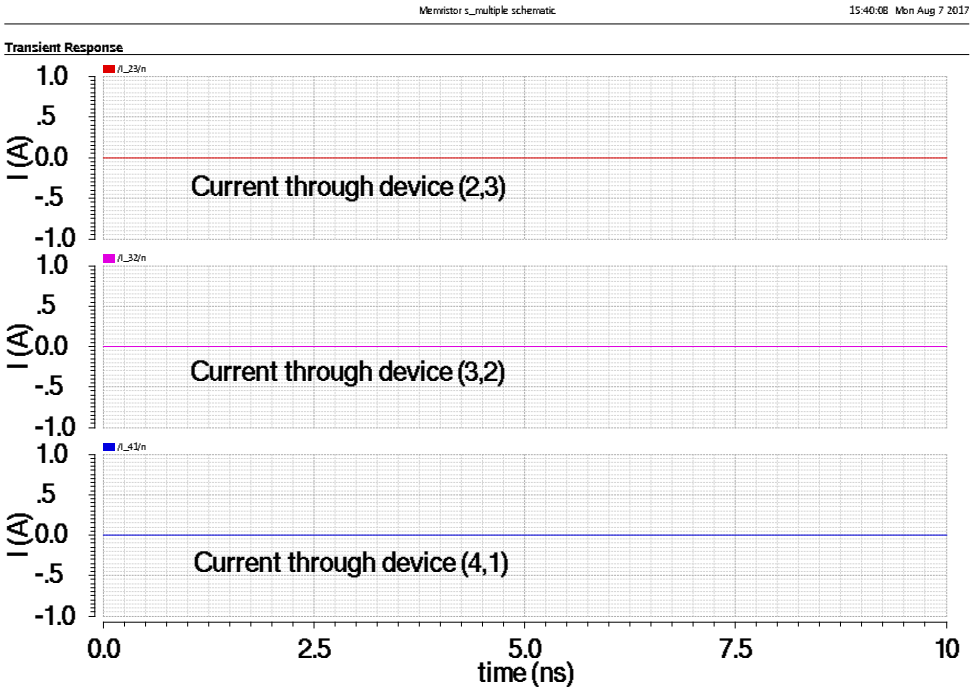


Figure 6.11: Simulation result demonstrating the zero-current flow through the unselected memristors in the $4\times$ crossbar array of Fig. 6.9.

All the memristors in a row will have approximately similar resistance values because of their identical initial conditions. Also note that when the resistance value of each row is being sensed, all other memristors in the array are shunted out of the circuit and they do not contribute to the sensed output. m-number of reads are required to detect the response of all the sensors in the $m \times n$ crossbar. Fig. 6.12 shows the simulation

result from directing a range of reducing gas concentration to a row of a 4×4 sensor array. The initial resistance value of the four memristors in the target row was set to $100\Omega, 123\Omega, 127\Omega$ and 140Ω, representing an extreme case of almost 33% variation between the smallest and largest memristance. Despite the large variation, an average response (red waveform) which represents the parallel combination of the four memristor was measured by the sensing architecture over the range of gas concentration (pink waveform). If all the rows in the crossbar are used for gas sensing, measurements will be done by a succession of m read steps for $i = 1, ..., m$ and all responses recorded accordingly. The sensing structure depicted by Fig. 6.6 also makes it possible to design a fault tolerant system by introducing a repair mechanism that addresses faulty sensors in the array. This can be achieved by checking if the response of a memristor (sensor) considerably differs from the rest, in such case, the resistance of the cell can be set to a high resistance state or such cell substituted with a spare one. A faulty memristor in a row can also be isolated by leaving it bitline floating.

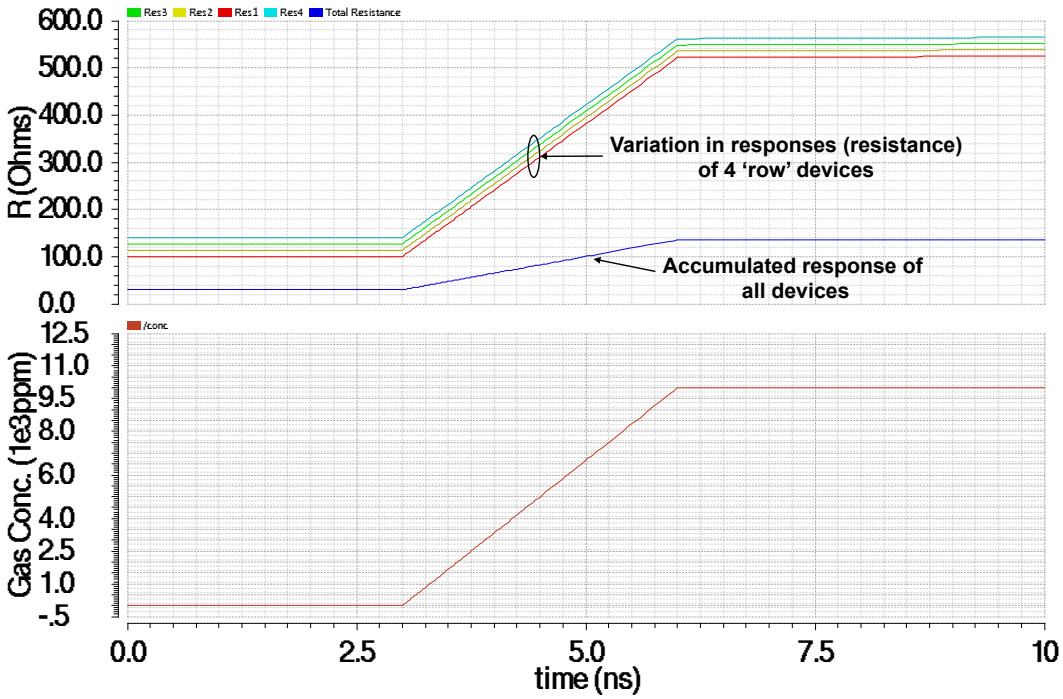


Figure 6.12: Simulation result of exposing a row from a 4×4 memristor crossbar array to a range of gas concentration (reducing gas). Resistance change waveforms at the top, Gas concentration waveform at the bottom.

As clearly shown in Fig. 6.13, the response of the row-based sensor gets better as the number of memristors in the row increases. In this simulation, a crossbar with five rows was created and each row initialised to different R_{on} values. The number of memristors in each row was progressively increased and results show an improved response as the

number of sensing memristor grows across the five rows in the crossbar. Measurements was done by a succession of m reads for $i = 1, \ldots, m$ and each of the responses are recorded.

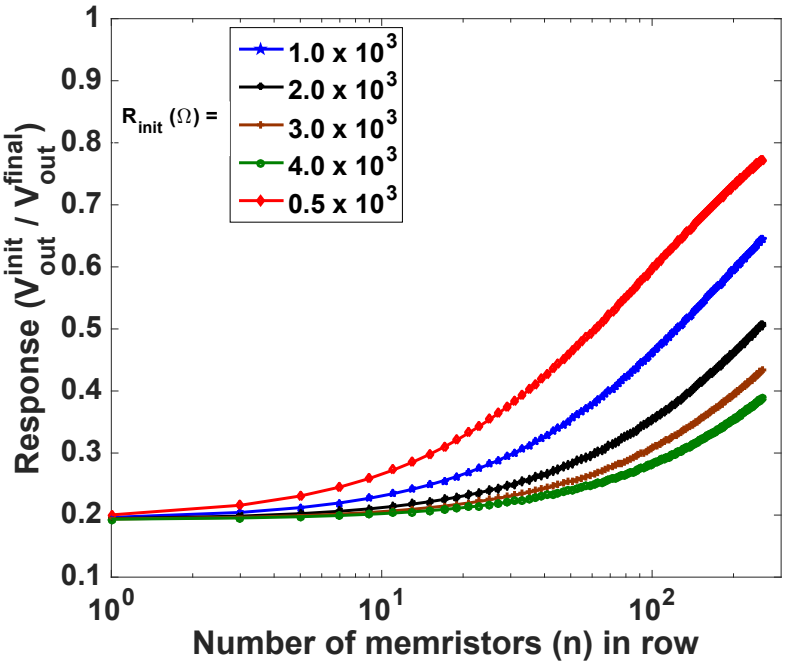


Figure 6.13: Response of five rows of memristors to gas presence. All memristors in the same row are set to the same initial resistance.

6.5.2 The m × n Array Structure

We propose a structure that consists of an $m \times n$ crossbar array of memristors such that the entire array acts as a single sensor. This is similar to the m $(1 \times n)$ array except for the number of sensors in the array. A parallel readout of the crossbar matrix could be achieved by connecting all the wordlines to the read voltage and all bitlines are grounded via the load resistor for sensing. The structure of the sensing mechanism and its equivalent circuit model are shown in Fig. 6.14(a) and 6.14(b) respectively. The sensitivity of this structure to any gas can be computed by measuring the overall resistance of memristor before and after interaction with the subject gas.

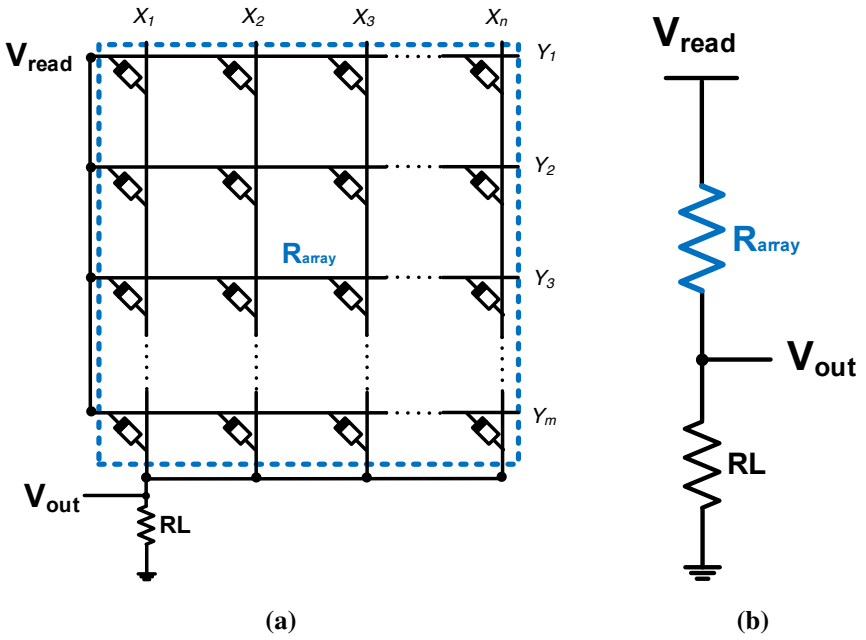


Figure 6.14: (a) Sensing mechanism for a crossbar sensor made up of $m \times n$ sensors. (b) Equivalent resistance model for the sensing mechanism.

The sensitivity measurement for a sensor that consists of an array of memristors will differ slightly from that of a single memristor shown in Fig. 6.4. This is because the ratio of the initial and final resistance of the crossbar remains constant as the size varies with fixed gas concentration. For simulation purposes, the overall output voltage of the array computed by Eqn. 6.8 will instead be used to measure sensitivity.

$$V_{out} = V_{read} \times \frac{R_L}{R_L + R_{array}} \tag{6.8}$$

where R_{array} is a parallel combination of all the memristance (R_{mem}) in the array. The simulation result of this sensing technique shows that the sensor's response improves with increasing number of memristors in parallel as shown in Fig 6.15. The rate of improvement to the sensor's responsiveness is however not linear with an increasing array size. The increment in the response rate drops to approximately 1% once the number of devices approach $5K$ and almost negligible by the time the $10,000th$ device is added to the array. Hence, an array size beyond $5K$ may not be justifiable.

6.5.3 The 1T1M Structure

The 1T1M structure is another memristor-based architecture that solves the sneak-path problem [78]. The 1T1M architecture shown in Fig. 6.16 consists of an access transistor

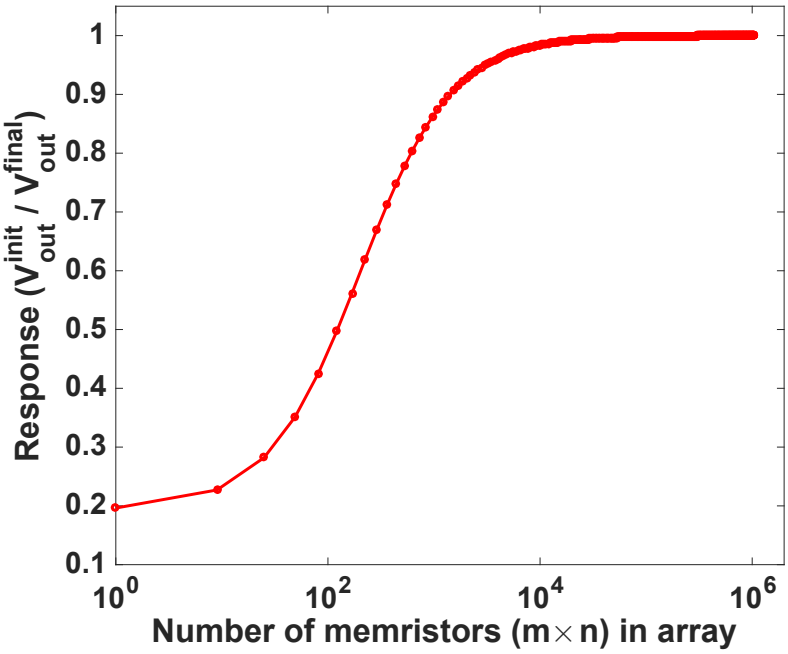


Figure 6.15: Sensor's responsiveness increases as the number of memristors in the $m \times n$ sensor structure increases. $R_{on} = 1K\,\Omega$, $R_L \ll R_{on}$, $C = 10^4\,ppm$.

and a memristor at every cross-point. The transistor enables independent access to each memristor cell without interference from other cells but at the expense of area per cell. The 1T1M structure has a bigger footprint because of the access transistor. To perform the sensing operation in this architecture, only the row that contains the target cell is enabled via the access transistor and only the corresponding column grounded. In this way, the desired sensor can be read without sneak-path and each cell can be used to independently sense different gases. Response of each memristor in the array will be similar to the simulation result of an isolated single memristor in Fig. 6.4. Clearly the 1T1M approach, which resembles the DRAM architecture, does not leverage the potential density improvement achievable in a purely memristive crossbar structure. This approach can be adopted whenever high density is less of a priority.

