Memristor Circuits and Systems

Thesis by
Mohammed Affan Zidan

In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy

King Abdullah University of Science and Technology, Thuwal,
Kingdom of Saudi Arabia

May, 2015
The thesis of Mohammed Affan Zidan is approved by the examination committee.

Committee Chairperson: Prof. Khaled N. Salama

Committee Member: Prof. David E. Keyes

Committee Member: Prof. K.T. Tim Cheng

Committee Member: Prof. Muhammad M. Hussain

Committee Member: Prof. Aurelien C. Manchon
ABSTRACT

Memristor Circuits and Systems

Mohammed Aflan Zidan

Current CMOS-based technologies are facing design challenges related to the continuous scaling down of the minimum feature size, according to Moore’s law. Moreover, conventional computing architecture is no longer an effective way of fulfilling modern applications demands, such as big data analysis, pattern recognition, and vector processing. Therefore, there is an exigent need to shift to new technologies, at both the architecture and the device levels. Recently, memristor devices and structures attracted attention for being promising candidates for this job. Memristor device adds a new dimension for designing novel circuits and systems. In addition, high-density memristor-based crossbar is widely considered to be the essential element for future memory and bio-inspired computing systems. However, numerous challenges need to be addressed before the memristor genuinely replaces current memory and computing technologies, which is the motivation behind this research effort.

In order to address the technology challenges, we begin by fabricating and modeling the memristor device. The devices fabricated at our local clean room enriched our understanding of the memristive phenomenon and enabled the experimental testing for our memristor-based circuits. Moreover, our proposed mathematical modeling for memristor behavior is an essential element for the theoretical circuit design stage. Designing and addressing the challenges of memristor systems with practical complexity, however,
requires an extra step, which takes the form of a reliable and modular simulation platform. We, therefore, built a new simulation platform for the resistive crossbar, which can simulate realistic size arrays filled with real memory data. In addition, this simulation platform includes various crossbar nonidealities in order to obtain accurate simulation results.

Consequently, we were able to address the significant challenges facing the high density memristor crossbar, as the building block for resistive-based memory systems and neural computing. For gateless arrays, we present multiport array structure and readout technique, which for the first time introduces a closed-form solution for the challenging crossbar sneak-paths problem. Moreover, a new adaptive threshold readout methodology is proposed, which employs the memory hierarchy locality property in order to improve the access time to the memristor crossbar. Another fast readout technique based on binary counters is presented for locality-less crossbar systems. On the other hand, for gated arrays, we present new readout technique and circuitry that combines the advantages of the gated and gateless memristor arrays, namely the high-density and low-power consumption. In general, the presented structures and readout methodologies empower much faster and power efficient access to the high-density memristive crossbar, compared to other works presented in the literature. Finally, at the circuit level, we propose novel reactance-less oscillators based on memristor devices, which find promising applications in embedded systems and bio-inspired computing. Altogether, we believe that our contributions to the emerging technology help to push it to the next level, shortening the path towards better futuristic computing systems.
ACKNOWLEDGMENTS

All-Praise is due to Allah, the Lord of the worlds, for “everything”.

Thank you, my mother, my father, my wife, my brother, and my sister for all your support and encouragement throughout my life. I would not have been able to achieve anything without you being with me. Thank you for believing in me and helping me whenever I needed you.

Thank you, Professor Khaled Salama, for teaching me how to be a real researcher, who cares about the big picture, as well as the details. Your unparalleled support and guidance throughout my graduate studies are priceless. Whenever I had an idea to discuss or a technical question, I just had to pass by your office to get my problem solved.

Thank you, Professor Hossam Fahmy, for guiding me throughout my whole career. If I achieve anything in my career, it is because of you. Whenever I have a technical or even a personal problem, you are always there to help.

Thank you, Hesham Omran. You are the best office mate anyone could ever have. You are always there to help when I need you. I always enjoyed brainstorming with you, my friend.

Thank you, Professor Ahmed Radwan, for guiding me to start working on memristors and chaotic systems fields. I always enjoyed working with you and having the pleasure of learning from you.

Thank you, Professor Ahmed Sultan, for your valuable time and support. Brainstorming with you was always a joy.

Thank you, Doctor Amro Elshorafa, for all your valuable comments and feedback.
I enjoyed working with you.

Thank you, Professor Ahmed Eltawil and Professor Fadi Kordahi for all your support during my internship at UCI. I still remember how productive our meetings were and how much I benefited from them.

Thank you, Professor David Keyes, Professor K.T. Tim Cheng, Professor Muhammad Hussain, and Professor Aurelien Manchon for being on my PhD committee. Your feedback and valuable comments were of a great help.

Thank you, all my friends.

Thank you, all my professors and colleagues at KAUST, UCI, GUC, Cairo University, and IAET.
# TABLE OF CONTENTS

Examination Committee Approval .......................... 2

Abstract .................................................................. 4

Acknowledgments .................................................. 6

List of Figures ...................................................... 12

List of Tables ........................................................ 15

List of Abbreviations ............................................. 16

List of Symbols ..................................................... 18

1 Introduction ....................................................... 20

2 Memristor Modeling & Fabrication ..................... 25
  2.1 Introduction ................................................... 25
  2.2 Theoretical Definition of the Memristor ............... 25
  2.3 Device Models ............................................... 28
     2.3.1 Ideal Model .......................................... 28
     2.3.2 Dopant Drift Model ................................. 29
     2.3.3 Quantum-Tunneling Model ....................... 30
  2.4 Mathematical Analysis .................................... 31
     2.4.1 DC Input ............................................. 31
     2.4.2 Generic Periodic Waveform Input ................ 32
     2.4.3 Sinusoidal Waveform Input ....................... 34
     2.4.4 Square Waveform Input ........................... 35
     2.4.5 Triangular Waveform Input ....................... 37
  2.5 Device Fabrication .......................................... 37
  2.6 Summary ..................................................... 39
# Crossbar Sneak-Paths

3.1 Introduction .......................................................... 40
3.2 Solutions in the Literature ........................................... 41
    3.2.1 Transistor Gating ........................................... 41
    3.2.2 Nonlinear-Saturation Memristor Devices ............... 42
    3.2.3 Diode Gating ............................................... 42
    3.2.4 Back-to-Back Diodes ..................................... 43
    3.2.5 Threshold Devices ........................................ 44
    3.2.6 Complimentary Memristor Devices .................... 44
    3.2.7 Unfolded Array ......................................... 44
    3.2.8 AC Sense ................................................ 45
    3.2.9 Multistage Readout ....................................... 45
3.3 Array Connectivity ................................................ 46
3.4 Simulation Platform .............................................. 49
3.5 Crossbar Parasitic Resistance .................................. 50
3.6 Aspect Ratio Effect ............................................. 52
3.7 Summary .......................................................... 53

# Readout Techniques for High-Density Gateless Crossbar

4.1 Introduction ........................................................ 54
4.2 Multiport Readout ................................................ 54
    4.2.1 Multipoint Architecture ............................... 56
    4.2.2 Readout Technique ..................................... 57
    4.2.3 Simulation Results .................................... 62
4.3 Analysis of Sneak-Paths Correlation .......................... 66
4.4 Adaptive Threshold per Segment Readout .................... 69
    4.4.1 Readout Technique .................................... 70
    4.4.2 Multi-Read for Initial Bits ............................ 71
    4.4.3 Predefined Dummy Bits ............................... 72
    4.4.4 Simulation Results .................................... 74
4.5 Balance Counters Readout ...................................... 75
    4.5.1 Readout Technique .................................... 75
    4.5.2 Simulation Results .................................... 76
4.6 Readout Power Consumption ................................. 78
    4.6.1 Device Nonlinearity ................................... 78
    4.6.2 Array Connectivity .................................... 79
A  Device Models  140
B  Memory Simulation Platform  144
C  List of Publications  159
LIST OF FIGURES

1.1 Memristor crossbar structure. ............................................. 22
2.1 The four relations describing the basic two-terminal passive elements. . . . . 26
2.2 The current-voltage characteristic for the four basic elements in the case of a sinusoidal input voltage. ............................................. 27
2.3 Measured I-V pinched hysteresis for a memristor device. ....................... 27
2.4 The $Q - \phi$ and $I - V$ characteristics for the ideal memristor ............... 29
2.5 Structure of the linear dopant drift memristor model. ............................................. 29
2.6 Structure of the memristor quantum tunneling model. ......................... 30
2.7 Intervals of the memristor resistance. ............................................. 33
2.8 Simulated vs. calculated memristor $I-V$ hysteresis. ............................ 35
2.9 Simulated and calculated $R_M$ and $R_i$ versus time, for asymmetrical square waveform input voltage. ............................................. 36
2.10 SEM and measured I-V hysteresis for the fabricated memristor device. . . . . 38
2.11 Different layers forming our memristor device. ..................................... 39
3.1 Gateless crossbar showing desired and sneak currents. .......................... 40
3.2 Gated and Gateless arrays and their equivalent circuits .......................... 42
3.3 Simple memory array with one diode and one memristor used for each memory cell. ............................................. 43
3.4 Unfolded array structure. ............................................. 45
3.5 Simple memory array showing the added column capacitors for the AC sense. ............................................. 45
3.6 Various crossbar accessing connections and their equivalent circuits. ........ 47
3.7 Flow chart representing the implemented simulation platform ............... 49
3.8 Crossbar model per unit cell. ............................................. 50
3.9 The effect of crossbar resistance on the readout current. ......................... 51
3.10 Examples of different organizations with different aspect ratios for a 16-cell memory array. ............................................. 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>2:4 row decoder cells.</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>First three-reading combination for evaluating the desired cell resistance using multiport technique.</td>
<td>57</td>
</tr>
<tr>
<td>4.3</td>
<td>Second three-reading combination for evaluating the desired cell resistance using multiport technique.</td>
<td>58</td>
</tr>
<tr>
<td>4.4</td>
<td>A circuit realizing the threshold equation for the multiport readout.</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>HSPICE simulations showing the normalized memory readings for different array lengths.</td>
<td>64</td>
</tr>
<tr>
<td>4.6</td>
<td>The percentage of correctly read data versus the array size for different number of quantization bits, for the multiport readout.</td>
<td>65</td>
</tr>
<tr>
<td>4.7</td>
<td>The required number of bits for error free output versus the array length, for the multiport readout.</td>
<td>66</td>
</tr>
<tr>
<td>4.8</td>
<td>Maximum change in segment resistance related to its original value vs array size and the number of ones per array.</td>
<td>69</td>
</tr>
<tr>
<td>4.9</td>
<td>Equivalent circuit for the “connected terminals” accessing mode in case of forcing the rows and column terminals to the same voltage</td>
<td>71</td>
</tr>
<tr>
<td>4.10</td>
<td>Array accessing sequence for adaptive threshold readout.</td>
<td>72</td>
</tr>
<tr>
<td>4.11</td>
<td>Adaptive threshold readout overhead</td>
<td>73</td>
</tr>
<tr>
<td>4.12</td>
<td>Histogram for the readout current for the adaptive threshold readout.</td>
<td>74</td>
</tr>
<tr>
<td>4.13</td>
<td>Simulation results for multiple readouts for the adaptive threshold readout.</td>
<td>77</td>
</tr>
<tr>
<td>4.14</td>
<td>Quantized output and number of the required bits for the adaptive threshold readout.</td>
<td>78</td>
</tr>
<tr>
<td>4.15</td>
<td>Reading power consumed by a crossbar array for various device types and array connectivity methods.</td>
<td>80</td>
</tr>
<tr>
<td>5.1</td>
<td>The four different cell types building up the gated array and schematic showing the biasing condition of each of them.</td>
<td>84</td>
</tr>
<tr>
<td>5.2</td>
<td>SPICE simulation for the transistor current versus its gate voltage for a single memory cell.</td>
<td>86</td>
</tr>
<tr>
<td>5.3</td>
<td>The equivalent gated array circuit.</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>SPICE simulation for the area-power normalized figure-of-merit (FOM) versus the gate transistor technology node.</td>
<td>88</td>
</tr>
<tr>
<td>5.5</td>
<td>SPICE simulations for the reading current boundaries and interval for various data patterns.</td>
<td>90</td>
</tr>
<tr>
<td>5.6</td>
<td>Schematic and trace for the proposed compensated readout circuit.</td>
<td>92</td>
</tr>
</tbody>
</table>
5.7 SPICE for the proposed compensated readout circuit showing the capacitors voltages and the comparator output voltage. 93
6.1 Schematic of the MRLO family. 96
6.2 Transfer function showing transitions between different operating points. 98
6.3 Circuit simulation showing the tuning curve for the oscillation frequency of MRLO. 103
6.4 Spectre transient simulation results for MRLO type ‘A’. 107
6.5 Tuning curve for the oscillation frequency of MRLO type ‘A’. 109
6.6 Spectre transient simulation results for MRLO type ‘B’. 111
6.7 Tuning curve for the oscillation frequency of MRLO type ‘B’. 112
6.8 Tuning curve for the oscillation frequency of MRLO type ‘B’. 114
6.9 A Comparison of the maximum and minimum frequencies of the three types of MRLO family. 115
6.10 Regions of operation where each type of MRLO can provide the highest maximum oscillation frequency. 116
6.11 Maximum frequency of the three types of MRLO versus the memristor constant. 117
6.12 Snapshot of the oscillator PCB. 117
6.13 Measurement MRLO circuit output. 118
7.1 Random Access Memory vs. Content Addressable Memory. 120
7.2 CAM evaluation phase and the different states of a “2T2M” CAM cell. 122
LIST OF TABLES

1.1 Detailed comparison between memristor-based memory, traditional memories, and other emerging memories. .............................................. 21

6.1 All possible combinations of memristor constants and suitable configuration .............................................................. 100

6.2 Comparison between the different types of MRLOs family. .......... 114

7.1 Various CAM cells candidates. ................................................... 121
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Alternating Current.</td>
</tr>
<tr>
<td>ALD</td>
<td>Atomic Layer Deposition.</td>
</tr>
<tr>
<td>AP</td>
<td>Associative Processing.</td>
</tr>
<tr>
<td>AR</td>
<td>Aspect Ratio.</td>
</tr>
<tr>
<td>C</td>
<td>Capacitor, Capacitance.</td>
</tr>
<tr>
<td>CAM</td>
<td>Content Addressable Memories.</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal–Oxide–Semiconductor.</td>
</tr>
<tr>
<td>D</td>
<td>Diode.</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current.</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic Random-Access Memory.</td>
</tr>
<tr>
<td>FeRAM</td>
<td>Ferromagnetic Random-Access Memory.</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface.</td>
</tr>
<tr>
<td>HDD</td>
<td>Hard Disk Drive.</td>
</tr>
<tr>
<td>I</td>
<td>Electrical current.</td>
</tr>
<tr>
<td>ITRS</td>
<td>International Technology Road-map for Semiconductors.</td>
</tr>
<tr>
<td>L</td>
<td>Inductor, Inductance.</td>
</tr>
<tr>
<td>M</td>
<td>Memristor, Memristance.</td>
</tr>
<tr>
<td>MRAM</td>
<td>Magnetoresistive Random-Access Memory.</td>
</tr>
<tr>
<td>MRLO</td>
<td>Memristor Reactance-Less Oscillator</td>
</tr>
<tr>
<td>PCM</td>
<td>Phase Change Memory.</td>
</tr>
<tr>
<td>PVD</td>
<td>Physical Vapor Deposition.</td>
</tr>
<tr>
<td>R</td>
<td>Resistor, Resistance.</td>
</tr>
<tr>
<td>RAM</td>
<td>Random-Access Memory.</td>
</tr>
<tr>
<td>ReRAM</td>
<td>Resistive Random-Access Memory.</td>
</tr>
<tr>
<td>SEM</td>
<td>Scanning Electron Microscope.</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random-Access Memory.</td>
</tr>
<tr>
<td>SSD</td>
<td>Solid-State Drive.</td>
</tr>
</tbody>
</table>
T  Transistor.
TEM  Tunneling Electron Microscope.
V  Voltage.
LIST OF SYMBOLS

$\alpha$ Duty cycle.