6.5.4 Experimental Results and Discussions

As earlier mentioned, the accuracy and reliability of gas sensors can be improved statistically by taking multiple measurements from the same sensor or from different independent sensors. This section examines the accuracy of the three proposed sensing structures. An important figure of merit is the resistance swing of a gas sensor. We define resistance swing as the window between the maximum and minimum values in the resistance dis-

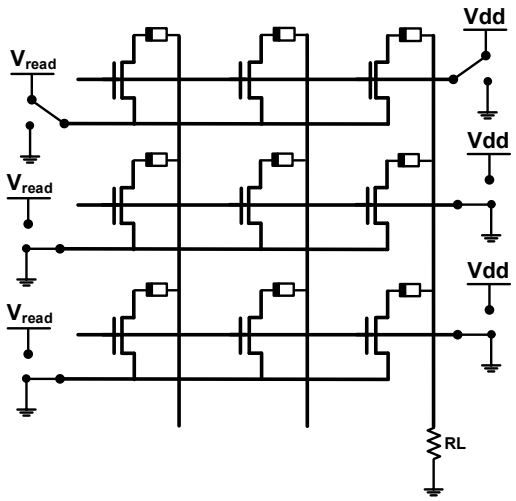


Figure 6.16: One transistor, one memristor structure for minimizing sneak-path effect in an array of sensors. The sensor in the top right corner is selected for sensing.

tribution of the sensing device after measurement. A reliable sensor will have a small resistance swing which will make it possible to detect slight changes in gas concentration. In order to estimate the resistance swing of the proposed sensing structures, we simulate multiple measurements with multiple samples. The initial and final resistance values of 5×10^4 samples of each of the sensing structures described in Section 6.5 were measured. We assume a 5% standard deviation in the initial resistance value of each memristor used in the sensor, the same standard deviation was also applied to the final resistance after 10^3 ppm volume of gas was applied. The aim of applying the deviation to the resistance values is to evaluate how well each structure can tolerate resistance variation that often affects memristive devices. Figures in Fig. 6.17 shows the Monte Carlo simulation of resistance distribution using 5×10^4 samples from each of the sensing structures described in Section 6.5. The $m \times n$ structure has the least resistance swing before and after gas exposure. The presence of more memristors in the $m \times n$ led to a desirable reduction in the standard deviation of the overall resistance of the array. The $m \times n$ structure benefits from the idea of averaging the response of multiple memristors in order to increase the sensor's reliability and precision as shown in Fig. 6.17(a). Similarly, the $m(1 \times n)$ structure has a fairly less standard deviation (Fig. 6.17(b)) compared with that of the single memristor in Fig. 6.17(c). Simulation results presented in Fig. 6.17(c), 6.17(b) and 6.17(a) were performed using a single memristor sensor cell, 1×8 and 8×8 sensor array respectively. A possible 5% standard deviation in memristance could invariably lead up to a 60% progressive variation on the final response of the sensors over a sample size of 5×10^4 single memristors. However, the same sample size with a 8×8 sensor array

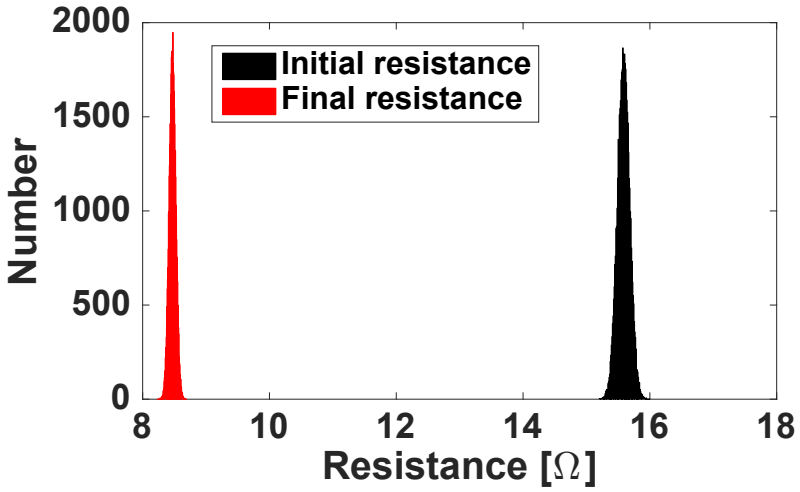
Table 6.2: Relative comparison of sensing structures

	$m \times n$	$1 \times n$	**1T1M**
Reliability	High	Moderate	Low
Sneak-path	No	No	No
Resistance Swing	Low	Moderate	High
Multi-Gas sensing	No	Yes (number of rows)	Yes (number of cells)

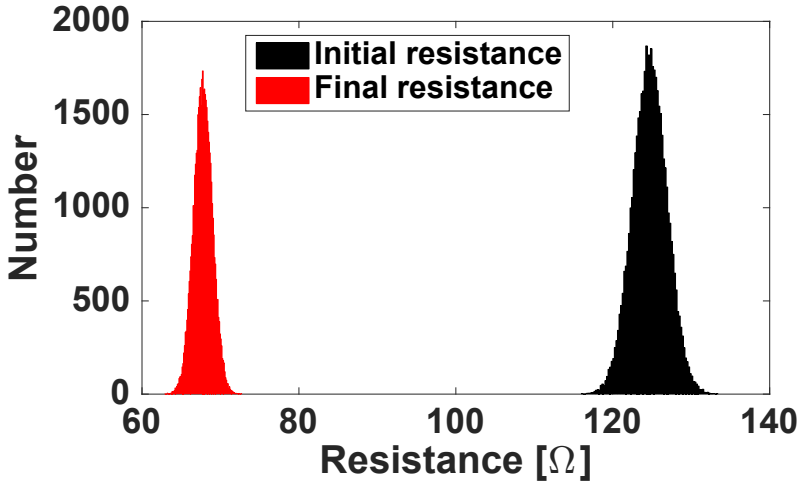
reduces the variation in the sensor responses to as low as 5.3%, thereby making the $m \times n$ structure a more reliable option due to the averaging effect of having more memristors in the sensing block. The $m \times n$ is however limited to sensing a single gas at a time unlike the $m(1 \times n)$ and 1T1M structures that can sense multiple gases depending on the number of rows (m) and number of memristors in the array respectively. The $m \times n$ structure gives the lowest resistance swing compared to the other two structures because of less variation in its measured response. Table 6.2 shows the comparison of the different proposed structures. These sensing approaches are essential for integrating very dense sensor arrays on well known and established memory-like architectures already proposed for memristive arrays. As earlier implied, memristive sensors can be lumped together to function as a single sensor, unlike memristor-based memory array, where the content of the individual device has to be explicitly measured to make sense of the data. For the purpose of gas detection, averaging responses from multiple sensing devices enhances the sensor's overall accuracy. This averaging technique is achieved via parallel combinations of all concerned devices which helps to knock out the effect of any fault within the crossbar array. Another interesting aspect of memristor's use as a sensor in crossbar architecture is the opportunity to sense multiple memristors without the sneak-path problem.

6.6 Conclusions

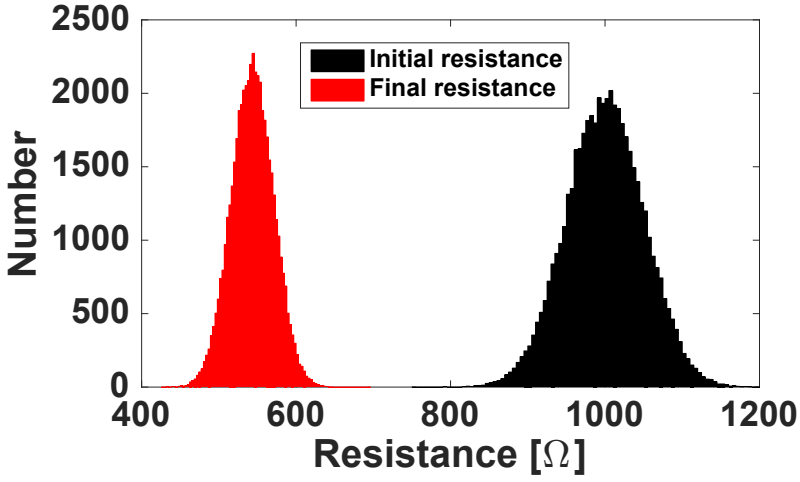
To the best of our knowledge, this is the first attempt to develop a novel sensing mechanism and architectures using memristive crossbar arrays. The contribution of this chapter starts with the development of a gas sensitive Verilog-A memristor model. This model takes into account the gas sensing properties of TiO_2 and could be expanded to other metal oxide materials. Using this model, three gas sensing techniques were proposed and investigated, namely, $m \times n$, $m(1 \times n)$ and 1T1M. Our analysis shows that the $m \times n$



(a)



(b)



(c)

Figure 6.17: Initial (before gas exposure) and final (before gas exposure) resistance distribution in (a) $m \times n$ sensing structure as explained in Section 6.5.2, (b) $1 \times n$ sensor representing one of the m sensors used for multi-gas sensor explained in Section 6.5.1 and (c) a single memristor sensor from use of the 1T1M structure. $m = 8$, $n = 8$, $C = 10^3\ ppm$.

and $m(1 \times n)$ structures are more efficient in terms of responsiveness and reliability but at the expense of the number of gases that could be sensed. On the other hand, the 1T1M

enables the deployment of high numbers of independent sensors in a single array. Simulation results show that an array of memristors can minimise measurement errors as well as provide a good redundancy measure during gas sensing. Measurements taken from the proposed sensor structures are also not affected by alternate current paths problem often experienced in crossbar architecture. We believe that the proposed analysis represents a fundamental cornerstone to the success of this novel approach to sensing: future developments include the manufacturing, analysis, and modelling of prototypes based on the proposed architectures.

The structural analysis presented in this chapter can be used in the early stages of new smart sensing system development, to simulate the overall behaviour and in identifying any vulnerabilities before any actual system is developed. If problems are identified sooner during development, then less time and cost will be required to fix those problems.

Chapter 7

Conclusions and Future work

Full commercial realisation of memristor-based architectures have been hindered largely due to some reliability issues. Memristor on its own is considered reliable and advantageous over current technologies. Full potential of memristors can fully be exploited if high density crossbar structures can be designed reliably. High density memristive structures will require reliable and well understood read and write techniques in order for them to be used as memory devices, gas detectors (presented in this thesis) and other identified application areas of the memristor. Resistive memories as an application of memristor is a promising emerging memory technology with the potential to design low power and high density memories. Memristive devices have several other potentials beyond its memory application which is why many more application fields are still being discovered till date. The chapters within this thesis extensively describe read and write operations in memristive architectures model as well as several novel circuit designs that addresses these reliability issues.

7.1 Summary of this Thesis

The work presented in this thesis has sought to extend the understanding and quality of memristor-based designs in three principal areas:

1. Detailed analysis of read operations in memristor-based crossbar arrays and proposal of reliable ways to read from memristor-based designs

    - The sneak-path current is one major problem that often lead to read failure in crossbar arrays by limiting crossbar size to a maximum of 3×3 in some

architectures. A robust solutions to this problem is difficult to come by without a proper understanding of the crossbar's design. This thesis presented an extended model of the crossbar's design and further analysis of current read schemes.

- This contribution area also proposed a multiple cells read technique as a way of minimising the effects of sneak-path current in crossbar arrays.

2. Detailed analysis of write operations in memristor-based crossbar arrays and proposal of reliable ways to execute write operations in memristor-based designs

    - Write operation in crossbar arrays is quite involved due to the fact that both the targeted and untargeted memory cells can be affected in case of a write failure. The write operation becomes even more complicated when line resistances are factored in. Line resistance often cause write voltage degradation, for instance, a resistance ratio of up to 0.01 can limit the maximum array size of a crossbar to 100×100 in some architectures. A new compensated write voltage technique is proposed in Chapter 5. This technique helps to ensure sufficient voltage reaches memory cells further away from the voltage driver.
    - A novel multiple cells write scheme is also presented in Chapter 5. This technique benefits from the reliability of the existing $V/3$ write scheme and results in a much reduced energy consumption as the crossbar array size increases. Hence, making it possible to design highly dense memory structures. This technique has the disadvantage of extra area overhead because of the control circuitry.

3. A novel application of memristor-based crossbar array in gas sensing

    - Detection and sensing is one of the recently discovered application areas of the memristor. Chapter 6 presents a novel Verilog-A memristor model suitable for designing and simulating gas sensing memristive architectures in EDA tools.
    - Proposal of gas sensing structures using insight and information obtained from working on read and write operations in crossbar arrays. Memristive gas sensors were designed with crossbar arrays and sensed without the effects of sneak-path currents.

4. Development of various parameterised simulation tools in C++ to aid the various

research contributions of work presented in this thesis (appendix B). Existing EDA tools are currently limited when it comes to working with emerging technologies especially designs that involves new devices such as memristor. These EDA tools do not have inbuilt memristor models as it does for resistors, capacitors and many others. In cases where Verilog-A models were developed to aid simulation with the EDA tools, simulation time for a 128×128 array can take up to 24hours. Also, crossbar array sizes and structures are not scaleable without the need to redesign the circuit with inbuilt parameters. All these limitations make designing and simulation of larger circuit tedious and extremely slow.

In summary, this work contributes a deeper and clearer understanding of the operations of memristor-based crossbar arrays. A detailed analyses of the crossbar's performances and limitations due to sneak-path current and line resistances and the development of improved memristive architectures and applications were all covered in this thesis.

7.2 Future Research

Although the proposed techniques in this thesis represents significant improvements to memristor-based architectures, there still remain several opportunities to improve and extend the work in this thesis. This section list some of those opportunities.

1. The multiple cells read scheme presented in Chapter 4 was designed with the Floating line read scheme. Other read schemes for single cells such as the GBFW, FBGW and GBGW can also be applied to the multiple cells read technique.

2. The compensated write voltage technique in its current state ensures sufficient write voltage reaches the target cell(s). However, some research work is still required on the control circuitry to ensure that the state of unselected cells are preserved at all times. The low power multiple cells write technique presented in Chapter 5 was designed with negligible line resistances. The presented results can be taken as input into further analysis of multiple cells write scheme with full consideration for line resistances.

3. The Verilog-A memristor model developed for gas sensing is suitable for memristor with titanium dioxide as it's switching material. This Verilog-A model requires

significant research in order to extend it to capture the behaviour of memristors with switching material other than titanium dioxide. An in-depth understanding of the chemical properties of the switching material is often required.

Bibliography

[1] **Adeyemo, Adedotun**, A. Jabir, J. Mathew, E. Martinelli, C. Di Natale, and M. Ottavi, "Efficient sensing approaches for high-density memristor sensor array," *Journal of Computational Electronics*, 2017, in-review.

[2] **Adeyemo, Adedotun**, A. Jabir, and J. Mathew, "Minimizing impact of wire resistance in low-power crossbar array write scheme," *Journal of Low Power Electronics*, vol. 13, no. 4, pp. 649–660, 2017.

[3] **Adeyemo, Adedotun**, A. Jabir, J. Mathew, E. Martinelli, C. Di Natale, and M. Ottavi, "Reliable gas sensing with memristive array," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017, pp. 244–246.

[4] **Adeyemo, Adedotun**, X. Yang, A. Bala, and A. Jabir, "Analytic models for crossbar write operation," in *Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on*, 2016, pp. 313–317.