$\gamma$ Crossbar metal thickness.

$\delta_{\psi}$ Change in the crossbar array density.

$\Delta_r$ Read current interval width.

$\zeta$ Electrical flux of the first half cycle of an input waveform.

$\zeta_{off}$ Electrical flux of the first half cycle of an input waveform required for the memristor to reach its OFF resistance.

$\zeta_{on}$ Electrical flux of the first half cycle of an input waveform required for the memristor to reach its ON resistance.

$\mu_v$ Dopant drift mobility.

$\nu_v$ Crossbar metal resistivity.

$\rho$ Memristor OFF-ON ratio.

$\tau$ Oscillation Period.

$\phi$ Electrical flux.

$\psi$ Crossbar array density.

$A$ Area.

$A_c$ Memory cell area.

$B$ Number of bits.

$C_c$ Crossbar column capacitance per unit cell.

$C_f$ Crossbar fringing capacitance per unit cell.

$C_r$ Crossbar row capacitance per unit cell.

$d$ Memristor device thickness.

$f$ Frequency.

$I_{df}$ Current passes through a fully selected desired cell.

$I_{dh}$ Current passes through a half selected desired cell.

$I_{leakage}$ Leakage current.

$k$ Memristor Constant.

$M$ MOS Transistor.
$N_{on}^a$ Number of ones per array, excluding desired row and column.
$N_{on}^c$ Number of ones per column.
$l$ Array Length.
$N_{on}^r$ Number of ones per row.
$P_r$ Average power consumption.
$Q$ Electrical charges.
$R_a$ Lumped array resistance, excluding desired row and column.
$R_c$ Lumped column resistance.
$R_{cb}$ Crossbar resistance per unit cell.
$R_d$ Difference between the maximum and the minimum memristor resistances.
$R_{hs}$ Half selected switch resistance.
$R_i$ Memristor initial resistance.
$R_m$ Memristor resistance.
$R_M$ Maximum or minimum memristor resistance for a given input waveform.
$R_{off}$ Memristor OFF resistance.
$R_{on}$ Memristor ON resistance.
$R_r$ Lumped row resistance.
$R_{sp}$ Sneak-paths resistance.
$t$ Time.
$u$ Memory cell width.
$V_b$ Bias voltage for array unselected terminals.
$w$ Memristor device doped region width.
Chapter 1:
Introduction

A memristor is a nonlinear resistor that changes its state according to the charge passing through it and retains this state after an electrical bias is removed. The memristor is widely accepted to be the fourth two-terminal passive element, alongside the resistor, the capacitor, and the inductor. The theoretical formulation of the memristor was introduced the 70s of the last century [1, 2]. While the physical properties of the device had been observed for about two hundred years [3], it was not until 2008 that a passive two-terminal physical implementation was directly related to the theory [4]. Since that time, memristors have received much attention from researchers in many different fields. Memristor devices are attractive candidates for a broad range of applications in both the analog and the digital domains [5]. Moreover, the high-density memristor crossbar is widely considered to be the key driving technology for future computer memory systems and bio-inspired computing [5–7].

Modern computer applications require larger sized and higher-performance memory and storage systems. Hence high speed, high density, and low cost per bit are the desirable properties of a memory system fulfilling rapidly increasing demands. However, there is a trade-off between these properties in the current memory system technologies. Computer architects design their systems based on a memory pyramid hierarchy. At the bottom level, there is the large yet slow permanent storage, and at the top level a small and very fast cache memory and processor registers. The goal of the memory hierarchy is to approach the performance of the fastest component and the cost of the cheapest one [8]. However, the emerging resistive memories are about to change the whole architecture.
Table 1.1 shows a detailed comparison between the current memory technologies and the emerging ones [6]. In the case of the traditional memories, there is a clear tradeoff between speed and density, where SRAM provides the highest speed with a very low density and, on the other hand, Flash memories are both very dense and very slow. However, this is not the case in the emerging technologies, where both high speed and high density could be achieved. Three of the four major emerging technologies are resistive-based memories, namely Phase Change Memory (PCM), Spin-Transfer Torque (STT), and REDOX. From an architecture point of view, there are many similarities between the three technologies. Nevertheless, the memristor is the most promising one amongst them. Memristor memory arrays are denser than those of Flash memories and Hard Disk Drives (HDD), and interestingly are of a speed comparable to the DRAM and SRAM. This is because the simple crossbar structure of a memristor-based array (Fig. 1.1), where each memory cell occupies just a few nanometers. Therefore memristors are currently being explored as the future replacement for the current CMOS-based memories and Solid State Drives (SSD) [9–20].

Another promising application for memristors is bio-inspired computing [7, 21–24]. This way of computing attempts to mimic the general way of how the biological brains process data, where parallel networks are used to execute complex operations. The human brain, for example, consumes only 20W of power [25], while trying to simulate

Table 1.1: Detailed comparison between memristor-based memory, traditional memories, and other emerging memories according to the 2011 ITRS report [6]. The abbreviations used are: T – transistor, C – capacitor, R – resistor, and D – diode. The bold font indicates the best value per row.

<table>
<thead>
<tr>
<th>Cell Element</th>
<th>Traditional Memories</th>
<th>Emerging Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SRAM</td>
<td>DRAM</td>
</tr>
<tr>
<td>Feature Size (nm)</td>
<td>6T</td>
<td>1T1C</td>
</tr>
<tr>
<td>Density (Gb/cm²)</td>
<td>45</td>
<td>36-65</td>
</tr>
<tr>
<td>Read Time (ns)</td>
<td>0.2</td>
<td>0.8 - 13</td>
</tr>
<tr>
<td>Write Time (ns)</td>
<td>0.2</td>
<td>2-10</td>
</tr>
<tr>
<td>Nonvolatile</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
‘1’ second of a biological brain activity takes ‘40’ minutes using ‘82,944’ powerful processors [26, 27]. Therefore, attempting to learn from such amazing creations could lead to more powerful and energy-efficient computing systems, and memristors could be the key technology for this. For instance, the high-density memristive crossbar is an excellent candidate for realizing synapses meshes in neural networks [28–34]. Additionally, memristive-based oscillators are recently considered the building blocks for synchronized oscillation and bio-inspired recognition and classification [35].

In addition to memory and bio-inspired computing, using memristors enriches the design possibilities of both analog and digital circuits. In the analog domain, memristors are used to build electronic oscillators [36–39], chaotic oscillators [40–42], basic analog circuits [43–47], FPGAs [48–52], programmable analog circuits [17, 53–56], hardware security [57–60], and sensors [61–64]. Moreover, in the classical computational domain memristors can be used to build computer logic and arithmetic circuits [65–73].

Memristor circuits and systems have the potential to change the way that computers and electronic systems of the future will work. However, in spite of all the attractive properties of these new devices, there are many challenges that need to be addressed before it genuinely replaces the current technologies. In order to deal with such challenges, we began by fabricating the memristor device at our local clean-room and performed
a mathematical analysis for the general behavior of the device, as presented in Chapter 2. This has led to a better understanding of the device characteristics and its design challenges. Based on the knowledge gained, we were able to address significant challenges at the system level.

The state-of-the-art high-density gate-less memristive arrays suffer from parasitic sneak-paths and leakage currents, which can ultimately ruin their operation. Such arrays are the building blocks for memristor-based memories and neural computing. To be able to deal with such problems, we built a highly descriptive simulation platform for the resistive crossbar, which can simulate realistically sized arrays filled with real memory data. This simulation platform includes various crossbar nonlinearities in order to obtain accurate simulation results, as discussed in Chapter 3. Based on this analysis and the simulations, we were able to introduce new accessing techniques to high-density crossbar arrays. The energy-aware methods proposed are able to eliminate the parasitic currents’ effect while reducing the access time of the array, aligning it in the same order of DRAM, as presented in Chapters 4 and 5.

On the circuit level, we introduced the first memristor-based reactance-less oscillators [36, 37]. In Chapter 6 we present a thorough mathematical derivation describing a complete family of memristor-based reactance-less oscillators (MRLOs) alongside the circuit realization of the oscillator using our fabricated memristor devices. MRLOs share the same applications spectrum with CMOS oscillators. However, they outperform CMOS oscillators at lower frequencies. This is an attractive frequency domain for sustainable systems with a duty cycle oscillator and biomedical application. In addition, memristor-based oscillators are essential components for synchronized oscillation bio-inspired computing.
The contributions presented in the thesis can be summarized as:

1. Introducing mathematical and behavior models for the memristor device [74–76].

2. Building an accurate simulation platform for memristor-based memories capable of simulating practical-size arrays, including many crossbar non-idealities [11,20].

3. Presenting an analysis and simulations for the memristor crossbar sneak-paths and studying the various parameters affecting them [5,11,20].


5. Introducing new readout techniques for the high-density gateless memristor crossbar, reducing both the access time and the power consumption [20,77].

6. Presenting new readout methodology and circuitry that combines the advantages of the gated and gateless memristor arrays while overcoming the leakage current effect [9,78].

7. Introducing the first memristor-based reactance-less oscillators, with potential applications in bio-inspired computing and embedded systems [36,37,39].

8. Fabricating memristor and memory devices at our local clean room [21,37,79,80].
Chapter 2:
Memristor Modeling & Fabrication

2.1 Introduction

There are different phases to building an electrical system, including design, simulation, and circuit realization. This also holds for memristor-based systems, where each of these phases require a different set of tools. In order to integrate the memristor device into the classical circuit design flow, a handy mathematical description of the device is required for the initial design phase. More descriptive circuit models are required for an accurate circuit simulation. Finally, a reliable fabrication process for the device is necessary for the circuit realization itself.

In this chapter we discuss the necessary foundation for the memristor device, and its primary modeling strategies as presented in the literature. We also propose a closed form equations description the behavior of the device for circuit design purposes. The equations provided were derived from our mathematical analysis of the device. In the last section we present the fabrication process that was followed in building the memristor device that was used in realizing the analog circuits presented in this work.

2.2 Theoretical Definition of the Memristor

Everything starts with a theoretical description of the memristor device [1,2,81]. As a basic circuit element, the memristor is defined by using a unique relation between electrical quantities. Each of the passive two terminal elements is described by using a relation between two of the four fundamental electrical quantities, the electrical flux
(\phi), the charge (Q), the current (I), and the voltage (V), as shown in Fig. 2.1 [1, 4]. The previously unused relation between the charge and the flux was the key to postulate the existence of the memristor devices in the 1970s, when the memristance was defined as [1],

\[ M = \frac{d\phi}{dq} \]  

(2.1)

This relation is translated into a pinched hysteresis loop in the current-voltage domain. This pinched curve is the fingerprint of memristor devices, where any device with the same current-voltage behavior is considered to be a memristor [82]. The hysteresis reflects the short-term memory effect of the device.

Unlike the other primary passive devices, the memristor is the only device to show a current hysteresis response to a periodic voltage input. This response cannot be achieved by any possible combination of the other two terminal passive devices. Fig. 2.2 shows the I-V relation for the four core devices in the case of a periodic voltage waveform input being applied to them. The resistor is characterized by a line relationship between the voltage and the current, which could be either linear or nonlinear, as in the case of a diode. Capacitors and inductors are energy storing elements, where a ±90 phase difference always exists. On the other hand, memristors have a variable phase difference between the voltage and the current. This is because the resistance of the device, which relates the voltage to the current, also changes. However, the device does not store any

Figure 2.1: The four relations describing the basic two-terminal passive elements [1, 4].
Figure 2.2: The current-voltage characteristic for the four basic elements in the case of a sinusoidal input voltage.

energy, while at zero voltage no current passes through it. Fig. 2.3 shows measured i-v pinched hysteresis for a memristor device that we fabricated at our local clean-room at KAUST (for more details about the device fabrication recipe please refer to Section 2.5).

The main attractive property for most of the memristor-based applications cannot be extracted from its I-V relation. This property is the device long-term memory, where

Figure 2.3: Measured I-V pinched hysteresis for a memristor device we fabricated at our local clean-room at KAUST.
the device should preserve its state after the electrical bias is removed. Theoretically speaking, a device lacking the long-term memory effect is still considered a memristor if it related the current to the voltage in a pinched hysteresis way [82]. However, devices without any long term memory effect are unusable for state-of-the-art applications of storage and for bio-inspired computing.

2.3 Device Models

Several circuit models for the memristor have been presented in the literature [76,83–91]. These models vary from being purely theoretical to very accurate physical models. In this section, we discuss the three main strategies for modeling the memristor device as presented in the literature.

2.3.1 Ideal Model

The original device introduced by Chua in 1971 was an ideal device, with only two possible values of resistance [1]. When the electrical flux (the integration of the input voltage applied to the device) increased above a certain value the device switches from one resistance to the other. It reverts again to the original resistance value after the flux falls below the same threshold. This threshold effect appears clearly in the $Q - \phi$ diagram shown in Fig. 2.4. It reflects directly on the shape of the i-v hysteresis, where it is comprised of two linear resistance values only, as shown in Fig. 2.4. Despite the simplicity of the ideal model, it did not provide an acceptable approximation of memristor response for the circuit designers. The resistance for this device model is given as,

$$
R_m = \begin{cases} 
R_a, & |\phi| \geq \phi_o \\
R_b, & |\phi| < \phi_o 
\end{cases}
$$

(2.2)
2.3.2 Dopant Drift Model

One of the simple explanations for the memristor’s property is the linear dopant drift phenomenon. The models created with this description provide a very simple and useful approximation for memristor behavior and are widely utilized by the circuit community. With the $T_iO_2$ memristor as a reference, its physical structure comprises two regions; one of these regions has highly doped titanium dioxide with oxygen vacancies ($T_iO_{2-x}$) the other has undoped titanium dioxide ($T_iO_2$). The two layers taken together have the equivalent resistance of $[4, 86]$,

$$R_m = \frac{w}{d} R_{on} + \left(1 - \frac{w}{d}\right) R_{off}$$

(2.3)

where $R_{on}$ is the ON resistance for an entirely doped device, and $R_{off}$ is the OFF resistance for a whole undoped device. The total thickness of the device is represented by $d$ and the thickness of the doped region is $w$. Fig. 2.5 shows an illustration for the device structure [4]. The thickness of the active device area is just a few nanometers. Therefore

![Diagram](image)

Figure 2.5: Structure of the linear dopant drift memristor model [4].
a vast electrical field is generated through relatively small voltages being applied. This electrical field is capable of drifting the dopant (oxygen vacancies) to increase or decrease the doped volume, based on the voltage polarity, therefore changing the total resistance of the device. After the bias is removed, the dopant retains its place, preserving the resistance of the device. Several circuit models have been presented base on dopant drift behavior [84, 86–88, 92, 93].

2.3.3 Quantum-Tunneling Model

A more realistic understanding of the memristance effect relates the change in resistance phenomenon to the quantum tunneling effect. The modulation of the width of a quantum tunnel determines the resistance of the device. Fig. 2.6 shows an abstracted illustration for the device structure. A conducting \((TiO_{2-x})\) channel creates a quantum tunnel barrier with the metal contact. The width of the tunnel varies based on the drift of the oxygen vacancies (dopant). While the dopant drift concept is still valid, the displacement distance is much smaller than that expected in the simple model. However, the change in the tunnel width produces very high ON/OFF ratios. The tunneling effect was verified using cross-sectional TEM photos, as shown [94]. The saturation values of this model can be approximated as a “\(sinh\)” function [10]. Several behavioral and circuit models based on the quantum tunneling effect are presented in the literature [87, 90, 95–97]. Although the quantum models use the most realistic explanation for the memristance

![Figure 2.6: Structure of the memristor quantum tunneling model [90].](image)
property, the simple drift models are more widely used in circuit simulations due to its simplicity and accepted accuracy.

### 2.4 Mathematical Analysis

From a circuit design perspective, there is a need for a mathematical model for the memristor that relates the resistance value to the commonly used circuit variables; current, voltage and time. In this work, we present one of the very first closed-form mathematical foundations for the memristor device [74,75]. The proposed equations provide necessary tools for initial circuit design phases.

The initial equation for our analysis is the memristor resistance as a function of time as derived in [98,99], which can generalize the resistance for linear dopant drift as,

$$R_m^2 = R_i^2 - 2kR_d\phi(t), \quad R_m \in (R_{on}, R_{off})$$  \hspace{1cm} (2.4)

where \( k = \mu_v \cdot \frac{R_{on}}{D^2} \), \( \mu_v \) is dopant drift mobility, \( \phi(t) = \int_0^t v(\tau) \, d\tau \) is the flux at time \( t \), \( R_d = R_{off} - R_{on} \) which is the difference between the boundary resistances, and \( R_i = R(x_o) \) is the initial resistance at \( t = 0 \).

### 2.4.1 DC Input

For DC input applied to the memristor, the dopant boundary will drift in one direction, reaching saturation. By evaluating the flux (\( \phi \)) in (2.4), the device resistance as a function of time can be calculated as,

$$R_m^2 = R_i^2 - 2kR_dV_{DC}t, \quad R_m \in (R_{on}, R_{off})$$  \hspace{1cm} (2.5)
where $V_{DC}$ is the value of the DC voltage. The time required for the device to reach saturation is calculated by using (2.5), and is given as,

$$t_{sat} = \begin{cases} \frac{R^2 - R^2_{off}}{2V_{DC}kR_d}, & V_{DC} < 0 \\ \frac{R^2 - R^2_{on}}{2V_{DC}kR_d}, & V_{DC} > 0 \end{cases}$$ (2.6)

where the saturation direction depends on the sign of $V_{DC}$. The maximum required saturation time, independent of the initial state of the device, is given as,

$$(t_{sat})_{max} = \frac{R_{off} + R_{on}}{2|V_{DC}|k} \approx \frac{D^2R_{off}}{2\mu_0R_{on}V_{DC}}$$ (2.7)

This is a very useful formula for digital applications where the saturation values are the region of interest.

The initial resistance for symmetric saturation times, $t_{sat} = 0.5 (t_{sat})_{max}$, in case of equal magnitude positive and negative voltage, is given by,

$$R^*_i = \sqrt{\frac{R^2_{on} + R^2_{off}}{2}} \approx \frac{R_{off}}{\sqrt{2}}$$ (2.8)

which is greater than the average value of “$0.5 (R_{on} + R_{off})$”, and the difference increases as $R_d$ increases.

### 2.4.2 Generic Periodic Waveform Input

For zero net-flux periodic signals the device may not reach the saturation values as for DC input. The maximum and minimum reachable resistance occurs at $t = (n + \alpha) \tau$, where $n = 0, 1, 2, \ldots, \infty$, $\tau$ is oscillation period, and $\alpha = V_0/(V_0 + V_1)$ where $V_0$ and $V_1$ are the peak voltages for the first and the second half cycles respectively. At these points,
the resistance extrema is given as,

\[ R_M^2 = R_i^2 - 2kR_d\zeta, \quad R_M \in (R_{on}, R_{off}) \]  \hspace{1cm} (2.9)

where \( \zeta_0 \) is the area under the curve of the voltage input (electrical flux) for the first half cycle. Saturation can be achieved with sufficient flux, such that,

\[ \zeta_{on} = \frac{1}{2k} \left( \frac{R_{on}^2 - R_i^2}{R_{on} - R_{off}} \right) > 0 \]  \hspace{1cm} (2.10)

\[ \zeta_{off} = \frac{1}{2k} \left( \frac{R_{off}^2 - R_i^2}{R_{on} - R_{off}} \right) < 0 \]  \hspace{1cm} (2.11)

where \( \zeta_{off} \) and \( \zeta_{on} \) are of the first half cycle required for reaching saturation at \( R_{off} \) and \( R_{on} \) respectively. The difference between these boundaries is given by,

\[ |\zeta_{sat}| = \frac{R_{off} + R_{on}}{2k} \]  \hspace{1cm} (2.12)

which is independent of \( R_i \). This value is considered as the minimax required half cycle area by the memristor resistance to reach saturation, either \( R_{off} \) or \( R_{on} \), when \( R_i \) equals to \( R_{on} \) or \( R_{off} \) respectively.
2.4.3 Sinusoidal Waveform Input

The device resistance for a sinusoidal waveform input can be derived from (2.4) as,

\[ R_m^2 = R_i^2 - \frac{2V_o k R_d}{\pi f} \sin^2 (\pi f t) \] (2.13)

where \( V_o \) is the amplitude of the input voltage signal and \( f \) is its frequency. The range of \( R \) depends on the sign of \( V_o \), where,

\[ R_m \in \begin{cases} [R_i, R_{off}], & V_o < 0 \\ \{R_i\}, & V_o = 0 \\ [R_{on}, R_i], & V_o > 0 \end{cases} \] (2.14)

The extreme values of ‘\( R_m \)’ are reached at \( t = (2n + 1)\tau/2 \), where \( n = 0, 1, 2, \cdots \), such that,

\[ R_M^2 = R_i^2 - \frac{2V_o k R_d}{\pi f}, \quad R_M \in (R_{on}, R_{off}) \] (2.15)

where \( R_M \) is the maximum or minimum reachable value of the device resistance under the applied sinusoidal waveform. The device resistance returns to its initial value every \( t = n \), where \( n = 0, 1, 2, \cdots \).

The implicit equation describing the \( i-v \) pinched hysteresis under a sinusoidal input is given as,

\[ R_m = \sqrt{R_i^2 - \frac{V_o k R_d}{\pi f}} \left( 1 \pm \sqrt{1 - \left(\frac{v}{V_o}\right)^2} \right) \] (2.16)

where \( v \) is the input voltage. Fig. 2.8 shows the matching between the derived formula and the device simulation using the dopant-drift circuit model presented in [86].
Figure 2.8: Memristor \( I-V \) hysteresis at frequency of 1Hz and peak voltage of \(-1.5\)V using \( R_{\text{off}} = 16k\Omega \), \( R_{\text{on}} = 100\Omega \), \( R_t = 4k\Omega \) and \( p = 10 \).

### 2.4.4 Square Waveform Input

#### Zero Net-Flux Waveform:

For a zero net-flux square waveform input, the device resistance at start of each cycle, \( t = n\tau \) where \( n = 0, 1, 2, \ldots \), is given by,

\[
R_m^2 (n) = R_m^2 (n - 1) \frac{2kR_d}{f} (V_{o1}\alpha + V_{o2} (1 - \alpha))
\]

\[
= R_i^2 - \frac{2nkR_d}{f} (V_{o1}\alpha + V_{o2} (1 - \alpha))
\]  

(2.17)  

(2.18)

where \( R_m (n) \in (R_{\text{on}}, R_{\text{off}}) \), \( R_m (0) = R_i \), \( f \) is the frequency, \( \alpha \) is the duty cycle, and \( V_{o1} \) and \( V_{o2} \) are the amplitude voltage at the first and second parts of the cycle respectively. The device resistance reaches its maximum or minimum value, at \( t = (n + \alpha) T \), where \( n = 0, 1, 2, \ldots \), as,

\[
R_m (n) = R_i^2 - \frac{2kR_d}{f} [(n + 1) V_{o1}\alpha + V_{o2} (1 - \alpha)]
\]

(2.19)

#### Non-Zero Net-Flux Waveform:

For the nonzero net-area square waveform the device resistance increase or decrease gradually until thoroughly saturated at either \( R_{\text{off}} \) or \( R_{\text{on}} \). The direction of saturation de-
pends on the sign of the total area under the curve, and the grade of the change depends on the total area magnitude. The value of the resistance at the start of each cycle, \( t = n\tau \) where \( n = 0, 1, 2, \ldots \), is given by,

\[
R_i^2 (n) = R_i^2 (n - 1) - \frac{2kR_d}{f} (V_0\alpha + V_1 (1 - \alpha))
\]

\[
= R_i^2 - \frac{2n^2R_d}{f} [V_0\alpha + V_1 (1 - \alpha)]
\]  \hspace{1cm} (2.20)

where \( R_i (n) \in (R_{on}, R_{off}) \), \( R_i (0) = R_i \), and \( f \) is the frequency. Where \([V_0\alpha + V_1 (1 - \alpha)]\) denotes the total total area under the curve, \( \alpha \) is the duty cycle, and \( V_0 \) and \( V_1 \) are the amplitude voltage at the first and second parts of the cycle respectively. The device resistance reaches its maximum or minimum value, at \( t = (n + \alpha)\tau \), where \( n = 0, 1, 2, \ldots \), as,

\[
R_M^2 (n) = R_i^2 (n) - \frac{2\alpha V_0 k R_d}{f}
\]

\[
= R_i^2 - \frac{2kR_d}{f} [(n + 1) V_0\alpha + V_0 (1 - \alpha)]
\]  \hspace{1cm} (2.21)

![Graph](image_url)

Figure 2.9: Simulated Memristor and calculated \( R_M \) and \( R_i \) versus time, for asymmetrical square waveform input voltage. \( V_0 = -2.5V, V_1 = 2.5V, \alpha = 0.52 \) and frequency = 10Hz. Memristor parameters: \( R_{on} = 100\Omega, R_{off} = 16k\Omega \) and \( R_i = 11k\Omega \). Simulation parameter \( p = 40 \).
where $R_M(n) \in (R_{on}, R_{off})$. Fig. 2.9 shows the device response based on the simple model for a nonzero net-area square waveform voltage input. The simulations are made using the SPICE model given in [86]. The parameters used are matched with the initial slow memristor reported in [4]. However, a faster response can be achieved by changing the value of $\mu_v$. Recently, devices with sub-nano second switching have been reported [100].