[5] **Adeyemo, Adedotun**, X. Yang, A. Bala, J. Mathew, and A. Jabir, "Analytic models for crossbar read operation," in *IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2016, pp. 3–4.

[6] **Adeyemo, Adedotun**, J. Mathew, A. M. Jabir, and D. Pradhan, "Exploring error-tolerant low-power multiple-output read scheme for memristor-based memory arrays," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2015, pp. 17–20.

[7] **Adeyemo, Adedotun**, J. Mathew, A. Jabir, and D. Pradhan, "Write scheme for multiple complementary resistive switch (crs) cells," in *IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2014, pp. 1–5.

[8] X. Yang, **Adeyemo, Adedotun**, A. Bala, and A. Jabir, "Parasitic effects on memristive logic architecture," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2017, pp. 1–5.

[9] X. Yang, **Adeyemo, Adedotun**, A. Bala, and A. Jabir, "Novel techniques for memristive multifunction logic design," *The VLSI Journal on Integration*, 2017.

[10] X. Yang, **Adeyemo, Adedotun**, A. Jabir, and J. Mathew, "High-performance single-cycle memristive multifunction logic architecture," *Electronics Letters*, vol. 52, no. 11, pp. 906–907, 2016.

[11] A. Bala, **Adeyemo, Adedotun**, X. Yang, and A. Jabir, "High level abstraction of memristor model for neural network simulation," in *Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on*, 2016, pp. 318–322.

[12] X. Yang, **Adeyemo, Adedotun**, A. Bala, and A. Jabir, "Novel memristive logic architectures," in *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016 26th International Workshop on*, 2016, pp. 196–199.

[13] **Adeyemo, Adedotun**, A. Jabir, J. Mathew, E. Martinelli, C. Di Natale, and M. Ottavi, *Memristive sensor array*, UK Patent App. 1616837.9, Oct. 2016.

[14] A. Jabir, **Adeyemo, Adedotun**, and X. Yang, *Memristive multifunction logic architecture*, UK Patent App. 1603089.2, Feb. 2016.

[15] G. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, no. 8, 1965.

[16] H.-S. P. Wong, L. Wei, and J. Deng, "The future of cmos scaling-parasitics engineering and device footprint scaling," in *Solid-State and Integrated-Circuit Technology, 2008. ICSICT 2008. 9th International Conference on*, 2008, pp. 21–24.

[17] *ITRS Report*, 2016. [Online]. Available: http://www.itrs2.net/itrs-reports.html.

[18] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker, "Logic circuits with carbon nanotube transistors," *Science*, vol. 294, no. 5545, pp. 1317–1320, 2001.

[19] O. Y. Loh and H. D. Espinosa, "Nanoelectromechanical contact switches," *Nature nanotechnology*, vol. 7, no. 5, pp. 283–295, 2012.

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

[21] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 24–33, 2009.

[22] W. J. Gallagher and S. S. Parkin, "Development of the magnetic tunnel junction mram at ibm: From first junctions to a 16-mb mram demonstrator chip," *IBM Journal of Research and Development*, vol. 50, no. 1, pp. 5–23, 2006.

[23] S. Parkin, K. Roche, M. Samant, P. Rice, R. Beyers, R. Scheuerlein, E. Osullivan, S. Brown, J. Bucchigano, D. Abraham, *et al.*, "Exchange-biased magnetic tunnel junctions and application to nonvolatile magnetic random access memory," *Journal of Applied Physics*, vol. 85, no. 8, pp. 5828–5833, 1999.

[24] H. Kohlstedt, Y. Mustafa, A. Gerber, A. Petraru, M. Fitsilis, R. Meyer, U. Böttger, and R. Waser, "Current status and challenges of ferroelectric memory devices," *Microelectronic Engineering*, vol. 80, pp. 296–304, 2005.

[25] S.-Y. Wu, "A new ferroelectric memory device, metal-ferroelectric-semiconductor transistor," *Electron Devices, IEEE Transactions on*, vol. 21, no. 8, pp. 499–504, 1974.

[26] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, *et al.*, "A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-ram," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, 2005, pp. 459–462.

[27] D. Brooks, "Reliable system design in the era of specialization," in *Keynote Talk, IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2015.

[28] J. Hutchby and M. Garner, "Assessment of the potential and maturity of selected emerging research memory technologies," *International Technology Roadmap for Semiconductors*, 2010.

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

[30] D. B. Strukov *et al.*, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[31] F. Miao, J. P. Strachan, J. J. Yang, M.-X. Zhang, I. Goldfarb, A. C. Torrezan, P. Eschbach, R. D. Kelley, G. Medeiros-Ribeiro, and R. S. Williams, "Anatomy of a nanoscale conduction channel reveals the mechanism of a high-performance memristor," *Advanced Materials*, vol. 23, no. 47, pp. 5633–5640, 2011.

[32] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie, "Design implications of memristor-based RRAM cross-point structures," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, 2011, pp. 1–6.

[33] S.-k. Park. (2011). The future memory technologies. Available at http://www.sematech.org/meetings/archives/symposia/10202/Keynote-Intro/Park_The%20future%20memory%20technologies.pdf [Online; accessed 2017-05-14].

[34] M. Durcan. (2016). Industry environment and micron's strategic priorities. Available at http://files.shareholder.com/downloads/ABEA-45YXOQ/2376125726x0x875021/4BEAA02E-BBC2-402C-A51D-B3B2C6B8C3D4/Winter_Analyst_Day_2016.pdf [Online; accessed 2017-05-14].

[35] D. Niu, Y. Chen, C. Xu, and Y. Xie, "Impact of process variations on emerging memristor," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 877–882.

[36] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *ACM SIGARCH Computer Architecture News*, ACM, vol. 38, 2010, pp. 141–152.

[37] C.-M. Jung, J.-M. Choi, and K.-S. Min, "Two-step write scheme for reducing sneak-path leakage in complementary memristor array," *Nanotechnology, IEEE Transactions on*, vol. 11, no. 3, pp. 611–618, 2012.

[38] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature materials*, vol. 9, no. 5, pp. 403–406, 2010.

[39] ITRS. (2013). Executive summary. Available at http://www.itrs2.net/itrs-reports.html [Online; accessed 2017-05-14].

[40] G. K. Johnsen, "An introduction to the memristor-a valuable circuit element in bioelectricity and bioimpedance," *Journal of Electrical Bioimpedance*, vol. 3, no. 1, pp. 20–28, 2012.

[41] L. O. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976.

[42] L. Chua, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011.

[43] R. Williams, "How we found the missing memristor," *Spectrum, IEEE*, vol. 45, no. 12, pp. 28–35, 2008.

[44] D. L. Lewis and H.-H. Lee, "Architectural evaluation of 3D stacked RRAM caches," in *3D System Integration, 2009. 3DIC 2009. IEEE International Conference on*, 2009, pp. 1–4.

[45] W. Fei, H. Yu, W. Zhang, and K. S. Yeo, "Design exploration of hybrid cmos and memristor circuit by new modified nodal analysis," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 6, pp. 1012–1025, 2012.

[46] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[47] D. B. Strukov, D. R. Stewart, J. Borghetti, X. Li, M. Pickett, G. M. Ribeiro, W. Robinett, G. Snider, J. P. Strachan, W. Wu, *et al.*, "Hybrid CMOS/memristor circuits," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 1967–1970.

[48] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *IEEE International Symposium on Nanoscale Architectures*, 2008, pp. 85–92.

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

[50] S. Shin, K. Kim, and S.-M. Kang, "Reconfigurable stateful nor gate for large-scale logic-array integrations," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 7, pp. 442–446, 2011.

[51] 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.

[52] F. Puppo, A. Dave, M.-A. Doucey, D. Sacchetto, C. Baj-Rossi, Y. Leblebici, G. De Micheli, and S. Carrara, "Memristive biosensors under varying humidity conditions," *NanoBioscience, IEEE Transactions on*, vol. 13, no. 1, pp. 19–30, 2014.

[53] S. Carrara *et al.*, "Memristive-biosensors: A new detection method by using nanofabricated memristors," *Sensors and Actuators B: Chemical*, vol. 171, pp. 449–457, 2012.

[54] C. Nyenke and L. Dong, "Fabrication of a $W/Cu_xO/Cu$ memristor with submicron holes for passive sensing of oxygen," *Microelectronic Engineering*, vol. 164, pp. 48–52, 2016.

[55] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008.

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

[57] Z. Biolek, D. Biolek, and V. Biolkova, "Spice model of memristor with nonlinear dopant drift," *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.

[58] S. Kvatinsky, K. Talisveyberg, D. Fliter, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Models of memristors for spice simulations," in *IEEE 27th Convention of Electrical & Electronics Engineers in Israel (IEEEI)*, 2012, pp. 1–5.

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

[60] J. J. Yang, M.-X. Zhang, M. D. Pickett, F. Miao, J. P. Strachan, W.-D. Li, W. Yi, D. A. Ohlberg, B. J. Choi, W. Wu, *et al.*, "Engineering nonlinearity into memristors for passive crossbar applications," *Applied Physics Letters*, vol. 100, no. 11, p. 113 501, 2012.

[61] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, no. 7, p. 074 508, 2009.

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

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

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

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

[66] R. Rosezin, E. Linn, L. Nielen, C. Kugeler, R. Bruchhaus, and R. Waser, "Integrated complementary resistive switches for passive high-density nanocrossbar arrays," *Electron Device Letters, IEEE*, vol. 32, no. 2, pp. 191–193, 2011.

[67] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of memristor-based memories," in *12th International Conference on Embedded Systems, 2013 26th International Conference on VLSID*, 2013, pp. 386–391.

[68] A. Flocke and T. G. Noll, "Fundamental analysis of resistive nano-crossbars for the use in hybrid nano/CMOS-memory," in *Solid State Circuits Conference, 2007. ESSCIRC 2007.*, 2007, pp. 328–331.

[69] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[70] J. Mustafa, "Design and analysis of future memories based on switchable resistive elements," PhD thesis, Universitätsbibliothek, 2006.

[71] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *Electron Devices, IEEE Transactions on*, vol. 60, no. 4, pp. 1318–1326, Apr. 2013, ISSN: 0018-9383.

[72] D. Niu, C. Xu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Design trade-offs for high density cross-point resistive memory," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ACM, 2012, pp. 209–214.

[73] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the challenges of crossbar resistive memory architectures," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, 2015, pp. 476–488.

[74] S. Shin, K. Kim, and S. Kang, "Analysis of passive memristive devices array: Data-dependent statistical model and self-adaptable sense resistance for RRAMs," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2021–2032, 2012.

[75] H. Manem and G. S. Rose, "A read-monitored write circuit for 1t1m multi-level memristor memories," in *Circuits and systems (ISCAS), 2011 IEEE international symposium on*, 2011, pp. 2938–2941.

[76] H. Manem, G. S. Rose, X. He, and W. Wang, "Design considerations for variation tolerant multilevel CMOS/Nano memristor memory," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, ACM, 2010, pp. 287–292.

[77] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2013.

[78] M. Zangeneh and A. Joshi, "Performance and energy models for memristor-based 1T1R RRAM cell," in *Proceedings of the great lakes symposium on VLSI*, ACM, 2012, pp. 9–14.

[79] B. Widrow *et al.*, *Adaptive" adaline" neuron using chemical" memistors."*. 1960.

[80] M. Zidan, H. Omran, A. Sultan, H. Fahmy, and K. Salama, "Compensated readout for high-density mos-gated memristor crossbar array," *Nanotechnology, IEEE Transactions on*, vol. 14, no. 1, pp. 3–6, Jan. 2015, ISSN: 1536-125X.

[81] M. A. Zidan, A. M. Eltawil, F. Kurdahi, H. A. Fahmy, and K. N. Salama, "Memristor multiport readout: A closed-form solution for sneak paths," *Nanotechnology, IEEE Transactions on*, vol. 13, no. 2, pp. 274–282, 2014.

[82] P. O. Vontobel, W. Robinett, P. J. Kuekes, D. R. Stewart, J. Straznicky, and R. S. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, p. 425 204, 2009.

[83] A. Chen, "Accessibility of nano-crossbar arrays of resistive switching devices," in *2011 11th IEEE Conference on Nanotechnology (IEEE-NANO)*, 2011, pp. 1767–1771.

[84] V. Bochenkov and G. Sergeev, "Sensitivity, selectivity, and stability of gas-sensitive metal-oxide nanostructures," *Metal Oxide Nanostructures and Their Applications*, vol. 3, pp. 31–52, 2010.

[85] N. Yamazoe, "New approaches for improving semiconductor gas sensors," *Sensors and Actuators B: Chemical*, vol. 5, no. 1, pp. 7–19, 1991.

[86] D. E. Williams, "Semiconducting oxides as gas-sensitive resistors," *Sensors and Actuators B: Chemical*, vol. 57, no. 1, pp. 1–16, 1999.

[87] P. Clifford and D. Tuma, "Characteristics of semiconductor gas sensors I. steady state gas response," *Sensors and Actuators*, vol. 3, pp. 233–254, 1982.

[88] G. F. Fine *et al.*, "Metal oxide semi-conductor gas sensors in environmental monitoring," *Sensors*, vol. 10, no. 6, pp. 5469–5502, 2010.

[89] A. DÁmico and C. Di Natale, "A contribution on some basic definitions of sensors properties," *IEEE Sensors Journal*, vol. 1, no. 3, pp. 183–190, 2001.

[90] T. Plecenik, M. Moško, A. Haidry, P. Ďurina, M. Truchlỳ, B. Grančič, M. Gregor, T. Roch, L. Satrapinskyy, A. Mošková, *et al.*, "Fast highly-sensitive room-temperature semiconductor gas sensor based on the nanoscale $Pt - TiO_2 - Pt$ sandwich," *Sensors and Actuators B: Chemical*, vol. 207, pp. 351–361, 2015.

[91] S. Vallejos, I. Grácia, O. Chmela, E. Figueras, J. Hubálek, and C. Cané, "Chemoresistive micromachined gas sensors based on functionalized metal oxide nanowires: Performance and reliability," *Sensors and Actuators B: Chemical*, vol. 235, pp. 525–534, 2016.

[92] A. Kay and M. Grätzel, "Low cost photovoltaic modules based on dye sensitized nanocrystalline titanium dioxide and carbon powder," *Solar Energy Materials and Solar Cells*, vol. 44, no. 1, pp. 99–117, 1996.

[93] A. Fujishima, T. N. Rao, and D. A. Tryk, "Titanium dioxide photocatalysis," *Journal of Photochemistry and Photobiology C: Photochemistry Reviews*, vol. 1, no. 1, pp. 1–21, 2000.

[94] C. Garzella, E. Comini, E. Tempesti, C. Frigeri, and G. Sberveglieri, "TiO_2 thin films by a novel sol–gel processing for gas sensor applications," *Sensors and Actuators B: Chemical*, vol. 68, no. 1, pp. 189–196, 2000.