### 2.4.5 Triangular Waveform Input

For a symmetrical zero-net area triangle waveform input voltage the device resistance can be derived from (2.4) as,

$$
R_m^2 = \begin{cases} 
R_i^2 - 4V_0 f k R_d t^2 & 1^{st} \text{ quarter cycle} \\
R_i^2 - \frac{V_0 k R_d}{2 f} \left(1 - 8 f^2 \left(\frac{\tau}{2} - t\right)^2\right) & 2^{nd} \& 3^{rd} \text{ quarter cycles} \\
R_i^2 - 4V_0 f k R_d (\tau - t)^2 & 4^{th} \text{ quarter cycle}
\end{cases}
$$

where $R_m \in (R_{on}, R_{off})$ and $\tau$ is the period time. The maximum or minimum resistance of the device, at $t = (n + 1/2) \tau$ where $n = 0, 1, 2, \ldots$, is given as,

$$
R_m^2 = R_i^2 - \frac{V_0 k R_d}{2 f}, \quad R_m \in (R_{on}, R_{off})
$$

### 2.5 Device Fabrication

For the sake of verifying the proposed systems and circuits, we fabricated the memristor device in our local clean room at KAUST. We thus became one of the few universities around the world that have access to a physical memristor device. The device was fabricated using different materials and fabrication methodologies. The first technique adopts titanium oxide as the active material [37], where the memristor devices were fabricated on n-type <100> silicon wafers with resistivity of 10-20 Ohm-cm. After
thermal oxidation of the wafer for electrical isolation, the bottom electrode stack of 200 Å Ti/1000 Å Pt was deposited via physical vapor deposition (PVD) and patterned with conventional contact lithography using a heated aqua regia based wet etch. Photore sist strip and dilute HF surface treatment preceded 130 Å TiO₂ dielectric deposition in an Oxford FlexAL atomic layer deposition (ALD) reactor using Titanium Isopropoxide precursor and remote Oxygen plasma. Contact lithography was again employed to create a 200 Å Ti/1000 Å Pt top electrode via liftoff. Figures 2.10a and 2.10b show SEM (Scanning Electron Microscope) photos for one of the fabricated memristor devices and its cross-section respectively. The active layer of the device (TiO₂) is sandwiched between the two platinum electrodes at their intersection, as indicated in the figures. A brief etch in HF diluted 50:1 in water was used to highlight the TiO₂ layer and the other interfaces in the cross section SEM. The top and bottom platinum electrodes have intersecting fingers of 25 μm width. The fabricated devices were screened using a Keithley SCS-4200 parameter analyzer without any additional anneals or forming steps. Figure 2.10c shows the measured hysteresis of the device using a DC dual voltage sweep from 2 V to -2 V and back with 10 mA compliance current. Fig 2.11 shows the different layers forming our device and their thicknesses.

The second fabrication scheme adopts aluminum oxide as the active material [21]. The memristor material system (Al, TaN and Al₂O₃), deposition and patterning processes (sputtering and ALD) are standard CMOS processes. The maximum temperature

![Image](image-url)

Figure 2.10: Fabricated memristor device (a) SEM, (b) cross-section SEM, and (c) measured i-v hysteresis.
required for our releasing step is $300\,^\circ C$, which is relatively low.

### 2.6 Summary

In this chapter we introduced the main background of the memristor device and its modeling techniques. We proposed a new mathematical analysis of the device behavior. Finally, the fabrication process used in our device realization was discussed.
Chapter 3: Crossbar Sneak-Paths

3.1 Introduction

The main advantage of a redox memristive array is its very high density [5, 6], where each memory cell occupies a few nanometers. This is because the array is built simply as a crossbar structure, as shown in Fig. 3.1a. At the intersection of each two metal lines lies a memory cell as a thin film of a reduced oxide [101]. Such simple assembly is inherently self-aligned and can be fabricated by using just one or two lithography masks [101]. While the simplicity of the structure is its principal advantage, it is also the source of its main problem, namely the sneak-paths problem. During reading from (or writing to) a cell in the array, current should flow through the desired cell only. However, there is nothing in the crossbar to prevent current from sneaking through other cells in the array as shown in 3.1b.

Figure 3.1: (a) Gateless crossbar structure. (b) Desired (green) and sneak (red) currents flowing through the memory array.
Sneak-paths impact on the performance of a crossbar-based system in two ways. First, a considerable amount of undesirable energy is consumed while current sneaks throughout the array cells. Second, the sneak-currents cannot be predicted because they are data dependent. Data stored in a memory array is naturally random, which leads to a random resistance of the sneak-paths. In addition, this parasitic resistance depends on the location of the memory cell that is being accessed. Moreover, the magnitude of the sneak-current is much higher than the current of the desired memory cell. Taken together, these undesirable effects create a severe impact on the reading and writing operation of the array. The power consumption and the correctness of the memory access are two sides of one coin and need to be addressed together rather than optimizing one while sacrificing the other. In general, power consumption can be limited by using nonlinear memristive devices. In addition, the way the crossbar is accessed can play a major role in reducing the power consumption. Handling the sneak-paths effect in an energy aware scheme, however, is a more challenging task to solve.

3.2 Solutions in the Literature

3.2.1 Transistor Gating

A transistor-gated array mimics the classical DRAM architecture [102, 103], where the memory cell is made of one transistor and one memristor, rather than a single memristor in the gateless crossbar, as shown in Fig. 3.2. The introduced gate devices cut the undesired sneak-paths, at the expense of the array density, which is then dominated by the transistor footprint, where high-density is considered the primary advantage of the memristive crossbar. The added gate thus introduces a trade-off between the array density and the severity of the sneak-paths. In addition, smaller transistors are used to maintain the high density, and the array suffers from leakage current, which prevents the memory from working correctly (see Chapter 5).
3.2.2 Nonlinear-Saturation Memristor Devices

The voltage drop on the desired cell is higher than any of the sneak-path elements since the shortest sneak-path will contain at least three memristors in series. In [104], a high nonlinear device is reported, such that \( I(V/2) \approx I(V)/100 \) at \( V \approx 1V \). This very useful property will significantly reduce the sneak-paths current relative to the desired cell current and will reduce the sneak-path effect. However, this is not a complete cure for the problem. The effect of the device nonlinearity on the sneak-paths is studied and presented in Section 4.6.

3.2.3 Diode Gating

One of the proposed solutions for the sneak-paths is to add a diode to each memory cell [105], producing a new cell of one diode and one memristor (1D1M), as shown in Fig. 3.3. Such a strategy would eliminate sneak-paths. According to [105], adding diodes
to the array will increase the system delay by adding capacitive loads, and diode threshold voltages will then reduce the output swing. However, the major problem facing such a strategy is that it will block the writing process in the native array structure, since writing to a memristor requires two different polarities. In [106] a 3D array structure is provided to enable the write operation with a diode present. In this technique, each cell will contain one programming element, two diodes, and four connecting crossbars. While this method allows the write operation, it consumes more area per cell. In addition, the 3D alignment of four bars may reduce the array density significantly. Finally, it is unclear whether the new structure containing four bars would still eliminate the sneak-paths.

3.2.4 Back-to-Back Diodes

Implementing the memory cell selector as two parallel diodes of opposite polarities solves the write problem of the single diode technique. The two parallel devices will enhance the saturation nonlinearity of the memory cell device. However, fabricating two parallel diodes requires more silicon as compared to a single MOS transistor. Moreover, aligning the diode threshold with the memristor threshold is a very tricky process.

Figure 3.3: Simple memory array with one diode and one memristor used for each memory cell.
3.2.5 Threshold Devices

A threshold device acts as two back-to-back diodes, but with a smaller footprint [107]. Such a threshold device can be used as a gate to the memory cell to increase the saturation nonlinearity of the memristor at the expense of crossbar density. Moreover, threshold devices have their shortfalls, such as the very slow writing speed of the MIEC (Mixed-Ionic-Electronic-Conduction) devices [6].

3.2.6 Complimentary Memristor Devices

In this technique the memory cell is made of a series of two memristor devices to reduce the sneak-paths effect, as presented in [108]. The two devices need to be of different write thresholds and fabricated in opposite polarities. In addition, a complex writing mechanism is required for the complimentary structure [109]. Devices of a very high ON/OFF ratio may block the writing process. Compared to the complimentary technique, the same or a better result can be achieved by using nonlinear devices, with a much simpler fabrication and writing process.

3.2.7 Unfolded Array

This solution is presented in [105], and is based on having a separate column for each memristor, as shown in Fig. 3.4. While this solution eliminates the sneak-paths problem, it enormously reduces the memory density. The decreased density can be defined as:

\[
\psi_{uf} = \frac{\psi_o}{\# \text{ of rows}}
\]  

(3.1)

where \(\psi_{uf}\) is the new density of the unfolded architecture and \(\psi_o\) is the initial density. An array with only one row will also eliminate the effect of the sneak-paths while occupying much less area than the unfolded architecture.
3.2.8 AC Sense

Instead of using a regular DC signal an AC signal is used for sensing the data stored in the desired cell, as introduced in [110]. This technique uses load capacitance at the input of the sense amplifier to implement a low-pass filter, as shown in Fig. 3.5. The response of the filter is mainly based on the resistive value of the desired cell. However, this method adds extra complexity to the memory system since AC input and sensing are required. Moreover, this method will not be as useful for realistic sizes of memory arrays.

3.2.9 Multistage Readout

The multistage readout technique was introduced by an HP Labs team in [10]. This method requires three reading, three writing, and one comparison operation. These are
for estimating the stored value in a memristor cell. Their proposed reading procedure is given as follows:

1. *Read from the target cell*;
2. *Write “Zero” to the cell*;
3. *Read again the target cell*;
4. *Write “One” to the cell*;
5. *Read the target cell for the third time*;
6. *Compare the measured values to determine the state of the cell*;
7. *Write back the memory cell to its (assumed) original state*.

These successive readings and writings of the desired cell enable better estimation of the sneak-current. It can be also interpreted as selecting an adaptive threshold for each reading. This technique is considered lengthy, and cannot be pipelined. In addition, the high usage of write operations for each read could have a strong impact on the lifetime of the memory device, based on its endurance qualities, which is generally one of the fabrication challenges facing memristors. Finally, an erroneous estimate will cause the error to propagate in the memory data.

### 3.3 Array Connectivity

The first design decision for a crossbar-based system is how to access the memory array. The resistive crossbar array can be accessed using one of two general modes. The first approach is the “floating terminals”, where the array is accessed through the desired row and column only and all other terminals are kept floating, as shown in Figures 3.6a and 3.6b. This is the most direct and power-efficient way to access the array. However, it lacks a detailed equivalent circuit, which is necessary to deal with the sneak-path problem effectively. Fig 3.6c shows the array equivalent circuit, where ‘$R_{sp}$’ represents the equiva-
lent resistance of the crossbar sneak-paths, which are dependent on the data stored in the array. These data are translated into high and low resistance values at each row-column intersection.

The second access mode is “connected terminals”, where the unused rows and columns are connected to two common nodes as shown in Figures 3.6d and 3.6f. Such nodes could be tuned towards predefined voltages or even used as extra access points to the array [11].

The main strength of the “connected terminals” mode is its equivalent circuit, which can be mathematically modeled. This advantage comes at the expense of a slightly more complex access circuit compared to the “floating terminals”. In general, dealing with the impact of sneak-paths on the readout is easier when using the “connected terminals” structure, but this should not be at the expense of increased power consumption, which
is the other side of the coin.

The data stored in the array can be directly mapped into resistances representing the sneak-paths and the desired cell. Fig. 3.6f shows the equivalent circuit of a shorted-terminals array. The circuit is made up of four resistances, the desired cell resistance \( R_m \) and three components forming the sneak path resistance,

\[
R_{sp} = R_r + R_a + R_c
\]  

(3.2)

where \( R_r \) is the lumped “row” resistance, \( R_c \) is the lumped “column” resistance, \( R_a \) is the lumped “array” resistance, as shown in Figurers 3.6e and 3.6f. Splitting \( R_{sp} \) into three lumped components enables an independent set of equations for solving \( R_m \). Each of the three sneak-path components is composed of parallel resistances that are shorted together from both directions. For a square memory array of length \( l \) the total sneak-path resistance is given as,

\[
R_{sp} = \left[ \frac{N^r_{on}}{R_{on}} + \frac{l - N^r_{on} - 1}{R_{off}} \right]^{-1} + \left[ \frac{N^c_{on}}{R_{on}} + \frac{l - N^c_{on} - 1}{R_{off}} \right]^{-1} + \left[ \frac{N^a_{on}}{R_{on}} + \frac{(l - 1)^2 - N^a_{on}}{R_{off}} \right]^{-1}
\]  

(3.3)

where \( N^r_{on}, N^c_{on}, \) and \( N^a_{on} \) are the number of ON resistances in the parallel arrays forming \( R_r, R_c, \) and \( R_a \) respectively. The binary “ONE” is represented by an ON resistance, and the ZERO is represented by an OFF resistance. For a memory containing values other than \( R_{on} \) and \( R_{off} \), the formula given in (3.3) can simply be generalized in a summation. The effect of the crossbar resistance on the proposed model is discussed in Subsection 3.5. It should be noted the metal lines resistance is not included in the equivalent circuit for the sake of simplicity, but it is fully considered in the simulations carried out in this work, as discussed earlier.
3.4 Simulation Platform

Accurate simulation for realistic crossbar sizes is essential to address the sneak-paths challenge. In order to achieve this goal, we wrote a Python script that creates SPICE netlists for practical size memories and sweep different parameters and data patterns by calling HSPICE or Cadence APS iteratively [11]. Thus, the test memory can be filled with any required data pattern, including worst-case data (all zeros and all ones), random data, and NIST standard RAM images [111]. For the transistors used, Predictive Technology Models (PTM) were employed to simulate high-density memories with various selector sizes [112]. For the memory element, the platform allows for the plugging of any model for any two-terminal resistive device. Fig. 3.7 shows a flowchart representing the implemented simulation platform.

Our simulation platform can simulate much larger and realistic memristor arrays compared to other work presented in the literature. Moreover, it models the crossbar in a more accurate scheme, by including many non-idealities in the model, such as the crossbar impedance lines. Such accurate simulations require running around the clock

![Flow chart](image)

Figure 3.7: Flow chart representing the implemented simulation platform, where SMAC stands for SPICE Memristor Array Creator. For full SMAC code please refer to Appendix B.
on powerful workstation machines in order to obtain reliable results for realistically sized arrays. Fig. 3.8 shows the crossbar equivalent circuit per unit cell, where $R_{cb}$ is the line resistance per unit cell, $C_r$ and $C_c$ are the lines to ground stray capacitance, and $C_f$ is the fringing capacitance between the crossbar lines per unit cell. To the best of our knowledge, our simulation platform is more descriptive compared to other works presented in the literature [11].

3.5 Crossbar Parasitic Resistance

The resistance of the crossbar is an unwanted parasitic component in the memory array, and in turn for the proposed model. However, it does have the useful effect of damping the sneak-paths current. In this section, we study the impact of the crossbar resistance on the sneak-current. Moreover, we include it in all our simulations, by simulating the complete memory array rather than its equivalent circuit. For calculating the crossbar resistance component between each two adjacent cells ($R_{cb}$), we assume a square memristor cell of dimensions of $u \times u$, the separation between two cells is $u/2$, and the metal bar is of thickness $\gamma$. The resistance can be simply defined as,

$$R_{cb} = \nu \frac{3u}{2u \times \gamma} = \frac{3\rho}{2\gamma}$$

(3.4)
where $\rho$ is the resistivity of the crossbar metal. Most of the memristor arrays uses platinum ($\nu = 105n\Omega m$) as the crossbar metal with a thickness ranges from 10 to 20nm [56, 101]. This leads to a relatively high crossbar resistance per cell ($R_{cb} \approx 10\Omega$). However, arrays with similar dimensions can be built by the regular CMOS process using copper ($\rho = 16.78n\Omega m$) with a metal thickness of around 100nm. This leads to a much smaller crossbar resistance per cell ($R_{cb} < 0.3\Omega$). We also considered the worst case of $R_{cb} = 10\Omega$ in all our memory simulations. For studying the crossbar effect, we simulated memristor arrays versus array size using Synopsys HSPICE. To obtain a trend describing the $R_{cb}$ effect, we used a checkered data pattern to represent an equi-probable zeros and ones memory. A wider set of random data patterns is used in the system simulations in the next section. Since memristors fabricated for memory applications are characterized by having high ON and OFF resistances [10, 56, 101, 113], we used the values reported in [10, 101], where $R_{on} = 1M\Omega$ and $R_{off} = 1G\Omega$.

The first undesirable effect of the crossbar resistance is reducing the effective OFF/ON ratio of the memristor device, since the parasitic resistance will act as DC value added to both $R_{on}$ and $R_{off}$. Fig. 3.9a figure shows the decrease in OFF/ON ratio versus

![Figure 3.9:](a) The decrease in OFF/ON ratio versus the memristor array size for various $R_{cb}$, where $\Delta_{OFF/ON} = (\text{Apparent}_{OFF/ON} - \text{Device}_{OFF/ON})/\text{Device}_{OFF/ON}$. (b) HSPICE simulation for the current consumed by multipoint array first reading versus the memristor array size for various $R_{cb}$ at $R_{ON} = 1M\Omega$, $R_{OFF} = 1G\Omega$, and checkered data patterns, and (c) HSPICE simulation for the current consumed by multipoint array first reading versus the memristor array size for various $\{R_{ON}, R_{OFF}, OFF - ON\text{ ratio}\}$ resistances at $R_{cb} = 10\Omega$ and checkered data pattern. For all the simulations a voltage source of 1V is used.)
the memristor array size, where,
\[
\Delta_{\text{OFF/ON}} = \frac{\text{Apparent}_{\text{OFF/ON}} - \text{Device}_{\text{OFF/ON}}}{\text{Device}_{\text{OFF/ON}}} \tag{3.5}
\]

The maximum parasitic series resistance is considered in the case of Apparent_{\text{OFF/ON}}, where we assume that two series resistances of two full crossbar lengths are added. Another effect of the crossbar resistance is damping the total current consumed by memristor array. Fig. 3.9b shows the total current consumed versus the array area for different \( R_{cb} \) values. \( R_{cb} \) adds a damping effect to the sneak-current, causing it to have a saturation behavior. For ‘\( R_{cb} \)’ up to 10Ω the total current saturates below 110\( \mu \)A. This damping effect is directly proportional to the crossbar resistance as shown in the figure. Such effect has a positive and a negative side. The main advantage of the current saturation is reducing the power consumption by a memristor array; however, it narrows the gap between the ON and OFF current. This can increase the complexity of the sensing circuit. The average steady state power consumption of a 256K array is around 91.3\( \mu \)W for a voltage source of 1V, during the reading and writing processes, and this value saturates below 110\( \mu \)W for larger arrays. Finally, the effect of the ON resistance of the memristor device on the saturation current is presented in Fig. 3.9c. The figure shows that the saturation current increases with smaller device resistance and that saturation starts to appear at smaller array sizes. In general, the different simulations presented in Fig. 3.9 show that the effect of crossbar parasitic resistance, with both its advantages and disadvantages, has to be considered in all memristor memory simulations.

### 3.6 Aspect Ratio Effect

In this section we study the effect of aspect ratio on the performance of memristor array. Non-unity aspect ratio can be thought of as a useful method of approaching a sneak-path free memory. The aspect ratio of an array is defined as its number of columns to the
number of rows. Typical square arrays have unified aspect ratios. The aspect ratio of the memory array is one of the main parameters which could be used to limit the effect of the sneak-paths. A memory with just one row or one column will not suffer from sneak-paths at all since there will only be one path available for the current, as shown in Fig. 3.10. As the aspect ratio approaches unity, the possibilities for sneak-paths increase.

An unbalanced aspect ratio structure could be fabricated into a square area by folding the array in a zigzag shape. However, the primary cost of using an aspect ratio other than one is the increase in the required area for the selection and sensing of circuitry. This area could be given as,

\[
\text{Sense Circuit Area} = l \left[ \frac{A_c}{\sqrt{AR}} + A_r \sqrt{AR} \right]
\]  \hspace{1cm} (3.6)

where \( l \) is the array length, \( AR \) is the aspect ratio, \( A_c \) is the column cell area, and \( A_r \) is the row cell area.

![Figure 3.10: Examples of different organizations with different aspect ratios for a 16-cell memory array.](image)

### 3.7 Summary

The sneak-path problem is the main challenge facing the high-density memristor crossbar on the circuit and architecture levels. In this chapter we discussed the basic foundation of the problem and the primary solutions presented in the literature. In addition we presented the basic properties of our simulation platform, which account for many of the crossbar non-idealities at realistic memory sizes. Finally, we present simulations and discussions on one of the main crossbar parasitics, which is the metal lines resistance.
Chapter 4:
Readout Techniques for High-Density Gateless Crossbar

4.1 Introduction

Sneak-paths inherently exist combined with the high-density memristor crossbar. However, this parasitic effect can be avoided by using unconventional array accessing techniques without impacting the array density. New sneak-paths that are immune to readout techniques are required for the high-density crossbar to be functional. Optimally, these new readout strategies should be as fast as the DRAM readout and should only consume the minimum power possible. Both factors should be considered simultaneously and not at the expense of each other. This can be achieved by utilizing various techniques, as presented in this chapter. We propose three new readout methods that utilize new structures, memory hierarchy properties and sneak-path correlation, which is presented in this work. The new techniques presented in this work consume less power and are faster than the state-of-the-art readout methodologies presented in the literature.

4.2 Multiport Readout

Reading the data stored in a memristor array is conventionally done in the form of resistive sensing between the selected row and column, as shown in Fig. 3.6a and 3.6b. This resistive sensing can be achieved using various voltage or current based techniques. However, the equivalent circuit for such an array consists of two parallel resistances,
one for the desired cell \((R_m)\) and another for the unknown sneak-paths resistance \((R_{sp})\), as shown in Fig. 3.6c. Therefore, trying to estimate the value of \((R_m)\) is equivalent to solving for two independent unknowns in one equation, where a unique solution is not available. Moreover, the estimated value of the desired resistance based on a single reading is useless, since as the array size increases, the value of \(R_{sp}\) dominates the total resistance, thus significantly impacting the reliability of any estimate.

An improved estimate of the stored data \((R_m)\) can be achieved using multiple observations [11]. The goal is to have increased information for better estimates, or even to create a set of equations with independent unknowns \((R_{sp} \text{ and } R_m)\) that can be solved exactly. However, not all multi-observation strategies lead to an exact solution since it is difficult to create an independent set of equations. Multiple observations for the array content can be made either in the time or the spatial domains. Time domain multiple observations are not practical for memory applications because multiple memory writings need to occur between every two successive observations. However, intentional editing of the memory data may help. In [10], a multistage reading technique is introduced, where three reading, three writing, and one comparison operations are used for estimating the stored value in a memristor cell. Their proposed reading procedure is given as:

1. Read from the target cell;
2. Write “Zero” to the cell;
3. Read again the target cell;
4. Write “One” to the cell;
5. Read the target cell for the third time;
6. Compare the measured values to determine the state of the cell;
7. Write back the memory cell to its (assumed) original state.

These successive readings and writings of the desired cell enable better estimation of the sneak-current.
4.2.1 Multipoint Architecture

Spatial based observations can be realized by using a multipoint architecture. One of the simplest techniques to create a multipoint design is shorting all the unselected columns together and all the unselected rows together. Hence, the memory array will now have four access points, as shown in Fig. 3.6d and 3.6e. These multiple points enable multiple observations and will allow more information for solving the value of $R_m$.

Shorting the row and the column can be fabricated as an extra row or column, or as a higher metal layer. For the first option, the decrease in density as a result of having an extra row and column per array is given as,

$$\delta_\psi = \frac{2l - 1}{l^2} \approx \frac{2}{l}$$  \hspace{1cm} (4.1)

where $\delta_\psi$ is the decline in the array density and $l$ is length of the array. $\delta_\psi$ equals 0.195% for array size of 256kb. For shorting the array terminals, the available row and column select circuits can be adopted. The unselected rows will be switched to a common bar, and so will the unselected columns, instead of being left floating. Fig. 4.1 shows both the original and the modified 2:4 row decoder cell, where the inverter is transformed into an analog MUX. The same concept can be used for column selection.

![Diagram of multipoint architecture](image)

*Figure 4.1: 2:4 row decoder cell. Typically $V_{n1} = V_{DD}$ and $V_{n3} \rightarrow \text{GND}$.*
4.2.2 Readout Technique

The proposed detection concept is based on using multiple readings to evaluate the desired cell resistance \( R_m \). By using the introduced circuit model, the different readings are represented as functions in the four circuit unknowns \( (R_m, R_r, R_a, \text{ and } R_c) \). For solving the four unknowns, four different readings are required, where each reading is a ‘resistive sensing’ between two of the four array nodes. The total number of possible readouts are six \((R_{1,2}, \ldots)\), where the numbers represent the node number. However, we are interested only in calculating \( R_m \), which we will call the main variable, and refer to the other three variables as auxiliary variables. Hence, solving the main variable requires fewer readings and equations. This can be achieved by lumping two of the auxiliary variables in all of the equations, as is next shown, where three readings are sufficient. The three readings need to be selected in such a way that they are represented in three independent equations after lumping together two of the auxiliary variables. Fig. 4.2 shows one of the possible three reading combinations for calculating \( R_m \), where their equivalent resistances are,

\[
R_{1,2} = \frac{R_m R_c + R_m R_a + R_m R_r}{R_m + R_c + R_a + R_r} \tag{4.2}
\]

![Diagram](image)

Figure 4.2: A three-reading combination for assessing the desired cell resistance. The green and the red colors represent the two parallel current paths.
Figure 4.3: A second three-reading combination for assessing the desired cell. The green and the red colors represent the two parallel current paths.

\[
R_{1,4} = \frac{R_t R_m + R_r R_c + R_s R_a}{R_m + R_c + R_a + R_r} \tag{4.3}
\]

\[
R_{2,4} = \frac{R_m R_c + R_m R_a + R_r R_c + R_r R_a}{R_m + R_c + R_a + R_r} \tag{4.4}
\]

where \(R_{1,2}\), \(R_{1,4}\), and \(R_{2,4}\) are the sensed resistances of readings ‘1,2’, ‘1,4’, and ‘2,4’ respectively. The number of variables can be reduced into three only by lumping the two auxiliary variables, \(R_a\) and \(R_c\) into \(R_r\).

**Solving for General \(R_m\):**

Without applying any constraints on the possible values of the unknowns, the number of unknowns in the three nonlinear equations (4.2, 4.3, and 4.4) can be reduced into three only by letting,

\[
R_x = R_a + R_c \tag{4.5}
\]

Consequently, the equations can be uniquely solved for ‘\(R_m\)’ as,

\[
R_m = \frac{1}{2} (R_{1,2} - R_{1,4} - R_{2,4}) - \frac{2R_{1,4}R_{2,4}}{R_{1,2} - R_{1,4} - R_{2,4}} \tag{4.6}
\]

Alternatively, the desired cell resistance can be calculated using a different combination of readings, as shown in Fig. 4.3. Using the readings ‘1,2’, ‘2,3’, and ‘1,3’, \(R_m\) can be
solved as,

\[
R_m = \frac{1}{2} (R_{1,2} - R_{2,3} - R_{1,3}) - \frac{2R_{2,3}R_{1,3}}{R_{1,2} - R_{2,3} - R_{1,3}}
\] (4.7)

where \( R_{1,2}, R_{2,3}, \) and \( R_{1,3} \) are resistances of readings ‘1,2’, ‘2,3’, and ‘1,3’ respectively.

Equations (4.6) and (4.7) are general solutions for \( R_m \) without considering any extra information or boundary conditions that exist due to the actual circuit implementation and/or operation. In the following sections, we show that a considerable reduction in complexity can be achieved by constraining the possible values of the unknowns using information about the operation of the system.