[95] Q. Wang, Y. Pan, S. Huang, S. Ren, P. Li, and J. Li, "Resistive and capacitive response of nitrogen-doped TiO_2 nanotubes film humidity sensor," *Nanotechnology*, vol. 22, no. 2, p. 025 501, 2011.

[96] C.-Y. Lin, D.-Y. Lee, S.-Y. Wang, C.-C. Lin, and T.-Y. Tseng, "Effect of thermal treatment on resistive switching characteristics in $Pt/Ti/Al_2O_3/Pt$ devices," *Surface and Coatings Technology*, vol. 203, no. 5, pp. 628–631, 2008.

[97] H. Lv, M. Wang, H. Wan, Y. Song, W. Luo, P. Zhou, T. Tang, Y. Lin, R. Huang, S. Song, *et al.*, "Endurance enhancement of cu-oxide based resistive switching memory with al top electrode," *Applied Physics Letters*, vol. 94, no. 21, p. 213 502, 2009.

[98] A. Mehonic, S. Cueff, M. Wojdak, S. Hudziak, O. Jambois, C. Labbé, B. Garrido, R. Rizk, and A. J. Kenyon, "Resistive switching in silicon suboxide films," *Journal of Applied Physics*, vol. 111, no. 7, p. 074 507, 2012.

[99] A. A. Haidry, A. Ebach-Stahl, and B. Saruhan, "Effect of Pt/TiO_2 interface on room temperature hydrogen sensing performance of memristor type $Pt/TiO_2/Pt$ structure," *Sensors and Actuators B: Chemical*, 2017.

[100] Y.-C. Chen, C. Chen, C. Chen, J. Yu, S. Wu, S. Lung, R. Liu, and C.-Y. Lu, "An access-transistor-free (0T/1R) non-volatile resistance random access memory (RRAM) using a novel threshold switching, self-rectifying chalcogenide device," in *Electron Devices Meeting, 2003. IEDM'03 Technical Digest. IEEE International*, 2003, pp. 37–4.

[101] Y. Yang, J. Mathew, R. A. Shafik, and D. K. Pradhan, "Verilog-A based effective complementary resistive switch model for simulations and analysis," *Embedded Systems Letters, IEEE*, vol. 6, no. 1, pp. 12–15, 2014.

[102] H. Mostafa and Y. Ismail, "Statistical yield improvement under process variations of multi-valued memristor-based memories," *Microelectronics Journal*, vol. 51, pp. 46–57, 2016.

[103] M. A. Lastras-Montano, A. Ghofrani, and K.-T. Cheng, "Hreram: A hybrid reconfigurable resistive random-access memory," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, EDA Consortium, 2015, pp. 1299–1304.

[104] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[105] C.-W. S. Yeh and S. S. Wong, "Compact One-Transistor-N-RRAM array architecture for advanced cmos technology," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 5, pp. 1299–1309, 2015.

[106] C. Yakopcic, T. M. Taha, and R. Hasan, "Hybrid crossbar architecture for a memristor based memory," in *Aerospace and Electronics Conference, NAECON 2014-IEEE National*, 2014, pp. 237–242.

[107] D. Williams, G. Henshaw, K. Pratt, and R. Peat, "Reaction–diffusion effects and systematic design of gas-sensitive resistors based on semiconducting oxides," *Journal of the Chemical Society, Faraday Transactions*, vol. 91, no. 23, pp. 4299–4307, 1995.

[108] J. Bai and B. Zhou, "Titanium dioxide nanomaterials for sensor applications," *Chemical reviews*, vol. 114, no. 19, pp. 10 131–10 176, 2014.

[109] H.-J. Kim and J.-H. Lee, "Highly sensitive and selective gas sensors using p-type oxide semiconductors: Overview," *Sensors and Actuators B: Chemical*, vol. 192, pp. 607–627, 2014.

[110] N. Barsan, C. Simion, T. Heine, S. Pokhrel, and U. Weimar, "Modeling of sensing and transduction for p-type semiconducting metal oxide based gas sensors," *Journal of Electroceramics*, vol. 25, no. 1, pp. 11–19, 2010.

[111] E. Linn, S. Menzel, R. Rosezin, U. Böttger, R. Bruchhaus, and R. Waser, "Modeling complementary resistive switches by nonlinear memristive systems," in *11th IEEE Conference on Nanotechnology (IEEE-NANO)*, 2011, pp. 1474–1478.

[112] S. Naisbitt *et al.*, "A microstructural model of semiconducting gas sensor response: The effects of sintering temperature on the response of chromium titanate (CTO) to carbon monoxide," *Sensors and Actuators B: Chemical*, vol. 114, no. 2, pp. 969–977, 2006.

[113] R. Binions *et al.*, "Zeolite-modified discriminating gas sensors," *Journal of The Electrochemical Society*, vol. 156, no. 3, J46–J51, 2009.

[114] D. Batas and H. Fiedler, "A memristor spice implementation and a new approach for magnetic flux-controlled memristor modeling," *IEEE Transactions on Nanotechnology*, vol. 10, no. 2, pp. 250–255, 2011.

[115] S. Kim, S.-J. Kim, K. M. Kim, S. R. Lee, M. Chang, E. Cho, Y.-B. Kim, C. J. Kim, U.-I. Chung, and I.-K. Yoo, "Physical electro-thermal model of resistive switching in bi-layered resistance-change memory," *Scientific reports*, vol. 3, 2013.

[116] J. Fraden, *Handbook of modern sensors: Physics, designs, and applications*. Springer Science & Business Media, 2004.

[117] H. Sundgren, I. Lundström, F. Winquist, I. Lukkari, R. Carlsson, and S. Wold, "Evaluation of a multiple gas mixture with a simple mosfet gas sensor array and pattern recognition," *Sensors and Actuators B: Chemical*, vol. 2, no. 2, pp. 115–123, 1990.

[118] E. J. Wolfrum, R. M. Meglen, D. Peterson, and J. Sluiter, "Metal oxide sensor arrays for the detection, differentiation, and quantification of volatile organic compounds at sub-parts-per-million concentration levels," *Sensors and Actuators B: Chemical*, vol. 115, no. 1, pp. 322–329, 2006.

Appendix A

Experimental Data for Read-out Voltages and Read Margin

Table A.1: Data for possible cases of read-out voltages across the four read schemes with varying array aspect ratio. Referenced in section 4.4.

m	*n*	FWFB				GBGB				GRFC				FWGB			
		V_{out_wc1}	V_{out_bc1}	V_{out_wc0}	V_{out_bc0}	V_{out_wc1}	V_{out_bc1}	V_{out_wc0}	V_{out_bc0}	V_{out_wc1}	V_{out_bc1}	V_{out_wc0}	V_{out_bc0}	V_{out_wc1}	V_{out_bc1}	V_{out_wc0}	V_{out_bc0}
Single		0.5000	0.5000	0.0005	0.0005	0.5000	0.5000	0.0005	0.0005	0.5000	0.5000	0.0005	0.0005	0.5000	0.5000	0.0005	0.0005
2	**2**	0.5714	0.5000	0.2503	0.0007	0.3333	0.4999	0.0002	0.0005	0.3333	0.4999	0.0002	0.0005	0.4000	0.4999	0.0003	0.0005
2	**4**	0.6154	0.5001	0.3752	0.0008	0.3333	0.4999	0.0002	0.0005	0.2000	0.4996	0.0001	0.0005	0.3636	0.4999	0.0003	0.0005
2	**8**	0.6400	0.5001	0.4377	0.0009	0.3333	0.4999	0.0002	0.0005	0.1111	0.4991	0.0001	0.0005	0.3478	0.4999	0.0003	0.0005
2	**16**	0.6531	0.5001	0.4689	0.0009	0.3333	0.4999	0.0002	0.0005	0.0588	0.4981	0.0000	0.0005	0.3404	0.4999	0.0003	0.0005
2	**32**	0.6598	0.5001	0.4845	0.0010	0.3333	0.4999	0.0002	0.0005	0.0303	0.4962	0.0000	0.0005	0.3368	0.4999	0.0003	0.0005
2	**64**	0.6632	0.5001	0.4923	0.0010	0.3333	0.4999	0.0002	0.0005	0.0154	0.4922	0.0000	0.0005	0.3351	0.4999	0.0003	0.0005
4	**2**	0.6154	0.5001	0.3752	0.0008	0.2000	0.4996	0.0001	0.0005	0.3333	0.4999	0.0002	0.0005	0.2857	0.4998	0.0002	0.0005
4	**4**	0.6957	0.5002	0.5626	0.0011	0.2000	0.4996	0.0001	0.0005	0.2000	0.4996	0.0001	0.0005	0.2353	0.4997	0.0002	0.0005
4	**8**	0.7442	0.5002	0.6563	0.0015	0.2000	0.4996	0.0001	0.0005	0.1111	0.4991	0.0001	0.0005	0.2162	0.4997	0.0001	0.0005
4	**16**	0.7711	0.5003	0.7032	0.0017	0.2000	0.4996	0.0001	0.0005	0.0588	0.4981	0.0000	0.0005	0.2078	0.4996	0.0001	0.0005
4	**32**	0.7853	0.5003	0.7266	0.0018	0.2000	0.4996	0.0001	0.0005	0.0303	0.4962	0.0000	0.0005	0.2038	0.4996	0.0001	0.0005
4	**64**	0.7926	0.5004	0.7383	0.0019	0.2000	0.4996	0.0001	0.0005	0.0154	0.4922	0.0000	0.0005	0.2019	0.4996	0.0001	0.0005
8	**2**	0.6400	0.5001	0.4377	0.0009	0.1111	0.4991	0.0001	0.0005	0.3333	0.4999	0.0002	0.0005	0.1818	0.4996	0.0001	0.0005
8	**4**	0.7442	0.5002	0.6563	0.0015	0.1111	0.4991	0.0001	0.0005	0.2000	0.4996	0.0001	0.0005	0.1379	0.4993	0.0001	0.0005
8	**8**	0.8101	0.5004	0.7657	0.0021	0.1111	0.4991	0.0001	0.0005	0.1111	0.4991	0.0001	0.0005	0.1231	0.4992	0.0001	0.0005
8	**16**	0.8477	0.5006	0.8203	0.0028	0.1111	0.4991	0.0001	0.0005	0.0588	0.4981	0.0000	0.0005	0.1168	0.4992	0.0001	0.0005
8	**32**	0.8678	0.5007	0.8477	0.0033	0.1111	0.4991	0.0001	0.0005	0.0303	0.4962	0.0000	0.0005	0.1139	0.4992	0.0001	0.0005
8	**64**	0.8782	0.5008	0.8613	0.0036	0.1111	0.4991	0.0001	0.0005	0.0154	0.4922	0.0000	0.0005	0.1125	0.4991	0.0001	0.0005
16	**2**	0.6531	0.5001	0.4689	0.0009	0.0588	0.4981	0.0000	0.0005	0.3333	0.4999	0.0002	0.0005	0.1053	0.4991	0.0001	0.0005
16	**4**	0.7711	0.5003	0.7032	0.0017	0.0588	0.4981	0.0000	0.0005	0.2000	0.4996	0.0001	0.0005	0.0755	0.4986	0.0000	0.0005
16	**8**	0.8477	0.5006	0.8203	0.0028	0.0588	0.4981	0.0000	0.0005	0.1111	0.4991	0.0001	0.0005	0.0661	0.4984	0.0000	0.0005
16	**16**	0.8920	0.5009	0.8789	0.0041	0.0588	0.4981	0.0000	0.0005	0.0588	0.4981	0.0000	0.0005	0.0623	0.4982	0.0000	0.0005
16	**32**	0.9159	0.5012	0.9082	0.0054	0.0588	0.4981	0.0000	0.0005	0.0303	0.4962	0.0000	0.0005	0.0605	0.4982	0.0000	0.0005
16	**64**	0.9284	0.5015	0.9229	0.0064	0.0588	0.4981	0.0000	0.0005	0.0154	0.4922	0.0000	0.0005	0.0596	0.4982	0.0000	0.0005
32	**2**	0.6598	0.5001	0.4845	0.0010	0.0303	0.4962	0.0000	0.0005	0.3333	0.4999	0.0002	0.0005	0.0571	0.4981	0.0000	0.0005
32	**4**	0.7853	0.5003	0.7266	0.0018	0.0303	0.4962	0.0000	0.0005	0.2000	0.4996	0.0001	0.0005	0.0396	0.4971	0.0000	0.0005
32	**8**	0.8678	0.5007	0.8477	0.0033	0.0303	0.4962	0.0000	0.0005	0.1111	0.4991	0.0001	0.0005	0.0343	0.4966	0.0000	0.0005
32	**16**	0.9159	0.5012	0.9082	0.0054	0.0303	0.4962	0.0000	0.0005	0.0588	0.4981	0.0000	0.0005	0.0322	0.4964	0.0000	0.0005
32	**32**	0.9420	0.5019	0.9385	0.0081	0.0303	0.4962	0.0000	0.0005	0.0303	0.4962	0.0000	0.0005	0.0312	0.4963	0.0000	0.0005
32	**64**	0.9557	0.5026	0.9536	0.0107	0.0303	0.4962	0.0000	0.0005	0.0154	0.4922	0.0000	0.0005	0.0308	0.4962	0.0000	0.0005
64	**2**	0.6632	0.5001	0.4923	0.0010	0.0154	0.4922	0.0000	0.0005	0.3333	0.4999	0.0002	0.0005	0.0299	0.4961	0.0000	0.0005
64	**4**	0.7926	0.5004	0.7383	0.0019	0.0154	0.4922	0.0000	0.0005	0.2000	0.4996	0.0001	0.0005	0.0203	0.4942	0.0000	0.0005
64	**8**	0.8782	0.5008	0.8613	0.0036	0.0154	0.4922	0.0000	0.0005	0.1111	0.4991	0.0001	0.0005	0.0175	0.4932	0.0000	0.0005
64	**16**	0.9284	0.5015	0.9229	0.0064	0.0154	0.4922	0.0000	0.0005	0.0588	0.4981	0.0000	0.0005	0.0164	0.4927	0.0000	0.0005
64	**32**	0.9557	0.5026	0.9536	0.0107	0.0154	0.4922	0.0000	0.0005	0.0303	0.4962	0.0000	0.0005	0.0159	0.4925	0.0000	0.0005
64	**64**	0.9699	0.5039	0.9690	0.0159	0.0154	0.4922	0.0000	0.0005	0.0154	0.4922	0.0000	0.0005	0.0156	0.4924	0.0000	0.0005

Table A.2: Data for possible cases of read margin across the four read schemes with varying array aspect ratio. Referenced in section 4.4.

m	*n*	FWFB				GBGB				GRFC				FWGB			
		ΔV_{out_c1}	ΔV_{out_c2}	ΔV_{out_c3}	ΔV_{out_c4}	ΔV_{out_c1}	ΔV_{out_c2}	ΔV_{out_c3}	ΔV_{out_c4}	ΔV_{out_c1}	ΔV_{out_c2}	ΔV_{out_c3}	ΔV_{out_c4}	ΔV_{out_c1}	ΔV_{out_c2}	ΔV_{out_c3}	ΔV_{out_c4}
Single		0.5000	0.5000	0.0005	0.0005	0.5000	0.5000	0.0005	0.0005	0.5000	0.5000	0.0005	0.0005	0.5000	0.5000	0.0005	0.0005
2	**2**	0.3211	0.5708	0.2498	0.4994	0.3331	0.3328	0.4996	0.4994	0.3331	0.3328	0.4996	0.4994	0.3997	0.3995	0.4996	0.4994
2	**4**	0.2402	0.6146	0.1249	0.4993	0.3331	0.3328	0.4996	0.4994	0.1999	0.1995	0.4995	0.4991	0.3634	0.3631	0.4996	0.4994
2	**8**	0.2023	0.6391	0.0624	0.4992	0.3331	0.3328	0.4996	0.4994	0.1110	0.1106	0.4991	0.4986	0.3476	0.3473	0.4996	0.4994
2	**16**	0.1842	0.6521	0.0312	0.4992	0.3331	0.3328	0.4996	0.4994	0.0588	0.0583	0.4981	0.4976	0.3402	0.3399	0.4996	0.4994
2	**32**	0.1753	0.6588	0.0156	0.4991	0.3331	0.3328	0.4996	0.4994	0.0303	0.0298	0.4961	0.4957	0.3366	0.3363	0.4996	0.4994
2	**64**	0.1709	0.6622	0.0078	0.4991	0.3331	0.3328	0.4996	0.4994	0.0154	0.0149	0.4922	0.4918	0.3348	0.3346	0.4996	0.4994
4	**2**	0.2402	0.6146	0.1249	0.4993	0.1999	0.1995	0.4995	0.4991	0.3331	0.3328	0.4996	0.4994	0.2855	0.2852	0.4996	0.4993
4	**4**	0.1331	0.6945	-0.0624	0.4990	0.1999	0.1995	0.4995	0.4991	0.1999	0.1995	0.4995	0.4991	0.2351	0.2348	0.4996	0.4992
4	**8**	0.0879	0.7427	-0.1561	0.4988	0.1999	0.1995	0.4995	0.4991	0.1110	0.1106	0.4991	0.4986	0.2161	0.2157	0.4995	0.4992
4	**16**	0.0679	0.7694	-0.2029	0.4986	0.1999	0.1995	0.4995	0.4991	0.0588	0.0583	0.4981	0.4976	0.2077	0.2073	0.4995	0.4992
4	**32**	0.0587	0.7835	-0.2263	0.4985	0.1999	0.1995	0.4995	0.4991	0.0303	0.0298	0.4961	0.4957	0.2037	0.2033	0.4995	0.4991
4	**64**	0.0543	0.7907	-0.2380	0.4984	0.1999	0.1995	0.4995	0.4991	0.0154	0.0149	0.4922	0.4918	0.2018	0.2014	0.4995	0.4991
8	**2**	0.2023	0.6391	0.0624	0.4992	0.1110	0.1106	0.4991	0.4986	0.3331	0.3328	0.4996	0.4994	0.1817	0.1813	0.4995	0.4991
8	**4**	0.0879	0.7427	-0.1561	0.4988	0.1110	0.1106	0.4991	0.4986	0.1999	0.1995	0.4995	0.4991	0.1379	0.1374	0.4993	0.4988
8	**8**	0.0445	0.8080	-0.2652	0.4983	0.1110	0.1106	0.4991	0.4986	0.1110	0.1106	0.4991	0.4986	0.1230	0.1226	0.4992	0.4987
8	**16**	0.0274	0.8449	-0.3198	0.4978	0.1110	0.1106	0.4991	0.4986	0.0588	0.0583	0.4981	0.4976	0.1167	0.1163	0.4991	0.4987
8	**32**	0.0201	0.8645	-0.3470	0.4974	0.1110	0.1106	0.4991	0.4986	0.0303	0.0298	0.4961	0.4957	0.1138	0.1134	0.4991	0.4987
8	**64**	0.0169	0.8746	-0.3606	0.4972	0.1110	0.1106	0.4991	0.4986	0.0154	0.0149	0.4922	0.4918	0.1124	0.1120	0.4991	0.4986
16	**2**	0.1842	0.6521	0.0312	0.4992	0.0588	0.0583	0.4981	0.4976	0.3331	0.3328	0.4996	0.4994	0.1052	0.1048	0.4990	0.4986
16	**4**	0.0679	0.7694	-0.2029	0.4986	0.0588	0.0583	0.4981	0.4976	0.1999	0.1995	0.4995	0.4991	0.0754	0.0750	0.4986	0.4981
16	**8**	0.0274	0.8449	-0.3198	0.4978	0.0588	0.0583	0.4981	0.4976	0.1110	0.1106	0.4991	0.4986	0.0661	0.0656	0.4983	0.4979
16	**16**	0.0131	0.8879	-0.3780	0.4968	0.0588	0.0583	0.4981	0.4976	0.0588	0.0583	0.4981	0.4976	0.0622	0.0618	0.4982	0.4978
16	**32**	0.0077	0.9105	-0.4070	0.4958	0.0588	0.0583	0.4981	0.4976	0.0303	0.0298	0.4961	0.4957	0.0605	0.0600	0.4982	0.4977
16	**64**	0.0055	0.9219	-0.4214	0.4951	0.0588	0.0583	0.4981	0.4976	0.0154	0.0149	0.4922	0.4918	0.0596	0.0591	0.4981	0.4977
32	**2**	0.1753	0.6588	0.0156	0.4991	0.0303	0.0298	0.4961	0.4957	0.3331	0.3328	0.4996	0.4994	0.0571	0.0566	0.4980	0.4976
32	**4**	0.0587	0.7835	-0.2263	0.4985	0.0303	0.0298	0.4961	0.4957	0.1999	0.1995	0.4995	0.4991	0.0396	0.0391	0.4971	0.4966
32	**8**	0.0201	0.8645	-0.3470	0.4974	0.0303	0.0298	0.4961	0.4957	0.1110	0.1106	0.4991	0.4986	0.0343	0.0338	0.4966	0.4961
32	**16**	0.0077	0.9105	-0.4070	0.4958	0.0303	0.0298	0.4961	0.4957	0.0588	0.0583	0.4981	0.4976	0.0322	0.0317	0.4964	0.4959
32	**32**	0.0036	0.9340	-0.4366	0.4938	0.0303	0.0298	0.4961	0.4957	0.0303	0.0298	0.4961	0.4957	0.0312	0.0307	0.4963	0.4958
32	**64**	0.0021	0.9450	-0.4511	0.4919	0.0303	0.0298	0.4961	0.4957	0.0154	0.0149	0.4922	0.4918	0.0307	0.0303	0.4962	0.4957
64	**2**	0.1709	0.6622	0.0078	0.4991	0.0154	0.0149	0.4922	0.4918	0.3331	0.3328	0.4996	0.4994	0.0298	0.0294	0.4961	0.4956
64	**4**	0.0543	0.7907	-0.2380	0.4984	0.0154	0.0149	0.4922	0.4918	0.1999	0.1995	0.4995	0.4991	0.0203	0.0198	0.4942	0.4937
64	**8**	0.0169	0.8746	-0.3606	0.4972	0.0154	0.0149	0.4922	0.4918	0.1110	0.1106	0.4991	0.4986	0.0175	0.0170	0.4932	0.4927
64	**16**	0.0055	0.9219	-0.4214	0.4951	0.0154	0.0149	0.4922	0.4918	0.0588	0.0583	0.4981	0.4976	0.0164	0.0159	0.4927	0.4922
64	**32**	0.0021	0.9450	-0.4511	0.4919	0.0154	0.0149	0.4922	0.4918	0.0303	0.0298	0.4961	0.4957	0.0159	0.0154	0.4925	0.4920
64	**64**	0.0009	0.9541	-0.4651	0.4880	0.0154	0.0149	0.4922	0.4918	0.0154	0.0149	0.4922	0.4918	0.0156	0.0151	0.4924	0.4919

Appendix B

Analytic Simulation Tools in C Language

```
float SneakPathR(float R, float m, float n)
{
   return( (R/(n-1)) +  (R/((m-1)*(n-1))) +  (R/(m-1)));
}
float parallelRes(float R1, float R2)
{
   return ((R1*R2)/(R1+R2));
}

int main()
{
   float Vin = 1;
   float Ron = 100;
   float Roff = 200000;
   float RsneakLRS, RsneakHRS;
   float RotherLRS = 100;
   float RotherHRS = 200000;

   int n, m;
   float Rl = 100;

   int size = 131072;
   float Von = Vin * (Rl / (Ron + Rl));
   float Voff = Vin * (Rl / (Roff + Rl));

   ofstream myfile;
   myfile.open ("margin.dat");

   for (n = 2; n <= size; n = n*2 )
   {
      m=n;
      RsneakLRS = SneakPathR(RotherLRS,m,n);
      RsneakHRS = SneakPathR(RotherHRS,m,n);

      float RmLRS = RotherLRS/(m-1);
      float RmnLRS = RotherLRS/((m-1)*(n-1));
      float RnLRS = RotherLRS/(n-1);

      float RmHRS = RotherHRS/(m-1);
      float RmnHRS = RotherHRS/((m-1)*(n-1));
      float RnHRS = RotherHRS/(n-1);

      float MemWC_on = (Ron * RsneakLRS)/(Ron + RsneakLRS);
      float MemBC_on = (Ron * RsneakHRS)/(Ron + RsneakHRS);

      float MemWC_off = (Roff * RsneakLRS)/(Roff + RsneakLRS);
      float MemBC_off = (Roff * RsneakHRS)/(Roff + RsneakHRS);

      float VWC_on = Vin * (Rl / (MemWC_on + Rl));
      float VBC_on = Vin * (Rl / (MemBC_on + Rl));

      float VWC_off = Vin * (Rl / (MemWC_off + Rl));
      float VBC_off = Vin * (Rl / (MemBC_off + Rl));

      float PWC_on = (Vin*Vin)/(parallelRes(Ron, RnLRS+RmLRS+RmnLRS) + Rl);
      float PBC_on = (Vin*Vin)/(parallelRes(Ron, RnHRS+RmHRS+RmnHRS) + Rl);
      float PWC_off = (Vin*Vin)/(parallelRes(Roff, RnLRS+RmLRS+RmnLRS) + Rl);
      float PBC_off = (Vin*Vin)/(parallelRes(Roff, RnHRS+RmHRS+RmnHRS) + Rl);

      //myfile <<setprecision(8) << setw(5) << n << "  " <<  setw(13) << VWC_on  << "  "<< setw(15) <<  ↩
           VBC_on << "  "<< setw(15) << VWC_off << "  "<< setw(15) << VBC_off << "  " << endl ;
      myfile <<setprecision(6) << setw(4) << m << "  "<< setw(4) << n << " "<< setw(10) << VWC_on - VWC_off ↩
           << "  "<< setw(20) << VWC_on - VBC_off << "  "<< setw(20) << VBC_on - VWC_off << "  "<< setw(20) ↩
           << VBC_on - VBC_off << "  " << endl ;
      //myfile <<setprecision(8) << setw(5) << n << "  " << setw(13) <<  PWC_on  << "  " << setw(13) << ↩
           PBC_on << "  "<< setw(13) << PWC_off << "  "<< setw(13) <<  PBC_off << "  " << endl ;
   }
```

```
64 }
```

Listing B.1: Computing the read margin and power cases for the FWFB read scheme

```
1  float SneakPathR(float R, float m, float n)
2  {
3     return( (R/(n-1)) +  (R/((m-1)*(n-1))) +  (R/(m-1)));
4  }
5  float parallelRes(float R1, float R2)
6  {
7     return ((R1*R2)/(R1+R2));
8  }
9  int main()
10 {
11    float Vin = 1;
12    float Ron = 100;
13    float Roff = 200000;
14    float RsneakLRS, RsneakHRS;
15    float RotherLRS = 100;
16    float RotherHRS = 200000;
17
18    int n, m;
19    float RL = 100;
20
21    int size = 131072;
22    float Von = Vin * (RL / (Ron + RL));
23    float Voff = Vin * (RL / (Roff + RL));
24
25    ofstream myfile;
26    myfile.open ("margin_GRFC.dat");
27
28    for (n = 2; n <= size; n = n*2 )
29    {
30       m=n;
31       RsneakLRS = SneakPathR(RotherLRS,m,n);
32
33       float RmLRS = RotherLRS/(m-1);
34       float RmnLRS = RotherLRS/((m-1)*(n-1));
35       float RnLRS = RotherLRS/(n-1);
36
37       float RmHRS = RotherHRS/(m-1);
38       float RmnHRS = RotherHRS/((m-1)*(n-1));
39       float RnHRS = RotherHRS/(n-1);
40
41       float RL_effLRS = (RL * (RmnLRS + RmLRS))/(RL + (RmnLRS + RmLRS));
42       float RL_effHRS = (RL * (RmnHRS + RmHRS))/(RL + (RmnHRS + RmHRS));
43
44       float VWC_on = Vin * (RL_effLRS / (Ron + RL_effLRS));
45       float VBC_on = Vin * (RL_effHRS / (Ron + RL_effHRS));
46
47       float VWC_off = Vin * (RL_effLRS / (Roff + RL_effLRS));
48       float VBC_off = Vin * (RL_effHRS / (Roff + RL_effHRS));
49
50       float PWC_on = (Vin*Vin)/parallelRes(Ron+RL_effLRS, RnLRS);
51       float PBC_on = (Vin*Vin)/parallelRes(Ron+RL_effHRS, RnHRS);
52       float PWC_off = (Vin*Vin)/parallelRes(Roff+RL_effLRS, RnLRS);
53       float PBC_off = (Vin*Vin)/parallelRes(Roff+RL_effHRS, RnHRS);
54
55       myfile <<setprecision(8) << setw(5) << n << "  " <<  setw(13) << VWC_on  << "  "<< setw(15) <<  VBC_on↩
               << "  "<< setw(15) << VWC_off << "  "<< setw(15) << VBC_off << "  " << endl ;
56       //myfile <<setprecision(6) << setw(4) << m << "  "<< setw(4) << n << " "<< setw(10) << VWC_on - ↩
               VWC_off << "  "<< setw(20) << VWC_on - VBC_off << "  "<< setw(20) << VBC_on - VWC_off << "  "<< ↩
               setw(20) << VBC_on - VBC_off << "  " << endl ;
57       //myfile <<setprecision(8) << setw(5) << n << "  " << setw(13) <<  PWC_on  << "  " << setw(13) << ↩
               PBC_on << "  "<< setw(13) << PWC_off << "  "<< setw(13) <<  PBC_off << "  " << endl ;
58    }
59 }
```