**Solving for Binary \( R_m \):**

For a binary memory system, each memristor cell is written as one of the two saturated values representing zeros and ones, \( R_m = \{R_{on}, R_{off}\} \), where \( R_{on} \) is the minimum device resistance and \( R_{off} \) is the maximum one. Therefore, the calculated value of (4.6) or (4.7) will be compared with a threshold to estimate the desired cell value. A more efficient technique is to do a first stage off-line thresholding based on the equation before calculating the value of \( R_m \). For instance, solving equation (4.6) for \( R_{1,2} \), such that,

\[
R_{1,2} = R_{1,4} + R_{2,4} + R_m \pm \sqrt{4R_{1,4}R_{2,4} + R_m^2}
\] (4.8)

By considering the binary nature of \( R_m \), equation (4.8) has four possible solutions. However, not all of these four solutions are valid for a resistive array system, since,

\[
\{R_{1,2}, R_{1,4}, R_{2,4}, R_m\} \geq 0
\] (4.9)

In addition, the first reading is always smaller than \( R_m \),

\[
R_{1,2} = R_m \parallel R_{sp} < R_m
\] (4.10)
Using the constraints given by equations (4.9) and (4.10), equation (4.8) will have two possible solutions only such that,

\[ R_{1,2}(R_{on}) = R_{1,4} + R_{2,4} + R_{on} + \sqrt{4R_{1,4}R_{2,4} + R_{on}^2} \]  
(4.11)

\[ R_{1,2}(R_{off}) = R_{1,4} + R_{2,4} + R_{off} + \sqrt{4R_{1,4}R_{2,4} + R_{off}^2} \]  
(4.12)

By applying the different constraints, the possible relation between the three readings is restricted by the two equations (4.11) and (4.11). A threshold equation can be defined as located between the two surfaces representing Ron and Roff cases. The threshold equation needs to satisfy the boundary conditions between the two surfaces, which are defined as,

\[ R_{1,2} = R_{1,4}, \text{ at } R_{2,4} \to 0 \]  
(4.13)

\[ R_{1,2} = R_{2,4}, \text{ at } R_{1,4} \to 0 \]  
(4.14)

It should be noted that the readings are distributed around the symmetry plane \( R_{2,4} = R_{1,4} \), and the boundary conditions are only open interval limits. A threshold plane which satisfies the two equations (4.13) and (4.14) is defined as,

\[ R_{1,2} = R_{1,4} + R_{2,4} \]  
(4.15)

This threshold can be realized using a very simple circuit as shown in Fig. 4.4. The figure demonstrates that such a circuit can be implemented using a 3-operand adder/subtractor, where,

\[ R_t = R_{1,4} + R_{2,4} - R_{1,2} \]  
(4.16)

where the output is detected as \( R_{on} \) for \( R_t \) greater than a constant threshold, and \( R_{off} \) otherwise. The output of the addition operation is then compared with a fixed threshold, which can be realized using a simple circuit. The circuit can also be implemented
using sequential 2-operand adder/subtractor, since the different readings are captured in a series. The proposed multipoint reading procedure is summarized as follows:

1. **Read the desired cell using the ports ‘1’ and ‘2’**;
2. **Read the desired cell using the ports ‘1’ and ‘4’**;
3. **Read the desired cell using the ports ‘2’ and ‘4’**;
4. **Take a decision based on calculating $R_t$.**

Compared to the multistage reading technique introduced by HP Labs in [10], our proposed system requires three readings only while the multistage reading requires three writings and three readings. Moreover, the proposed method significantly reduces the write cycles of the memory and so increases its endurance lifetime compared to the multistage reading. This is a valuable property, knowing that device endurance is one of the main challenges facing memristor fabrication, where the reported high endurance devices have order of magnitude less endurance than the DRAM [113, 114]. Another advantage of the proposed technique is that its decision stage can be pipelined. This is not true for the multistage reading, since in the multistage procedure the final writing to the memory occurs after the decision stage. Further, in [10] an erroneous reading will corrupt the cell for all the following readings, since a false estimated value is written back to the cell. This is not the case for the multipoint technique, however. It is worth noting that the fourth step in the multipoint readout scheme is equivalent to the sixth step of the multistage reading [10] requiring a similar complexity circuit. However, our technique uses a fixed threshold comparison as opposed to the variable threshold used in the multistage reading. It should be noted here that the memristor devices fabricated for memory applications are engineered to have a writing threshold [10], and reading with
a voltage below this threshold will not affect the stored data. For other threshold-less devices, a zero net flux reading strategy can be used [16].

4.2.3 Simulation Results

In order to verify the proposed concept, the memristor memory system is simulated using Synopsys HSPICE, where the whole array is simulated rather than just its equivalent circuit. This enables the inclusion of all the unmodeled or overlooked parasitic effects in our simulations. It also allows capturing of the crossbar resistance effect precisely, where the worst case of $R_{cb} = 10\Omega$ was used in our simulations. Furthermore, we include the effect of the switches used to connect the array terminals to the shorting bars in our simulations. For estimating the switch resistance, we used $32nm$ NMOS devices provided by the Predictive Technology Model (PTM) [112]. According to HSPICE simulations, using minimum size devices ($W = L = 32nm$), the ON resistance of each device is less than $5k\Omega$. However, we used a more conservative value of $10k\Omega$ in our memory simulations. Including the different nonlinearities leads to a very long simulation time.

A Python script was written to create the SPICE netlist and to do sweeps over the area and different data patterns by calling HSPICE iteratively. All the simulations used 64 data patterns, one checkered, and 63 pseudo-random data. This is a total of more than 8K simulation runs of the whole set of arrays. In general, binary coding used within the computer systems is characterized by having equal probability for each of its two symbols. Therefore, the random patterns used are set to have equiprobable chance of zeros and ones, to mimic realistic data distributions. For more details about the simulation platform, please refer to Section 3.4.

As with other emerging memory technologies (e.g. PCM [115]), the memristor hierarchy was assumed to be similar to DRAM. This hierarchy includes banks, blocks, sub-blocks to increase the memory bandwidth and reduce the parasitic effects, such as the capacitive loading of long crossbars [116, 117]. Other techniques used to reduce the
effective size of an array, e.g., folding arrays and open digilne [117], can be borrowed from DRAM architectures. For DRAM memory, the largest continuous array is about 256Kb and circuitry is used to connect arrays building larger memory blocks [116]. Coping with memory splitting strategies, we believe that simulating continuous memristor array up to the size of 1Mb is sufficient, which we used in all the simulation presented in this work.

The primary goal of the multipoint system is to eliminate the effect of the sneak-paths. Those sneak-paths cause zeros and ones reading to be mixed, where no single threshold can be defined at the design stage. This effect is represented using HSPICE simulations in Fig. 4.5a, where no single threshold can be distinguished from the different memory readings. The figure shows normalized values for memristor memory readings in case of either “One” or “Zero” stored in the desired cell for different array sizes. On the other hand, applying equation (4.16) changes the picture completely, where this equation is a single addition/subtraction for three memory readings. Fig. 4.5b shows the normalized value of $R_v$ versus the array size, and how equation (4.16) correctly ordered all the readings into two distinguishable sets. A single fixed threshold can then be defined for the memory system. It should be noted that a reduced data set is used in Fig. 4.5 for the sake of aiding visualization.

**Decision Circuit:**

The proposed decision system can be directly implemented by quantizing three readings and applying the digital realization of equation (4.16) on the quantized values. This relatively small digital circuit can be directly added to the memory pipeline. The number of bits used for quantizing the sensed analog readings is one of the main design aspects of the multi reading array because it affects the percentage of correctly evaluated readings. The number of required bits for having a 100% correct output is directly proportional to the array size. As this magnitude increases, the impact of the sneak-paths noise becomes
Figure 4.5: HSPICE simulations showing the normalized memory readings in the case of either “One” or “Zero” stored in the desired cell for different array lengths \( L \) for, (a) Reading between the nodes “1” and “2” \( (R_{1,2}) \), and (b) \( R_t \). The array size equals \( L^2 \). The normalization factors for \( R_{1,2} \) are \{18.1, 11.8, 9.9, 9.4\} \times 10^3\), for the array lengths of 256, 512, 768, and 1024 respectively.

more dominant. Therefore, more bits are required to correctly evaluate the stored data in larger arrays. In other words, as sneak-paths noise increases with the increase of the array size, we need to decrease the quantization noise for a correct detection. The simulations show that, for a given array size, an error-free output can be achieved using a sufficient number of quantization bits, as shown in Fig. 4.6. Fig. 4.7 shows the required number of bits for an error free output versus the array length, where the array size is the square of its length. It should be noted that ADCs are typically designed to quantize the dynamic range of a signal after normalization [118]. The DC value should be selected at the design phase of the ADC. This helped in reducing the required number of bits.

Fig. 4.7 shows that for an array size of 256Kb eight quantization bits are required,
which are used in the circuit area estimation. This array size is selected since it is a typical number used for a continuous array of memory as discussed before [116]. The decision circuit is made of two main blocks, the ADC and the digital circuit. These two blocks need to work at a frequency that enables a fast reading operation. The International Technology Roadmap for Semiconductors (ITRS) report the current REDOX memory (including memristor) reading delay to be less than 50ns [6]. To comply with such number, a 100MS/s ADC would be sufficient. However, The current state-of-the-art of the ADC designs is at least four times faster speed than such number. In [119] a 65 nm 8-bit 400MS/s SAR ADC is reported to have a total area of $2.4 \times 10^{-2} \text{mm}^2$. According to the empirical equation scaling equation proposed in [120], the estimated area for 100MS/s version of the ADC is around $1.65 \times 10^{-2} \text{mm}^2$ in area. This is considered as 0.016% of a memory die area, compared to Micron’s 78nm 1Gb DDR3 of 102mm$^2$ die area [121]. This overhead area should be multiplied by the number of ADCs. The power consumption for the ADC is around 1.34mW for 100MS/S operation mode [119, 120]. It should be noted that one ADC is required for each column decoder, and a single ADC is active per each memory bank at a given time. In addition, the required ADC constraints are simpler than the one reported in [119]. In addition, according to the empirical equation proposed in [120], using a more recent technology as 22 nm can scale down a SAR ADC area by a factor of 4.6x, where the new area should be around $3.59 \times 10^{-3}\text{mm}^2$. 

![Figure 4.6: The percentage of correctly read data versus the array size for different number of quantization bits, for the multiport readout.](image_url)
The digital circuit was written in Verilog HDL and synthesized using Cadence RTL compiler and TSMC 65nm standard cell libraries. The circuit is designed as a 3-operand adder/subtractor combinational circuit, where the inputs are three 8-bit registers. The output of the adder/subtractor is then compared with a hardwired threshold as shown in Fig. 4.4. The estimated area for the synthesized circuit is $3.2 \times 10^{-4}$mm$^2$. This area is considered as $3.14 \times 10^{-4}\%$ of a memory chip area, compared to Micron’s 78nm 1Gb DDR3 of 102mm$^2$ die area [121], which is considered as negligible overhead. An even smaller area can be reported if a more recent fabrication node is used. The digital circuit power consumption is in the order of a few microwatts at 100MHz frequency and 65nm node [122].

### 4.3 Analysis of Sneak-Paths Correlation

One of the main advantages of the connected terminals structure is its simple model. In our case of biasing all the unselected terminals at $V_{DD}/2$, the sneak-paths resistance will be made of $R_r$ and $R_c$ only, while $R_a$ is shorted out, since nodes $n_3$ and $n_4$ (Fig. 3.6) are connected to the same potential level. Hence, only the cells that belong to the desired rows and columns will contribute in the sneak-paths, where all of them will have a potential drop of $V_{DD}/2$. The sneak-paths component due to the accessed row
$(R_r)$ is a parallel combination of all the row cells except the desired one, which will be given by,

$$R_r = \left( \frac{\sum_{i=1}^{l-1} 1}{R_{x1}} \right)^{-1}$$

(4.17)

where ‘$R_x$’ is the resistance of one-row cell. The row cell resistance can be either ‘$R_{on}'$’ or ‘$R_{off}'$, where these are the ON and OFF resistance of the device under ‘$V_{DD}/2$’ voltage drop respectively. The row resistance can be rewritten as,

$$R_r = \left( \frac{N_{on}}{R_{on}} + \frac{l - N_{on} - 1}{R_{off}} \right)^{-1}$$

(4.18)

$$= \frac{R_{on}R_{off}}{(l - 1)R_{on} + N_{on}(R_{off} - R_{on})}$$

(4.19)

where ‘$N_{on}$’ is the number of ON cells within the accessed row while not counting the accessed cell itself. The sneak-paths component due to the accessed column ($R_c$) can be derived similarly.

For instance, the sneak-paths row resistance at two different locations of the same row will have all the cells in common except the two cells which are swapped because of the accessed location. The swapped cells could be of the same value, and in this case the segment resistance (‘$R_r$’ or ‘$R_c$’) will retain their exact values. Otherwise, there are two cases for swapped cells storing different bits.

**Case 1: Desired cell is swapped from ON to OFF**

In this case, the number of ones per row (or column) got incremented, and the number of zeros got decremented. The new ‘$R_r$’ is given as,

$$R_{new} = \left( \frac{N_{on} + 1}{R_{on}} + \frac{l - N_{on} - 2}{R_{off}} \right)^{-1}$$

(4.20)

$$= \frac{R_{on}R_{off}}{(l - 2)R_{on} + N_{on}(R_{off} - R_{on}) + R_{off}}$$

(4.21)
The relative change in \( R_r \) is then defined as,

\[
\frac{\Delta R_r}{R_r} = \left| \frac{R_r^{\text{new}} - R_r}{R_r} \right| = \left| \frac{R_r^{\text{new}}}{R_r} - 1 \right| \quad (4.22)
\]

By substituting (4.19) and (4.21) into (4.22), the relative change in \( R_r \) is rewritten as,

\[
\frac{\Delta R_r}{R_r} = \left| \frac{(l - 1) R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}})}{(l - 2) R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}}) + R'_{\text{off}}} - 1 \right| \quad (4.23)
\]

\[
= \left| \frac{R'_{\text{off}} - R'_{\text{on}}}{(l - 2) R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}}) + R'_{\text{off}}} \right| \quad (4.24)
\]

For \( (R'_{\text{off}} \gg R'_{\text{on}}), (l \gg 1) \) and \( (N_{\text{on}} \gg 1) \), the relative change in \( R_r \) can be approximated as,

\[
\frac{\Delta R_r}{R_r} \leq \left| \frac{\rho}{l + (N_{\text{on}} - 1) \rho} \right| \quad (4.25)
\]

where \( \rho \) is the OFF/ON ratio of the used device.