Listing B.2: Computing the read margin and power cases for the FWGB read scheme

```
1  float SneakPathR(float R, float m, float n)
2  {
3     return( (R/(n-1)) +  (R/((m-1)*(n-1))) +  (R/(m-1)));
4  }
5  float parallelRes(float R1, float R2)
6  {
7     return ((R1*R2)/(R1+R2));
8  }
9  int main()
10 {
11    float Vin = 1;
12    float Ron = 100;
13    float Roff = 200000;
14    float RsneakLRS, RsneakHRS;
15    float RotherLRS = 100;
16    float RotherHRS = 200000;
17
18    int n, m;
19    float RL = 100;
20
21    int size = 131072;
```

```
    float Von = Vin * (RL / (Ron + RL));
    float Voff = Vin * (RL / (Roff + RL));

    ofstream myfile;
    myfile.open ("margin_GRFC.dat");

    for (n = 2; n <= size; n = n*2 )
    {
       m=n;
       float RmLRS = RotherLRS/(m-1);
       float RmnLRS = RotherLRS/((m-1)*(n-1));
       float RnLRS = RotherLRS/(n-1);

       float RmHRS = RotherHRS/(m-1);
       float RmnHRS = RotherHRS/((m-1)*(n-1));
       float RnHRS = RotherHRS/(n-1);


       float RL_effLRS = (RL * RnLRS)/(RL + RnLRS);
       float RL_effHRS = (RL * RnHRS)/(RL + RnHRS);

       float VWC_on = Vin * (RL_effLRS / (Ron + RL_effLRS));
       float VBC_on = Vin * (RL_effHRS / (Ron + RL_effHRS));

       float VWC_off = Vin * (RL_effLRS / (Roff + RL_effLRS));
       float VBC_off = Vin * (RL_effHRS / (Roff + RL_effHRS));

       float PWC_on = (Vin*Vin)/parallelRes(Ron+RL_effLRS, RmLRS+RmnLRS);
       float PBC_on = (Vin*Vin)/parallelRes(Ron+RL_effHRS, RmHRS+RmnHRS);
       float PWC_off = (Vin*Vin)/parallelRes(Roff+RL_effLRS, RmLRS+RmnLRS);
       float PBC_off = (Vin*Vin)/parallelRes(Roff+RL_effHRS, RmHRS+RmnHRS);

       myfile <<setprecision(8) << setw(5) << n << "  " <<  setw(13) << VWC_on  << "  "<< setw(15) <<  VBC_on↩
              << "  "<< setw(15) << VWC_off << "  "<< setw(15) << VBC_off << "  " << endl ;
       //myfile <<setprecision(6) << setw(4) << m << "  "<< setw(4) << n << " "<< setw(10) << VWC_on - ↩
            VWC_off << "  "<< setw(20) << VWC_on - VBC_off << "  "<< setw(20) << VBC_on - VWC_off << "  "<< ↩
            setw(20) << VBC_on - VBC_off << "  " << endl ;
       //myfile <<setprecision(8) << setw(5) << n << "  " << setw(13) <<  PWC_on  << "  " << setw(13) << ↩
            PBC_on << "  "<< setw(13) << PWC_off << "  "<< setw(13) <<  PBC_off << "  " << endl ;
    }
}
```

Listing B.3: Computing the read margin and power cases for the GWFB read scheme

```
float SneakPathR(float R, float m, float n)
{
   return( (R/(n-1)) +  (R/((m-1)*(n-1))) +  (R/(m-1)));
}
float parallelRes(float R1, float R2)
{
   return ((R1*R2)/(R1+R2));
}
int main()
{
   float Vin = 1;
   float Ron = 100;
   float Roff = 200000;
   float RsneakLRS, RsneakHRS;
   float RotherLRS = 100;
   float RotherHRS = 200000;

   int n, m;
   float RL = 100;

   int size = 131072;
   float Von = Vin * (RL / (Ron + RL));
   float Voff = Vin * (RL / (Roff + RL));

   ofstream myfile;
   myfile.open ("margin_GRFC.dat");

   for (n = 2; n <= size; n = n*2 )
   {
      m=n;
      float RmLRS = RotherLRS/(m-1);
      float RmnLRS = RotherLRS/((m-1)*(n-1));
      float RnLRS = RotherLRS/(n-1);

      float RmHRS = RotherHRS/(m-1);
      float RmnHRS = RotherHRS/((m-1)*(n-1));
      float RnHRS = RotherHRS/(n-1);

      float RL_effLRS = (RL * RmLRS)/(RL + RmLRS);
      float RL_effHRS = (RL * RmHRS)/(RL + RmHRS);

      float VWC_on = Vin * (RL_effLRS / (Ron + RL_effLRS));
      float VBC_on = Vin * (RL_effHRS / (Ron + RL_effHRS));

      float VWC_off = Vin * (RL_effLRS / (Roff + RL_effLRS));
      float VBC_off = Vin * (RL_effHRS / (Roff + RL_effHRS));

      float PWC_on = (Vin*Vin)/parallelRes(Ron+RL_effLRS, RnLRS);
      float PBC_on = (Vin*Vin)/parallelRes(Ron+RL_effHRS, RnHRS);
      float PWC_off = (Vin*Vin)/parallelRes(Roff+RL_effLRS, RnLRS);
      float PBC_off = (Vin*Vin)/parallelRes(Roff+RL_effHRS, RnHRS);

      myfile <<setprecision(8) << setw(5) << n << "  " <<  setw(13) << VWC_on  << "  "<< setw(15) <<  VBC_on↩
             << "  "<< setw(15) << VWC_off << "  "<< setw(15) << VBC_off << "  " << endl ;
```

```
54        //myfile <<setprecision(6) << setw(4) << m << "  "<< setw(4) << n << " "<< setw(10) << VWC_on - ↩
              VWC_off << "  "<< setw(20) << VWC_on - VBC_off << "  "<< setw(20) << VBC_on - VWC_off << "  "<< ↩
              setw(20) << VBC_on - VBC_off << "  " << endl ;
55        //myfile <<setprecision(8) << setw(5) << n << "  " << setw(13) <<  PWC_on  << "  " << setw(13) << ↩
              PBC_on << "  "<< setw(13) << PWC_off << "  "<< setw(13) <<  PBC_off << "  " << endl ;
56     }
57  }
```

Listing B.4: Computing the read margin and power cases for the GWGB read scheme

```
1
2
3   float Ron = 100; float Roff = 200000;
4
5   float SneakPathR(int Koffn, int Koffm, int Koffmn, int m, int n)
6   {
7      float Rn = (Ron*Roff)/( ((n-1)*Roff) - (Koffn*(Roff-Ron)) );
8      float Rm = (Ron*Roff)/( ((m-1)*Roff) - (Koffm*(Roff-Ron)) );
9      float Rmn = (Ron*Roff)/( ((m-1)*(n-1)*Roff) - (Koffmn*(Roff-Ron)) );
10     return (Rm + Rn + Rmn);
11  }
12
13  int main()
14  {
15     float Vin = 1; float Rsneak1, Rsneak2; float Koff1 = 0; float Koff2 = 0; float Rl = 100; int n, m, size =↩
           512;
16
17     ofstream myfile;
18     myfile.open ("margin_rand_512x512.dat");
19
20     int Koffn1, Koffm1, Koffmn1;
21
22     m=n=size;
23     for (int samples = 1; samples <= 1000000; samples=samples+1 )
24     {
25        //Koff is the no of memristor in Off state
26
27        Koffn1 = rand()%n; Koffm1 = rand()%m; Koffmn1 = rand()%((n-1)*(m-1)+1);
28
29        Koff1 = Koffn1 + Koffm1 + Koffmn1;
30        Rsneak1 = SneakPathR(Koffn1, Koffm1, Koffmn1, m, n);
31
32        float MemWC_on = (Ron * Rsneak1)/(Ron + Rsneak1);
33        float MemWC_off = (Roff * Rsneak1)/(Roff + Rsneak1);
34
35        float VWC_on = Vin * (Rl / (MemWC_on + Rl));
36        float VWC_off = Vin * (Rl / (MemWC_off + Rl));
37
38        myfile <<setprecision(9) << setw(6) << samples << "  " << setw(5) << Koff1 << "  "<< setw(15) << ↩
              VWC_on << "  "<< setw(15) << VWC_off << setw(15) << VWC_on-VWC_off << "  " << endl ;
39     }
40  }
```

Listing B.5: Computing the read margin for the FWFB read scheme using random resistance states

```
1   //Generate multiple samples with random data distribution with GWGB, FWGB, GWFB
2   int main()
3   {
4      float Vin = 1; float Koff1 = 0; float Rl = 100; int n, m, size = 4; float Ron = 100; float Roff = 200000;
5
6      ofstream myfile;
7      myfile.open ("margin_rand_gnd2_4x4.dat");
8
9      int Koffn1, Koffm1, Koffmn1;
10
11     m=n=size;
12     for (int samples = 1; samples <= 1000000; samples=samples+1 )
13     {
14        //Koff is the no of memristor in Off state
15        Koffn1 = rand()%n;
16        Koffm1 = rand()%m;
17        Koffmn1 = rand()%((n-1)*(m-1)+1);
18
19        float Rn = (Ron*Roff)/( ((n-1)*Roff) - (Koffn1*(Roff-Ron)) );
20        float Rm = (Ron*Roff)/( ((m-1)*Roff) - (Koffm1*(Roff-Ron)) );
21        float Rmn = (Ron*Roff)/( ((m-1)*(n-1)*Roff) - (Koffmn1*(Roff-Ron)) );
22
23        //float Rl_eff = (Rl * (Rm + Rmn))/(Rl + (Rm + Rmn)); // FWGB
24        float Rl_eff = (Rl * Rn)/(Rl + Rn); // GWFB
25        //float Rl_eff = (Rl * Rm)/(Rl + Rm);  // GWGB
26
27        float MemWC_on = Ron ;  //Rsel is not affected
28        float MemWC_off = Roff ;
29        float VWC_on = Vin * (Rl_eff / (MemWC_on + Rl_eff));
30        float VWC_off = Vin * (Rl_eff / (MemWC_off + Rl_eff));
31
32        myfile <<setprecision(9) << setw(6) << samples << "  " << setw(5) << Koffmn1+Koffm1 << "  "<< setw(15)↩
              << VWC_on << "  "<< setw(15) << VWC_off << setw(15) << VWC_on-VWC_off << "  " << endl ;
33     }
34  }
```

Listing B.6: Computing the read margin for GWGB, FWGB and GWFB read scheme using random resistance states

```
float SneakPathR(float Ron, float Rl, float m, float n, int type)
{
   float row, column, middle, divisor;
   row = ((Ron/(m-1)) + (Ron/((n-1)*(m-1)))) * (Ron/(n-1));
   column = ((Ron/(m-1)) + (Ron/((n-1)*(m-1)))) * (Rl/(n-1));
   middle =  (Ron/(n-1)) * (Rl/(n-1))  ;

   if (type == 0) {
      divisor = Rl/(n-1);
   }
   else {divisor = Ron/(n-1); }
   return( (row  +  middle + column)/divisor );
}

int main()
{
   float Vin = 1; float Ron = 100; float Roff = 200000;
   float RsneakLRS, RsneakHRS; float RsneakLRS_b, RsneakHRS_b; int n, m;
   float Rl = 100; int size = 32; float Von = Vin * (Rl / (Ron + Rl)); float Voff = Vin * (Rl / (Roff + Rl))↩
        ;

   ofstream myfile;
   myfile.open ("new3d.dat");

   for (n = size; n <= size; n = n+1 )
   {
      for (Rl = 0.00001; Rl <= 100000; Rl*=1.1)
      {
         for (Roff = Ron; Roff <= Ron*100000; Roff*=1.2)
         {
            m=32;       //row
            n=size;     //column
            RsneakLRS = SneakPathR(Ron,Rl,m,n,0);
            RsneakHRS = SneakPathR(Roff,Rl,m,n,0);

            RsneakLRS_b = SneakPathR(Ron,Rl,m,n,1);
            RsneakHRS_b = SneakPathR(Roff,Rl,m,n,1);

            float MemWC_on = (Ron * RsneakLRS)/(Ron + RsneakLRS);
            float MemBC_on = (Ron * RsneakHRS)/(Ron + RsneakHRS);

            float MemWC_off = (Roff * RsneakLRS)/(Roff + RsneakLRS);
            float MemBC_off = (Roff * RsneakHRS)/(Roff + RsneakHRS);

            float Rl_eff_on = (Rl * RsneakLRS_b)/(Rl + RsneakLRS_b);
            float Rl_eff_off = (Rl * RsneakHRS_b)/(Rl + RsneakHRS_b);

            float VWC_on = Vin * (Rl_eff_on/(MemWC_on + Rl_eff_on));
            float VBC_on = Vin * (Rl_eff_off/(MemBC_on + Rl_eff_off));

            float VWC_off = Vin * (Rl_eff_on/(MemWC_off + Rl_eff_on));
            float VBC_off = Vin * (Rl_eff_off/(MemBC_off + Rl_eff_off));

            myfile <<setprecision(8) << setw(13) << Roff/Ron << "  "<< setw(20) <<  Rl << "  " << setw(20) ↩
                 << (VBC_on - VWC_off)*100  << setw(20) << (VBC_on - VBC_off)*100 << setw(20) << (VWC_on - ↩
                 VWC_off)*100 << setw(20) << (VWC_on - VBC_off)*100 << "  " <<endl ;
         }
      }
   }
}
```

Listing B.7: Computing all the possible cases of sneak-path with all cells in a row are selected

```
float SneakPathR(float R, float Rl, float m, float n, int type, int k)
{
   float R1, R2, R3, Ra, Rb, Rc, Rnk, Rmnk, Rk, Rmk, Rlk, Rm, divisor;

   Rnk = R/(n-k);
   Rmnk = R/((m-1)*(n-k));
   Rk = R/(k-1);
   Rmk = R/((m-1)*(k-1));
   Rlk = Rl/(k-1);
   Rm = R/(m-1);

   Ra = ( (R/(k-1)) * (R/((m-1)*(k-1))) ) + ( (Rl/(k-1)) * (R/((m-1)*(k-1))) ) + ( (Rl/(k-1)) * (R/(k-1)) ) ↩
        ;
   Rb = ( (R/(k-1)) * (R/((m-1)*(k-1))) ) + ( (Rl/(k-1)) * (R/((m-1)*(k-1))) ) + ( (Rl/(k-1)) * (R/(k-1)) ) ↩
        ;
   Rc = ( (R/(k-1)) * (R/((m-1)*(k-1))) ) + ( (Rl/(k-1)) * (R/((m-1)*(k-1))) ) + ( (Rl/(k-1)) * (R/(k-1)) ) ↩
        ;

   Ra = Ra/Rlk; Rb = Rb/Rmk; Rc = Rc/Rk ;

   Ra = (Ra * (Rnk+Rmnk) )/ ( Ra + (Rnk+Rmnk) );

   R1 = (Ra*Rb)/(Ra+Rb+Rc);
```

```
    R2 = (Ra*Rc)/(Ra+Rb+Rc);
    R3 = (Rb*Rc)/(Ra+Rb+Rc);

    R2 = R2 + Rm;

    Ra = ((R1*R2) + (R2*R3) + (R1*R3) )/R3 ;
    Rb = ((R1*R2) + (R2*R3) + (R1*R3) )/R2 ;
    Rc = ((R1*R2) + (R2*R3) + (R1*R3) )/R1 ;

    if (type == 0) {
       return Ra;
    }
    else {return Rc;  }
}

int main()
{
    float Vin = 1; float Ron = 100; float Roff = 200000;
    float RsneakLRS, RsneakHRS; float RsneakLRS_b, RsneakHRS_b;

    int n, m; float Rl = 100; int size = 4;

    m=4; n=size;   //row //column
    RsneakLRS = SneakPathR(Ron,Rl,m,n,0, 2);
    RsneakHRS = SneakPathR(Roff,Rl,m,n,0, 2);

    RsneakLRS_b = SneakPathR(Ron,Rl,m,n,1, 2);
    RsneakHRS_b = SneakPathR(Roff,Rl,m,n,1, 2);

    float MemWC_on = (Ron * RsneakLRS)/(Ron + RsneakLRS);
    float MemBC_on = (Ron * RsneakHRS)/(Ron + RsneakHRS);

    float MemWC_off = (Roff * RsneakLRS)/(Roff + RsneakLRS);
    float MemBC_off = (Roff * RsneakHRS)/(Roff + RsneakHRS);

    float Rl_eff_on = (Rl * RsneakLRS_b)/(Rl + RsneakLRS_b);
    float Rl_eff_off = (Rl * RsneakHRS_b)/(Rl + RsneakHRS_b);

    float VWC_on = Vin * (Rl_eff_on / (MemWC_on + Rl_eff_on));
    float VBC_on = Vin * (Rl_eff_off / (MemBC_on + Rl_eff_off));

    float VWC_off = Vin * (Rl_eff_on / (MemWC_off + Rl_eff_on));
    float VBC_off = Vin * (Rl_eff_off / (MemBC_off + Rl_eff_off));

    cout <<setprecision(8) << setw(13) << VWC_on << "  "<< setw(20) <<  VBC_on << "  "<< setw(20) << VWC_off
          << "  "<< setw(20) <<  VBC_off << "  " << endl ;
    cout <<setprecision(8) << setw(13) << Roff/Ron << "  "<< setw(20) <<  Rl << "  " << setw(20) << (VBC_on -
          VWC_off)*100  << setw(20) << (VBC_on - VBC_off)*100 << setw(20) << (VWC_on - VWC_off)*100 << setw
         (20) << (VWC_on - VBC_off)*100 << "  " <<endl ;

}
```

Listing B.8: Computing all the possible cases of read-out voltages and read margin when some of the cells in the desired row are selected for read

```
long long  *data; long long *inputdata; long long  *outputdata;
string inputdataStr; int parity[1000][1000] = {0}; int parity1[100] = {0}; int databit = 0; int rbit = 1;
int counter = 0; char *str[10000]; char *filestr;
int parityfunc[10000][10000] = {0};

long long *AllocateMemoryInt(int n);
char *AllocateMemory(int n);
char *DecTo2Com(long long int DecimalNumber, int NumOfBits);
int random(int rbit);
long long int i; long long j=0;

int main()
{
    ofstream Generateinput ;
    Generateinput.open("inputdata.txt");
    ofstream Generateoutput ;
    Generateoutput.open("outputdata.txt");
    ofstream Generateparity ;
    Generateparity.open("paritydata.txt");

    filestr = AllocateMemory(1000);
    for (i=0; i<=7; i++)  //Generate input. Data to be worked on
    {
       if(i>=1){
          j = pow(2,i)-1;
       }

       filestr = DecTo2Com (j, 8);// cout <<  j << endl;   //convert integer to binary
       Generateinput << filestr << endl;
    }

    inputdataStr = AllocateMemory(1000);

    ifstream Readinput ("inputdata.txt");   // Read input generated
    if (Readinput.is_open())
    {
       while ( getline (Readinput,inputdataStr) )
       {
          databit = inputdataStr.length();
          srand((unsigned)time(NULL));
          while(pow(2,rbit)<(databit + rbit+1))    //Compute number of parity bits needed
            {rbit++; }
```

```
43            for ( i = 0; i < databit; i++)
44            {
45               str[i] = AllocateMemory(rbit+1);             //str is the syndrome. The no of column(i) is same ↩
                      as databit
46               //p[i]  = AllocateMemoryInt(rbit+1);      //the no of row depends on the no of parity bit
47            }
48             data = AllocateMemoryInt(rbit+1);           //For storing paritydata generated
49             inputdata = AllocateMemoryInt(rbit+1);
50             outputdata = AllocateMemoryInt(databit+rbit+1);
51            j = 0;
52            parity1[rbit-1] = 1;
53            for ( i = 0; i < rbit; i++)
54            {
55               parity[i][j] = 1;                //identity matrix. put 1 when i=j
56               j++;
57               parity1[rbit-1-j] = parity1[rbit-j]*2;    //store decimal equivalent of identity parity in each ↩
                      row
58            }                                   //
59
60            for ( i = 0; i < databit; i++)            //Need to generate syndrome equall to the no of databit.
61            {
62               counter = 0;
63               int datatemp = random (rbit);       //Generate and store paritydata temp
64               for ( j = 0; j < databit; j++)
65               {
66                  if (datatemp == data[j] || datatemp == parity1[j])    //exclude parity identities(2,4,8...), ↩
                         zero and repetitions
67                  {
68                     counter++;
69                     i = i - 1;        //avoid repetion and parity data
70                  }
71               }
72               if (counter == 0)       //Save paritydata if conditions are satified
73               {data[i] = datatemp; }
74            }
75
76            for ( i = 0; i < databit; i++)            //convert integer paritydata to binary
77            {
78               str[i] = DecTo2Com (data[i], rbit);
79            }
80
81            for (j=0; j < rbit; j++)
82            {
83               for (i=0; i < databit ; i++)
84               {
85                  if(str[i][j] == '1')
86                  {parityfunc[j][i] =  1; }        //Overall Syndrome. rbit rows and databit column.
87               }cout << endl;
88            }
89
90            for(i=0; i<databit; i++) {                //Convert input read as string to integer
91               if (inputdataStr[i] == '1') {
92                  inputdata[i] = 1;
93               }
94               else {inputdata[i] = 0; }
95            }
96
97            long long *newparity ;
98            newparity = AllocateMemoryInt(rbit+1);
99
100           memcpy(&outputdata, &inputdata, sizeof(outputdata));
101
102           for (j=0; j < rbit; j++)
103           {
104              for (i=0; i < databit ; i++)
105              {
106                 if (parityfunc[j][i] == 1)               //content of syndrome is 1 use for xor
107                 {newparity[j] = newparity[j] ^ inputdata[i]; }   //XOR the data in that position in the input↩
                        data
108              }
109              outputdata[databit+j] = newparity[j];          //append to end of originaldata
110              cout <<  "j = " << j <<  ", " << newparity[j] << endl;
111           }
112
113           for (j=0; j < rbit+databit ; j++)
114           {
115              Generateoutput << outputdata[j] ;
116           }
117           Generateoutput << endl;
118           for (j=0; j < rbit ; j++)
119           {Generateparity << outputdata[databit+j]; }
120           Generateparity << endl;
121
122        }
123        Readinput.close();
124     }
125     else cout << "Unable to open file";
126  }
127
128  int random(int rbit)  // Generate random parity data between 0 and 2^paritybit
129  {
130     int randomno = rand() % (int)(pow(2, rbit) -1) + 1; //  +1 helps to exclude 0, -1 helps to avoid 2^↩
           paritybit
131     return randomno ;
132  }
133
134  char *DecTo2Com(long long int DecimalNumber, int NumOfBits)
135  {
136     int temp, i,j; int flag = 0; char *binary;
137     long long *binary1;
138     binary = AllocateMemory(NumOfBits+1);
139     binary1 = AllocateMemoryInt(NumOfBits+1);
140     temp = --NumOfBits;
```

```
141
142     while (DecimalNumber > 0) {
143        binary1[NumOfBits--] = DecimalNumber%2;
144        DecimalNumber=DecimalNumber/2;
145     }
146     for(i=0; i<=temp; i++) {
147        if (binary1[i] == 1) {
148           binary[i] = '1';
149        }
150        else {binary[i] = '0'; }
151     }
152     return binary;
153  }
154
155  char *AllocateMemory(int n)
156  {
157     char *str;
158     str = (char *)calloc (n, sizeof(char));
159     if (str == NULL){
160        printf("Cannot Allocate Memoery");
161        exit(2);
162     }
163     return str;
164  }
165
166  long long *AllocateMemoryInt(int n)
167  {
168     long long *str;
169     str = (long long *)calloc (n, sizeof(long long));
170     if (str == NULL){
171        printf("Cannot Allocate Memoery");
172        exit(2);
173     }
174     return str;
175  }
```

Listing B.9: Designing memristor-based structure with Hamming code

```
1   int *data; int *inputdata; int *outputdata; string inputdataStr; int databit = 0;
2   int zeros = 0; int ones = 0; char *str[10000]; char *filestr;
3   int *AllocateMemoryInt(int n); char *AllocateMemory(int n);
4
5   char *DecTo2Com(long int DecimalNumber, int NumOfBits);
6   int i, j;
7   int main()
8   {
9      srand((unsigned)time(NULL));
10
11     ofstream Generateoutput ;
12     Generateoutput.open("outputdata.txt");
13     filestr = AllocateMemory(1000);
14     inputdataStr = AllocateMemory(1000);
15
16     ifstream Readinput ("../outputdata.txt");
17     if (Readinput.is_open())
18     {
19        while ( getline (Readinput,inputdataStr) )
20        {
21            databit = inputdataStr.length();
22            data = AllocateMemoryInt(databit+1);
23            inputdata = AllocateMemoryInt(databit+1);
24            outputdata = AllocateMemoryInt(databit+1);
25
26           ones = 0; zeros=0;
27           for(i=0; i<databit; i++)
28           {
29              if (inputdataStr[i] == '1')
30              {
31                 inputdata[i] = 1;
32                 ones = ones+1;
33              }
34              else
35              {
36                 inputdata[i] = 0;
37                 zeros+=1;
38              }
39           }
40           cout << "1s- " << ones << " 0s- " << zeros << endl;
41
42           if (ones > zeros)
43           {
44              for(i=0; i<databit+1; i++)
45              {
46                 inputdata[i] = inputdata[i] ^1;
47                 Generateoutput << inputdata[i] ;
48              }
49              Generateoutput << endl ;
50           }
51           else
52           {
53              for(i=0; i<databit+1; i++)
54              {Generateoutput << inputdata[i] ; }
55              Generateoutput << endl ;
56           }
57        }
58        Readinput.close();
59     }
60  }
```

```
61
62  char *DecTo2Com(long int DecimalNumber, int NumOfBits)
63  {
64     int temp, i,j;
65     int flag = 0;
66     char *binary;
67     binary = AllocateMemory(NumOfBits+1);
68
69     temp = --NumOfBits;
70
71     while (DecimalNumber > 0) {
72        binary[NumOfBits--] = DecimalNumber%2;
73        DecimalNumber=DecimalNumber/2;
74     }
75     for(i=0; i<=temp; i++) {
76        if (binary[i] == 1) {
77           binary[i] = '1';
78        }
79        else {binary[i] = '0'; }
80     }
81     return binary;
82  }
83
84  char *AllocateMemory(int n)
85  {
86     char *str;
87     str = (char *)calloc (n, sizeof(char));
88     if (str == NULL){
89        printf("Cannot Allocate Memory");
90        exit(2);
91     }
92     return str;
93  }
94
95  int *AllocateMemoryInt(int n)
96  {
97     int *str;
98     str = (int *)calloc (n, sizeof(int));
99     if (str == NULL){
100       printf("Cannot Allocate Memory");
101       exit(2);
102    }
103    return str;
104 }
```