**Case 2: Desired cell is swapped from OFF to ON**

In this case, the number of ones per row (or column) got decremented, and the number of zeros got incremented. The new \( R_r \) is given as,

\[
R_r^{\text{new}} = \left( \frac{N_{\text{on}} - 1}{R'_{\text{on}} + l - N_{\text{on}} R'_{\text{off}}} \right)^{-1}
\]

\[
= \frac{R'_{\text{on}} R'_{\text{off}}}{l R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}}) - R'_{\text{off}}} \quad (4.26)
\]

By substituting (4.19) and (4.27) into (4.22), the relative change in \( R_r \) is rewritten as,

\[
\frac{\Delta R_r}{R_r} = \left| \frac{(l - 1) R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}})}{l R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}}) - R'_{\text{off}}} - 1 \right| \quad (4.28)
\]

\[
= \left| \frac{R'_{\text{off}} - R'_{\text{on}}}{l R'_{\text{on}} + N_{\text{on}} (R'_{\text{off}} - R'_{\text{on}}) - R'_{\text{off}}} \right| \quad (4.29)
\]
Figure 4.8: Maximum change in segment resistance related to its initial value versus (a) the array size with balanced ones and zeros, and (b) percentage of ones for an array of 256kb of size. The ‘\(\Delta R / R\)’ represents both ‘\(\Delta R_r / R_r\)’ and ‘\(\Delta R_c / R_c\)’.

For \((R_{off} \gg R_{on}), (l \gg 1)\) and \((N_{on} \gg 1)\), the relative change in ‘\(R_r\)’ can be approximated as,

\[
\frac{\Delta R_r}{R_r} \leq \left| \frac{\rho}{l + (N_{on} - 1) \rho} \right| \quad (4.30)
\]

where ‘\(\rho\)’ is the OFF/ON ratio of the used device. This is the same expression derived for the first case in (4.25)

The good news is that ‘\(R_r\)’ and ‘\(R_c\)’ are almost constant over the same row or column respectively, as shown by (4.25) and (4.30). The maximum relative change in the row resistance versus the array size for a balanced number of zeros and ones is plotted in Fig. 4.8a. The figure shows that as the array size increases the effect of a single bit swap decreases. The other parameter that affects \(\Delta R / R\) is the number of ones, as given by (4.25). Fig. 4.8b shows that the maximum relative change of sneak-paths resistance is still small while the percentage of ones per row/column is swept, for a 256kb array.

### 4.4 Adaptive Threshold per Segment Readout

Sneak-paths correlation property can be effectively utilized in case of sequential reading for the stored data on an array. The good news is that this is the typical memory access scheme in computer systems. The cache fetches a block of data from the RAM, as well
as RAM itself, which fetches a block of data from the hard disk drive (HDD). This is possible because of the data locality property. When a bit location is accessed, its neighborhoods are likely to be accessed too [8]. Thus, data is transferred and shared between different memory layers as a block of contiguous bits, rather than random bits or words. Therefore, we know that the memory is accessed to read or write blocks of adjacent memory cells in sequence.

The locality property is of help only if the knowledge gained from reading a single bit can be adopted in reading its neighborhoods. This is the case for “connected terminals” crossbar, where the values of ‘\(R_r\)’ and ‘\(R_c\)’ can be safely shared over the same row or column respectively, as discussed in the previous sections. This is equivalent to defining an adaptive threshold that changes at each new row readout, which can be achieved with the aid of “connected terminals” crossbar structure.

4.4.1 Readout Technique

The generic “connected terminals” circuit model shown in Fig. 3.6f can be simplified for the case of ‘\(V_B\)’ terminals bias. Terminals ‘\(n_3\)’ and ‘\(n_4\)’ are connected to ‘\(V_B\)’, and terminals ‘\(n_1\)’ and ‘\(n_2\)’ are connected to ‘\(V_{DD}\)’ and virtual ground, which can be done using two different implementations as shown in Fig. 4.9. Using a virtual ground sensing circuit forces all the array elements to have a defined voltage drop independent of the data stored in the array. The desired cell experiences a full ‘\(V_{DD}\)’ voltage drop, while the sneak-paths components of ‘\(R_r\)’ and ‘\(R_c\)’ have only half of this voltage drop. Because of the device saturation nonlinearity, the full voltage drop on the desired cell makes the magnitude difference between its ON and OFF states much larger than any error introduced by sharing ‘\(R_r\)’ or ‘\(R_c\)’ over a segment. While both of ‘\(R_r\)’ and ‘\(R_c\)’ drain parasitic sneak-current, the current leaked through only one of them affects the correctness of the readout operation. In case of connecting the read circuit to node ‘\(n_1\)’, as shown in
Fig. 4.9a, the sense current \(I_{\text{sense}}\) is defined as,

\[ I_{\text{sense}} = I_m + I_r \]  

where ‘\(I_m\)’ is the desired current and ‘\(I_r\)’ is the row sneak current component. Sensing from node ‘\(n_2\)’ swaps the locations and the role of ‘\(R_r\)’ and ‘\(R_c\)’ in the circuit, as shown in Fig. 4.9b. The sense current is shifted from its desired value by the sneak-current of the row or the column. However this shift is constant with a given row or column, based on the connection orientation.

### 4.4.2 Multi-Read for Initial Bits

In general each bit has two unknowns, ‘\(R_m\)’ and ‘\(R_r\)’ (or ‘\(R_c\)’). Without adopting sneak-paths correlation and locality, multiple access stages are needed to estimate the bit value. However, a faster readout can be achieved by categorizing the bits into two types. The “initial bits”, which are the first bits accessed in a given column, and “regular bits”, which are any other bits in the array. To estimate the value of the “initial bit” two unknowns need to be solved, the desired resistance (\(R_m\)) and the row sneak resistance (\(R_r\)). However, the remaining bits in the row share the same ‘\(R_r\)’ value, and ‘\(I_r\)’ is treated as a

![Diagram of equivalent circuit for the "connected terminals" accessing mode in case of forcing the rows and column terminals to the same voltage, where the sense circuit is connected either to the desired (a)row or (b) column.](image)


threshold for a given row. To estimate the initial bit, any of the readout techniques presented in the literature can be adopted [5]. These “initial bits” readout dictates the threshold used for the rest of the bits in the same row. Fig. 4.10a shows the readout sequence for the array in case of adopting “initial bits” strategy. Therefore, the first (initial bit) could be any bit in the array, which requires ‘n’ stages of reading, then the rest of the bits in the same row are accessed in sequence, only one time for each. Reading from the next row requires a new “initial bit”, which is the first bit in the row, in this case, as shown in Fig. 4.10a. The same sequence is followed until the fetched data block for the cache is completed, i.e., each row contains one “initial bit”, and the rest of the bits are accessed in a single stage fashion. It should be noted that in the case of sensing from ‘n₁’ data is accessed in a column-wise scheme.

4.4.3 Predefined Dummy Bits

A more time efficient way to estimate the adaptive threshold is to add “dummy bits” with predefined value to the array. The general concept of adding predefined bits to an array for sneak-paths estimation is presented in [77]. In our case, for a “dummy bit” the

Figure 4.10: (a) Array accessing sequence, where the initial bit per row/column is accessed ‘n’ times, while the rest of the bits in the same row/column are accessed for one time. (b) The accessing sequence in case of using predefines “dummy bits”, where all the bits of the array are accessed in a single stage fashion. d: dummy bit, i: initial bit, and r: regular bit.
value of $R_m$ is known in advance, and a single readout is needed to estimate the value of $R_t$. This estimated value is reused with the other bits in the same row, where, in this case, a single readout is required to estimate the remaining unknown ($R_m$). This value is used for the rest of the bits in the same row. The “dummy bit” can be organized in several fashions, given that each row contains a single bit. Fig. 4.10b shows a possible organization of dummy bits which is suitable for row-wise readout analogy.

The “dummy bits” technique adds a small overhead on the readout process, since a “dummy bit” needs to be accessed a single time compared with ‘n’ times for an “initial bit”. However, for practical size arrays with 256k size or more, the average number of array accesses per bit is almost one for both methods, in case of fetching a block of data from memory. Fig. 4.11a shows the average number of readouts per memory bit, where the overhead is shared over “regular bits”, versus the fetched data size. The figure shows how the average number of readouts converges to one very fast. The rebels in the curve are because that start reading from a new row adds an extra overhead of an “initial bit” or a “dummy bit”. It should be noted that the typical cache line is 0.5kb (64 bytes), where multiple lines are fetched from memory in sequence based on the cache policy. This value is much larger in the case of RAM fetching from HDD. While “dummy bits”

![Figure 4.11](image_url)

Figure 4.11: (a) The average number of readouts versus the fetched data size for the adaptive threshold techniques, where the first accessed bit is the middle of a row. (b) The percentage of “dummy bits” versus the array size.
technique shows a better behavior, it comes at a small expense of the effective area of the array, since “dummy bits” are not used to store real data. This negligible overhead is shown in Fig. 4.11b

### 4.4.4 Simulation Results

In order to verify the proposed concept, we simulated the readout operation at different locations of a 256kb array of various NIST RAM images. In the first case, the readout locations are distributed over the array while in the second one all the readout are made for cells in the same column. Fig. 4.12 shows the histogram of the sensed read current in the two cases. The results shows that normally the distributions of reading ONE and reading Zero are highly overlapped, and it is not possible to define a threshold to distinguish between the two binary cases, as shown in Fig. 4.12a. However, for a given row or column, reading from different locations express a clear separation between the ones and zeros distribution, as shown in Fig. 4.12b. This verifies our proposed readout scheme, where an adaptive threshold should be defined for each column (or row) as discussed earlier. The simulation results show that a simple comparator is required to differentiate between the ONE and the ZERO states.
4.5 Balance Counters Readout

Besides being promising candidates for general purpose memory systems, memristor crossbar arrays are very attractive as well for embedded and ad-hoc systems. However, such systems may not share the accessing properties and hierarchy of the general purpose memory. Hence, the proposed “adaptive threshold” technique is not the best option for memory readout. It is necessary to design a new readout method which can utilize the low power “connected terminals” crossbar and the reusability of the sneak-path parameters over segments. A more complex control and sensing circuit is also required to compensate for the absence of the “locality” property of the memory access.

4.5.1 Readout Technique

Instead of estimating the values of ‘$R_r$’ or ‘$R_c$’ from the “initial bit” readout, they can also be calculated using (4.18). All what we need to know is the number of ONEs per column or row. The number of ones could be used to calculate the sneak-paths resistance components or more simply adjust the sensing threshold based on them. Only the number of ONEs per row or column is needed, based on the sense not location either ‘$n_1$’ or ‘$n_2$’, where we are only interested in knowing one of ‘$R_r$’ and ‘$R_c$’ based on the sensing location as shown in Fig. 4.9. For instance, in the case of current sensing through the node ‘$n_2$’, we need to keep track of the values or ‘$R_c$’ for each column, or more simply the number of ONEs per column. This can be realized by dedicating a counter responsible to keep track of the number of ONEs per each array column. A counter is alerted when a bit at its desired column is swapped. To know if the bit will be swapped or not, a read before write is required, where writing and counting will occur only in case of bit swapping. Bit swapping should happen in 50% of the writes, where a real write is required. This method will increase the writing time of the system by 1.5 time, but on the other side it reduce total power consumption and the reading stages of
the memory. However, for data with regular distribution, only one read stage can be used. Where this apples, the segment counter will only be incremented or decremented in 50 percent of the cases by adding a less significant dummy bit.

4.5.2 Simulation Results

For a given column, the sneak path current $I_c$ acts as DC shift to the sensed current, as shown in Fig. 4.9. This can be translated to linear threshold function in the number of ones in the desired column. Fig. 4.13a shows simulation results for multiple readouts from the same column in case of various NIST RAM images. The simulations show the threshold function in the number of ones per column. Fig. 4.13b shows a normalize version of the readout to demonstrate the margin available between ones and the zeros readout. This gap can be increased by adjusting the data using more descriptive circuit mode. Fig. 4.13c shows the normalized output in the case of considering the cell location in the array effect. However, the complexity added to the system is compensated by the partial improvement added. Therefore, we believe that using a function simple circuit model for the crossbar leads to a better system design. It should be noted that the simulation includes such nonlinearity and more. Our analogy is to use a complex and realistic simulation model for accurate results while using a simple equivalent circuit model for system design. The judge is the simulation results, where the design should be able to distinguish correctly between the binary values of the data.

There are two possible methods to compare the sensed current to the digital counter values. Either to digitize the readout using ADC and do the comparison in the digital domain, or to use DAC to convert the counter output into an analog value and do the comparison in the analog domain. We found that the first method is more power conservative and requires less circuit area compared to the second. It should be noted here that one ADC and comparison circuit is needed per array. Fig. 4.14a shows that a linear threshold still could be defined for the quantized version of the data. The minimum
number of quantization bits required to get a sneak-paths error free output is four, as shown in Fig. 4.14b.

In general, adopting the memory locality property in the “adaptive threshold” technique improve it significantly. It requires very simple control and sensing circuitry, and straightforward writing strategy. However, the “balance counters” readout is more generic for any type of memory systems lacking the hierarchical access or the locality property of the data. This could be of great help in systems with undefined data access patterns, or which lack the locality property. Although, in case of the presence of locality, “adaptive threshold” is a much better option. Both of the introduced techniques
Figure 4.14: (a) Five-bit quantized simulation output for multiple readouts from a 256kb array filled various NIST RAM images, and (b) the readout error versus the number of quantization bit.

provide very fast and power efficient methodology in delay with sneak-paths in gateless array, compared with other methods proposed in the literature. The static power consumed by the crossbar for the proposed methods are less than 0.33mW, for 256kb subarray filled with checkered data pattern. This is less than a sixth of the power consumption of Multiport, Multistage, and grounded techniques, simulated using the same parameters for all of them. Moreover, the proposed methods require a single crossbar access for a readout.

4.6 Readout Power Consumption

Undesirable power consumption due to sneak-paths is not avoidable in high-density gateless arrays. However, it can be significantly reduced by properly selecting the memristive device and crossbar connectivity technique.

4.6.1 Device Nonlinearity

Memristive devices can be characterized into two categories based on their saturation behavior. The first type of devices has a fixed value for its ON and OFF resistance, namely $R_{on}$ and $R_{off}$. The second category has a voltage-dependent saturation resis-
tance, where the ON and OFF resistances depend on the voltage drop across the device terminals. Such type of devices is very attractive for memory applications. Reducing the voltage applied to a nonlinear saturation device by fifty percent can increase its saturation resistance up to two orders of magnitude [104]. The nonlinear saturation behavior significantly mitigates the increased consumption caused by sneak-paths. For floating terminals crossbar, the shortest sneak-path is made of at least three series memristor devices, as depicted in Fig. 3.1b. Therefore, each of the sneak path contributing cells will experience a fraction of the reading voltage. This leads to an efficient decrease in the sneak current and power consumption.