Listing B.10: Computing the inversion flag bit from memristor-based structures with Inversion code

```
1   int main()
2   {
3      for (int size=4; size<=4; size=size*2)
4      {
5         char Resistance[50]; char wordline[50]; char bitline[50]; char Memristor[500]; char directory[500];
6         char choice[3] = "xy"; char filename[50]; double initial[200000] = {0};
7
8         double D = 3e-09; int p_coff = 2; double C = 9e-10;
9         double xo = 3e-09;          //value of unselected cells
10        double alpha = 20; double uv = 1e-15;
11        int Rl = 100; int index = 50; int selectedRow; int selectedCol;
12        int m = size, n = size, Ron = 100, Roff = 200000;
13
14        selectedRow = m/2;
15        selectedCol = n/2;
16
17        ofstream netlistFile;
18        ofstream oceanfile;
19        snprintf (filename, sizeof filename, "%dX%d", m, n);
20        netlistFile.open (filename);
21        snprintf (filename, sizeof filename, "oceanScript_%dX%d", m, n);
22        oceanfile.open (filename);
23
24        netlistFile << endl << "// Library name: Memristor" << endl << "// Cell name: Array_memristor_" << ↩
              filename << "_Write_Voltages" << endl << "// View name: schematic" << endl << endl;
25
26        oceanfile << "simulator( 'spectre ) " << endl;
27        sprintf (directory, "design( \"/home/cosc/csdkp/linux/pradhan/VLSI/Sim/↩
              Array_memristor_4X4_Write_voltages/spectre/schematic/netlist/netlist\")");
28        //sprintf (result, "resultsDir( \"/home/cosc/csxjm/linux/IC_Design/VLSI3_2014/Sim/↩
              Array_memristor_4X4_Write_voltages/spectre/schematic\" )");
29
30        oceanfile << directory << endl << "modelFile(\n \t '(\"/usr/local/cds-een/cadence/ams_4.10/spectre/c35↩
              /soac/cmos53.scs\" \"cmostm\")\n \t " << endl <<
31        "'(\"/usr/local/cds-een/cadence/ams_4.10/spectre/c35/soac/res.scs\" \"restm\")\n \t '(\"/usr/local/cds↩
              -een/cadence/ams_4.10/spectre/c35/soac/cap.scs\" \"captm\")\n \t " << endl <<
32        "'(\"/usr/local/cds-een/cadence/ams_4.10/spectre/c35/soac/bip.scs\" \"biptm\")\n \t '(\"/usr/local/cds↩
              -een/cadence/ams_4.10/spectre/c35/soac/ind.scs\" \"indtm\")\n \t " << endl <<
33        "'(\"/usr/local/cds-een/cadence/ams_4.10/spectre/c35/soac/esddiode.scs\" \"esddiodetm\")\n \t '(\"/usr↩
              /local/cds-een/home/dkmgr/gpdk180_v3.2/models/spectre/gpdk.scs\" \"stat\")\n)" << endl <<
34        "definitionFile(\n \t \"/usr/local/cds-een/cadence/ams_4.10/spectre/c35/soac/processOption.scs\"\n)" ↩
              << endl <<
35        "analysis('tran ?stop \"1n\"  ?errpreset \"moderate\"  ?start \"0\"  ?step \"200p\" ?maxstep \"500p\" ↩
              ?finalTimeOp nil)" << endl <<
36        "desVar(   \"Rw\" 0.1   )" << endl <<
37        "desVar(   \"x\" 3n  )" << endl <<
38        "desVar(   \"y\" 3n  )" << endl <<
39        "desVar(   \"V\" 0.5 )" << endl << endl;
40
```

```
41        oceanfile  << "envOption(\n 'analysisOrder  list(\"tran\")\n)\n" << endl ;
42        oceanfile  << "envSetVal(\"spectre.envOpts\" \"userCmdLineOption\" 'string \"+turbo +parasitics ++↩
              parasitics ++aps +cktpreset=sampled\")" << endl ;
43        oceanfile  << "temp( 27 )\nout = outfile(\"./Results"<< m << "x" << n << "_" << index <<".out\" \"w\")↩
              \n" << endl;
44        oceanfile  << "option('reltol 1e-3)"   << endl;
45        oceanfile  << "option( ?categ 'turboOpts 'errorLevel  \"Moderate\" )"   << endl;
46        oceanfile  << "option('maxnotestologfile  \"1\" 'maxwarnstologfile  \"1\" 'maxwarns  \"1\" 'maxnotes  ↩
              \"1\" ) "  << endl;
47        oceanfile << "saveOption( 'save \"none\" 'currents \"selected\")" << endl;
48        oceanfile << "save( 'i \"m" <<setw(5) << setfill('0') << selectedRow <<setw(5) << setfill('0') << ↩
              selectedCol  << ":p\" \"m" <<setw(5) << setfill('0') << m <<setw(5) << setfill('0') << ↩
              selectedCol  << ":p\")" << endl;
49        oceanfile  << "time = 1n"   << endl;
50
51        oceanfile  << "paramAnalysis(\"Rw\" ?values '(0.0001 0.001 0.01 0.1 1 ))"   << endl;
52        oceanfile << "paramRun()\nselectResult( 'tran )\noutputs()" << endl;
53
54        oceanfile << "\tocnPrint(?output out IT(\"m" <<setw(5) << setfill('0') << selectedRow <<setw(5) << ↩
              setfill('0') << selectedCol  << ":p\")*200 IT(\"m" <<setw(5) << setfill('0') << m <<setw(5) << ↩
              setfill('0') << selectedCol  << ":p\")*200 ?precision 6 ?numberNotation 'none ?from 1n ?to 1n ?↩
              step 0.1n)" << endl;
55
56        oceanfile << "close(out)" << endl;
57
58
59        for (i=1; i <= m; i++) //row
60        {
61           for (j=1; j <= n; j++)  //column
62           {
63              sprintf (Resistance, "R%05d%05d (M%05d%05d 0) resistor r=1G", i, j, i, j);
64              netlistFile << Resistance << endl;
65           }
66           netlistFile << endl;
67        }
68
69        for (i=1; i <= m; i++) //row
70        {
71           for (j=1; j <= n+1; j++)   //column
72           {
73              if (j == 1 && i == selectedRow)  //(j == 1 && i == 1)
74              {sprintf (wordline, "W%05d%05d (Vin netW%05d%05d) resistor r=Rw", i, j, i, j); }
75              else if (j == n+1 && i==selectedRow)   //extra Rw for selected wordline  (j == n+1 && i==1)
76              {sprintf (wordline, "W%05d%05d (netW%05d%05d Vin2) resistor r=Rw", i, j, i, j-1); }
77              else if(j==1 && i != selectedRow)   //(j==1 && i > 1)
78              {sprintf (wordline, "W%05d%05d (VWL%05d%05d netW%05d%05d) resistor r=Rw", i,j, i,j, i,j); }
79              else if(j==n+1 && i != selectedRow)    //extra Rw for unselected wordline (j==n+1 && i > 1)
80              {sprintf (wordline, "W%05d%05d (netW%05d%05d VWL%05d%05d) resistor r=Rw", i,j, i,j-1, i,j); }
81              else
82              {sprintf (wordline, "W%05d%05d (netW%05d%05d netW%05d%05d) resistor r=Rw", i,j, i,j-1, i,j); }
83              netlistFile << wordline << endl;
84           }
85           netlistFile << endl;
86        }
87
88        for (i=1-1; i <= m; i++) //row
89        {
90           for (j=1; j <= n; j++)  //column
91           {
92              if (i==m && j == selectedCol)  //(i==m && j == n)
93              {sprintf (bitline, "B%05d%05d (netB%05d%05d Vgnd) resistor r=Rw", i, j, i, j); }
94              else if (i==0 && j == selectedCol)  //extra Rw for selected bitline (i==0 && j == n)
95              {sprintf (bitline, "B%05d%05d (Vgnd2 netB%05d%05d) resistor r=Rw", i, j, i+1, j); }
96              else if(i==m && j != selectedCol)
97              {sprintf (bitline, "B%05d%05d (netB%05d%05d VBL%05d%05d) resistor r=Rw", i,j, i,j, i, j); }
98              else if(i==0 && j != selectedCol)       //extra Rw for selected bitline
99              {sprintf (bitline, "B%05d%05d (VBL%05d%05d netB%05d%05d ) resistor r=Rw", i,j, i,j, i+1, j);}
100             else
101             {sprintf (bitline, "B%05d%05d (netB%05d%05d netB%05d%05d) resistor r=Rw", i,j, i,j, i+1,j);}
102             netlistFile << bitline << endl;
103          }
104          netlistFile << endl;
105       }
106
107       for (i=1; i <= m; i++) //row
108       {
109          for (j=1; j <= n; j++)  //column
110          {
111             if(i==1 && j == n)
112             {sprintf (Memristor, "m%05d%05d (netW%05d%05d netB%05d%05d M%05d%05d) Memristor Ron=%d Roff=%d D↩
                    =%g p_coeff=%d \\ \n \t \t C=%g xo=x alpha=%g uv=%g",i,j, i,j, i,j, i,j, Ron, Roff,D, ↩
                    p_coff, C, alpha, uv); }
113             else if(i==m && j == n) //To cater for randomness, we need to fix 1,n & n,n so as to multiple by↩
                     Ron & Roff respectively in trans
114             {sprintf (Memristor, "m%05d%05d (netW%05d%05d netB%05d%05d M%05d%05d) Memristor Ron=%d Roff=%d D↩
                    =%g p_coeff=%d \\ \n \t \t C=%g xo=y alpha=%g uv=%g",i,j, i,j, i,j, i,j, Ron, Roff,D, ↩
                    p_coff, C, alpha, uv); }
115             else
116             {sprintf (Memristor, "m%05d%05d (netW%05d%05d netB%05d%05d M%05d%05d) Memristor Ron=%d Roff=%d D↩
                    =%g p_coeff=%d \\ \n \t \t C=%g xo=y alpha=%g uv=%g",i,j, i,j, i,j, i,j, Ron, Roff,D, ↩
                    p_coff, C,alpha, uv); }
117             netlistFile << Memristor << endl;
118          }
119          netlistFile << endl;
120       }
121       //Input voltage to the selected row on both side
122       netlistFile << "V16 (Vin 0) vsource type=pulse val1=V" << endl;
123       netlistFile << "V17 (Vin2 0) vsource type=pulse val1=V" << endl;
124
125       //Ground voltage to the selected column on both sides
126       //change to -K * V/2 instead of grounding
127       netlistFile << "V18 (Vgnd 0) vsource type=pulse val1=-0.5*V/2" << endl;
128       netlistFile << "V19 (Vgnd2 0) vsource type=pulse val1=-0.5*V/2" << endl;
```

```
129
130         for (i=0; i <= m; i=i+m)    //column  (i=0; i <= m; i=i+m)
131         {
132            for (j=1; j <= n; j++) //row (j=1; j < n; j++)
133            {
134               if (j != selectedCol)
135               {
136                  sprintf (Memristor, "VB%05d%05d (VBL%05d%05d 0) vsource type=pulse val1=V/2",i,j,i,j);
137                  netlistFile << Memristor << endl;
138               }
139            }
140         }
141         for (j=1; j <= m+1; j=j+m) //column (j=1; j <= m+1; j=j+m)
142         {
143            for (i=1; i <= m; i++) //row (i=2; i <= m; i++)
144            {
145               if (i != selectedRow)
146               {
147                  sprintf (Memristor, "VW%05d%05d (VWL%05d%05d 0) vsource type=pulse val1=0.5*V/2",i,j,i,j);
148                  netlistFile << Memristor << endl;
149               }
150            }
151         }
152         netlistFile.close();
153         oceanfile.close();
154         cout<< endl << m << "X" << n << " netlist successfull generated, check output file" << endl << endl;
155      }
156   }
```

Listing B.11: Generating netlist and Ocean Script file for simulation of write operation with line resistance in Cadence Virtuoso

```
1   float parallelRes(float R1, float R2)
2   {return ((R1*R2)/(R1+R2)); }
3
4   float parallelResArray(float *listDemo, int size)
5   {
6      if (size == 1)
7      {return listDemo[0];}
8      else
9      {
10        for (int i = size-1; i>0; i--){
11           listDemo[i-1] =  parallelRes(listDemo[i], listDemo[i-1]);
12        }
13        return listDemo[0];
14     }
15  }
16
17  float SneakPathR(float R, float m, float n)
18  {
19     if (m == 1)
20     {return (R/(n-1));}
21     if (n == 1)
22     {return (R);}
23     else
24     {return (R/(n-1)) +  (R/((m-1)*(n-1))) +  (R/(m-1));}
25  }
26
27  int main()
28  {
29     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
30     default_random_engine generator ;
31
32     float Vin = 1; float R_off = 200000, R_on = 1000; float conc = 2000;
33     long sample, m, n, size = 8; float Rl = 5;
34     float Vout = 0; float D = 3e-9; float w = 3e-9;
35     float M_final, M_init;
36     float *list;
37
38     float A = 4.2e-4; //Sensitivity parameter of sensor or constant of thin film 4.37e-3
39     float beta = 1;      //slope of Rs curve
40     if ((list = (float *)malloc(sizeof(float)*size*size+1)) == NULL) {
41        printf("unable to allocate memory \n");
42        return -1;
43     }
44     ofstream myfile;
45     myfile.open ("SensorModelDistribution.dat");
46
47     n = size;
48     float total = 0;
49     for (sample = 1; sample <= 50000; sample = sample + 1)
50     {
51        long senNum = n*n;      //n*n for MxN array, n for 1xn, 1 for single
52        float R_on_eff =  R_on / (1+A * pow(conc, beta))  ;
53        float R_off_eff =  R_off;
54
55        M_init = (R_on * (w/D)) + (R_off * (1-(w/D)));
56        for (int i = 0; i < senNum; i++)  //n*n for MxN array
57        {
58           normal_distribution<float> M_init_distribution (M_init,(5*M_init/100));   //M_init is average , (5*↩
                 M_init/100) is the deviation
59           list[i] = M_init_distribution(generator) ;
60        }
61        float Rtotal_init = parallelResArray( list, senNum); //find the total resistance of the random res
62
63        total = total + pow((Rtotal_init - M_init),2);
64
```

```
65        M_final = (R_on_eff * (w/D)) + (R_off_eff * (1-(w/D)));
66        for (int i = 0; i < senNum; i++)
67        {
68           normal_distribution<float> M_final_distribution (M_final,(5*M_final/100));
69           list[i] = M_final_distribution(generator) ;
70        }
71        float Rtotal_final = parallelResArray( list, senNum );
72        float V_out_innt = Rl/(Rtotal_init + Rl);
73        float V_out_final = Rl/(Rtotal_final + Rl);
74
75        myfile <<setprecision(8) << setw(3) << senNum  << "  " << setw(5) << conc << " " << setw(10) << M_init↩
                 << " " << setw(10) << Rtotal_init << "  "<< setw(10) << M_final << " " << setw(10) << ↩
               Rtotal_final << "  " << setw(10) << Rtotal_init/Rtotal_final << endl ;
76     }
77  }
```

Listing B.12: Computing the resistance distribution in memristor-based sensor array

```
1  //No line resistance of the 3 schemes used for the 3D plot. Was previously used for the power of all three
2  int main()
3  {
4     float Vin = 2; float Ron = 100; float Roff = 200000; int n, m;
5
6     int size = 524288;
7     //Floating, V/2 and v/3
8     float Vn, Vm, Vmn, Vn2, Vm2, Vmn2, Vn3, Vm3, Vmn3 ;
9     float Itotal, Rtotal, R;
10
11    ofstream myfile;
12    myfile.open ("write_voltages_FL_noL.dat");
13
14    for (n = 2; n <= size; n = n*2 )
15    {
16       R = Ron;
17       m=n;
18       Vn = Vin * (m-1)/(m+n-1);
19       Vm = Vin * (n-1)/(m+n-1);
20       Vmn = Vin / (m+n-1);
21       float Isneak = (Vin * (m-1)*(n-1) ) / (R * (m+n-1));  //Total sneak current. through each group
22       float Isel = Vin/R;
23       //Same current but different voltages
24       float P = (Vin * Isel) + (Vn * Isneak) + (Vmn * Isneak) + (Vm * Isneak);
25
26       Vn2 = Vin/2;
27       Vm2 = Vin/2;
28       Vmn2 = Vin/2 - Vin/2;
29       Isneak = (Vin * (m+n-2) ) / (2*R);
30
31       //total current is Isneak = (eachI=V/R) * (m+n-2) current has been multiple by the number of cells ↩
              already
32       float Pv2 = (Vin * Isel) + (Vn2 * Isneak);
33
34       Vn3 = Vin/3;
35       Vm3 = Vin/3;
36       Vmn3 = Vin/3;
37       Isneak = (Vin * (m*n-1) ) / (3*R);
38
39       //total current is Isneak = (eachI=V/R) * (m*n-1)
40       float Pv3 = (Vin * Isel) + (Vn3 * Isneak);
41
42       myfile <<setprecision(6) << setw(3) << m << "  " << setw(3) << n << "  "<< setw(8) <<  Vin/2 << "  "<<↩
                 setw(8) <<  Vn/2 << "  " << setw(8) << Vm/2 << "  "<< setw(8) << Vmn/2 << "  "<< setw(4)  << ↩
               endl ;
43
44    }
45 }
```

Listing B.13: Computing the performance metrics for crossbar write schemes