Fig. 4.15a shows the average power consumed by crossbar elements versus the array size for different types of memristive devices. The simulation used the nonlinear memory device presented in [10] and three linear devices with various ON resistances, where all the devices share the same OFF/ON ratio of one thousand. The simulation is carried out for arrays filled with checkered data pattern since it represents the average balanced case of binary data. The figure shows that introducing nonlinearity to the device reduces the power consumption much more than when the saturation ON resistance is increased by ten times. It should be noted that the saturation effect of the power consumption versus the array size is because of the parasitic resistance of crossbar metal lines. As the array size increases, the metal resistances add up in a series and become dominant compared to the equivalent resistance of memristive devices which appear in parallel [11].

### 4.6.2 Array Connectivity

Besides the memory device nonlinearity, array connectivity has a crucial role in the significance of the sneak-paths. The “floating terminals” technique is the simplest configuration and has the lowest power consumption, so we consider it the baseline for comparing other techniques. The first “connected terminals” method that can be considered to handle the sneak-paths effect is grounding the unused rows and columns, by
Figure 4.15: Reading power consumed by a crossbar array filled with checkered data pattern versus the array size for (a) different types of memristive devices, where the ON resistance of linear devices is indicated on the curves, and (b, c) different array connectivity methods. (d) Reading power consumed by a crossbar array filled with checkered data pattern versus the bias voltage of the unused array terminals.

connecting ‘$n_3$’ and ‘$n_4$’ to ground (see Fig. 3.6f). Combining this connectivity with applying $V_{DD}$ at ‘$n_1$’ and virtual ground at ‘$n_2$’ could be considered a straightforward way to obtain a clean readout. However, this technique is very power-hungry as it consumes 14 times more power to access a 256kb array compared to the floating terminals technique, as shown in Fig. 4.15b. In addition, in the “grounded terminals” technique, all the cells in the accessed row and column experience a full $V_{DD}$ voltage drop, thus the device non-linearity will not help in reducing sneak-paths power consumption. In addition, the desired current will sneak out to the new grounds as well as the undesired current.

In contrast to the grounded terminals method, the unused array terminals can be bi-
ased with an optimum voltage to get the maximum benefit out of the device nonlinearity. In [10], it was proposed to bias each of the three sneak-paths components ($R_f$, $R_a$, and $R_c$) at one third of the ‘$V_{DD}$’ by connecting ‘$n_3$’ to ‘$V_{DD}/3$’ and ‘$n_4$’ to ‘$2V_{DD}/3$’. While this might be thought of as the optimal way to utilize the device nonlinearity, it has a major drawback. Typically, ‘$R_a$’ is much smaller than ‘$R_f$’ and ‘$R_c$’ because other than the accessed row and column, it has all the array elements in parallel. Therefore, forcing the voltage drop on this component to be ‘$V_{DD}/3$’ will increase the power consumption dramatically, as shown in Fig. 4.15c.

Another way of biasing is to force ‘$n_3$’ and ‘$n_4$’ to the same voltage ‘$V_B$’. In this case, the small ‘$R_a$’ will be shorted out, and the nonlinearity of the other terminals will be efficiently utilized. Fig. 4.15c shows that by setting ‘$V_B$’ to ‘$V_{DD}/2$’ the power consumption of this method is almost the same as the baseline “floating terminals”. Moreover, the “connected terminals” method has an additional significant advantage which is the defined circuit model. In order to verify that $V_B = V_{DD}/2$ is the optimal voltage, we performed a sweep over ‘$V_B$’ from 0 to ‘$V_{DD}$’ for different array sizes as shown in Fig. 4.15d. Thus, we believe that biasing the unused rows and columns at half the supply voltage is the best selection to go with as the balanced voltage drop over ‘$R_c$’ and ‘$R_e$’ utilizes their nonlinearity the most, in addition to providing circuit model which would be of great help in designing an error free readout scheme.

### 4.7 Summary

In this chapter, we present new structures and techniques that enable correct assess and readout of the memristor gateless crossbar arrays, by eliminating the challenging sneak-paths effect. The presented methods provide the fastest and the most power efficient access to the high-density arrays compared to the work presented in the literature. The presented multi-point structure is considered the first closed-form solution to the sneak-
paths problem, and provides a generic way to access crossbar structures. Further, based on the sneak-paths correlation analysis presented, we present faster and more power efficient readout methodologies. The new techniques enable accessing the crossbar as fast as the DRAM with an optimal power consumption.
Chapter 5:
MOS-Gated Memristor Array

5.1 Introduction

A memristor memory array can be built using gated or gateless memory cells. Gateless cells provide the highest density, where each memory cell is a single memristor fabricated as thin film located at each intersection of each two bars, as shown in Figures 3.2a and 3.2b. This architecture suffers from sneak current, which flows through the memory cells parallel to the desired one and significantly impacts the readout operation [5, 11]. On the other hand, a transistor-gated array mimics the classical DRAM architecture [102, 103], as shown in Figures 3.2c and 3.2d. The introduced gate devices cut the undesired sneak-paths, at the expense of the array density, which is then dominated by the transistor footprint. If smaller transistors are used to reduce the impact on the crossbar density, the array will suffer from leakage current, which prevents the memory from working correctly. In this chapter, we present a new readout technique and its underlying circuitry, which can compensate the effect of leakage current in high density gated memristor arrays.

5.2 Leakage Current Analysis

Leakage current is an undesired current that flows through switched OFF transistors. This current acts as parasitic current for a crossbar array. The leakage current share increases as we scale down the transistors, and it becomes more significant for smaller technology nodes [78]. In general, circuits built using smaller transistors suffer from
higher static power component. For some systems, leakage current can lead to other problems that are even more severe. In our particular case of a gated-memristor array, resistive sensing is used to read the stored data in a particular memory cell, where the binary data are stored in the form of high and low resistance values. While it seems to be a simple process for a single isolated cell, this is unfortunately not the reality. The leakage current of other memory cells can ruin the whole process since the undesired leakage current acts as a parallel parasitic resistance to the desired cell.

During a readout operation, each memory cell falls under one of four types, as shown in Fig. 5.1a. The four cell types are,

- **S**: selected cell
- **R**: half-selected row cells \(\rightarrow (l - 1)\) cells
- **C**: half-selected column cells \(\rightarrow (l - 1)\) cells
- **U**: unselected cells \(\rightarrow (l - 1)^2\) cells

![Figure 5.1](image.png)

Figure 5.1: (a) The distribution of the four different cell types building up the array. (b) Schematic showing the biasing condition of each of the four types.
where 'l' is the array length. The half selected memory cells are the primary source of the leakage current in a memristor array. For the row cells, the half selected gate transistor has a positive $V_{ds}$ while its $V_{gs} = 0$, as shown in Fig. 5.1b. Hence, a leakage current flows from the drain to the ground. On the other hand, the column cells have transistors with zero $V_{ds}$ and $V_{gs} = V_{DD}$, as shown in Fig. 5.1b. Therefore, its leakage current flows from the gate towards the ground. While the leakage current of the two half-selected cell types contributes to power consumption, only the row cells affect the reading operation. For less dense architectures, where two columns or rows are used per each cell, the half selected column cells could also impact the readout process. It should be noted that the two possible series order of the memristor and the transistor are entirely equivalent from a circuit point of view if the correct column and row biasing is used.

To read the desired cell, resistive sensing is done between the desired row and the ground. Hence, leakage current from the desired row acts as a parallel parasitic resistance to the desired cell. However, such current could still ruin our reading. The problem is exacerbated by the parasitic resistance depending on the data stored in the memory, since the leakage current of a single cell depends on the value of the memristor resistance. Hence, simulations using realistic memory data are required to study the significance of the leakage current problem. In all the simulations included in this work, we used the Predictive Technology Models (PTM) for the memory array gates [112]. The same transistors are also used for the memory selectors and the readout circuitry. For memristors, we used the $R_{on}$ and $R_{off}$ values reported by HP Labs for their devices fabricated for memory applications [10, 101]. Synopsys HSPICE is used for simulating the complete memory array. We have built a simulation platform that includes many parasitic effects, such as the crossbar resistance and the driving circuit transistors. Including the different nonidealities led to more accurate results, albeit with a very long simulation time. A Python script has been written to create the SPICE netlists and sweep over the area and different data patterns by calling HSPICE iteratively. Fig. 5.2 shows an HSPICE simu-
loration for the transistor current versus its gate voltage for a single memory cell, where a 16nm PTM transistor is used [112]. The simulations show that the amount of drain current depends on the state of the memristor, whether $R_{on}$ or $R_{off}$. The figure also shows that the desired current (at $V_{gs} = V_{DD}$) is much higher than the leakage one (at $V_{gs} = V_{DD}$). However, the sum of the leakage current from all half-selected row cells will have an aggravated effect.

The equivalent circuit for the array during readout is a resistive ladder network, as shown in Fig. 5.3. The ladder legs represent the desired memory cell and the parallel half-selected row cells. Each cell is represented by the gate resistance ($R_{Si}$) is a series with the memristor resistance ($R_{M}$). Memristors resistance are data dependent, while all the row cells are half selected, except the desired cell gate which is fully selected. However, the resistance of a half-selected switch depends on the series memristor resistance since this series resistance sets the drain-source voltage drop on the transistor. If we neglect the crossbar resistance effect ($R_{cb}$), an approximated expression describing the circuit can be given as,

$$R_{\text{read}} = R_{\text{cell}} \parallel R_{\text{leakage}}$$  \hspace{1cm} (5.1)

where $R_{\text{read}}$ is the equivalent readout resistance, $R_{\text{cell}}$ is the desired cell resistance, and

![HSPICE simulation for the transistor current versus its gate voltage for a single memory cell, with a memristor resistance of $R_{on}$, and $R_{off}$. 16nm PTM transistor used for this simulation [112]. Insight: single memory cell.](image)
$R_{\text{leakage}}$ is equivalent resistance for the rest of the row cells which are all in parallel and given by,

$$R_{\text{leakage}} = \frac{R_{\text{on}} R_{\text{off}} R_{\text{hszo}} R_{\text{hszz}}}{N_{\text{on}} R_{\text{off}} R_{\text{hszo}} + (L - N_{\text{on}} - 1) R_{\text{on}} R_{\text{hszo}}}$$

(5.2)

where $R_{\text{on}}$ and $R_{\text{off}}$ are the ON and OFF resistances of the memristor, $N_{\text{on}}$ is the number bits storing ones ($R_{\text{on}}$) in the ‘R’ cells, and $R_{\text{hszo}}$ and $R_{\text{hszz}}$ are the half-selected switch resistance with series $R_{\text{on}}$ and $R_{\text{off}}$ respectively. The introduced circuit model is very useful at early design stages since it reduces the simulation time and the design process by orders of magnitude. However, it should be noted that all the simulations included in the work used the complete array rather than the equivalent circuit. This enables the inclusion of all the un-modeled or overlooked nonidealities’ effects in the simulations. In addition, this allows capturing the crossbar resistance effect precisely, where the worst-case of $R_{\text{cb}} = 10\Omega$ was used in our simulations [11].

**Gated Array Figure-of-Merit:**

One of the main design aspects for the memristor memory is the trade-off between the array density and its power consumption. The highest density is achieved by the gateless arrays. However, they consume a large amount of energy since current sneaks freely throughout the whole array. On the other hand, introducing a gate element reduces both the power consumption and the array density. This reduction is directly proportional
to the gate area. Larger transistors limit the current to the desired cell only while leaky smaller transistors do not affect the area significantly. Hence, we compare the different technologies based on the figure-of-merit,

\[
FOM = \left( P_r \times A_c \right)^{-1}
\]  

(5.3)

where \( P_r \) is the average power consumed by the array during the readout process, for a memory filled up with checkered data pattern, and \( A_c \) is the memory cell area, which is calculated in accordance with the ITRS reports [6]. It should be noted that an FOM would only make sense in the case of a fully functional memory (i.e., no readout errors). Fig. 5.4 shows the simulation results for the normalized FOM versus different gate length, where the larger is better. It shows that using smaller transistors is the optimal selection for maintaining low power consumption and high array density. The power consumption used in calculating the FOM is extracted from simulating the complete memristor. While the simulations show that the smaller transistors achieve better FOM, it also reveals that they suffer from having relatively high leakage current. In the simulations, we used the Predictive Technology Models (PTM) for the memory array gates [112]. In addition, we include the effect of array switching and the readout circuitry using the same transistors. For memristors, we used \( R_{on} = 1 \text{M} \Omega \) and \( R_{off} = 1 \text{G} \Omega \),

![Figure 5.4: HSPICE simulation for the normalized area-power figure-of-merit (FOM) versus the gate transistor technology node, for various array sizes.](image-url)
in line with the values reported for devices fabricated for memory applications [10, 101]. Synopsys HSPICE and Python scripting are used for simulating the complete memory array. Many parasitic effects are included in the simulations, such as the crossbar resistance, where the worst-case of 10Ω/cell is used [11], as detailed in Section 3.4.

5.3 Impact of Leakage Current on Readout

Noise Margins

The other face of the coin is the effect of the row leakage current on the readout operation. It will act as a data dependent parallel parasitic resistance to the desired cell since the leakage current depends on the value of the memristor resistance of each cell, as shown in Fig. 5.2. This acts as a new source of noise besides the unavoidable thermal noise. Thus, the readout noise margins of the circuit decrease or even vanish. This causes each of the two binary states of a memory cell to be represented by intervals, rather than a single value in the ideal case. The width of each interval defines the severity of problem. In the bounding case scenario, where the memory can be filled with “all zeros” or “all ones”, the current interval width can be approximated as,

$$\Delta_r \approx V_{DD} (l - 1) \left[ \frac{1}{R_{on} R_{hs0}} - \frac{1}{R_{off} R_{hs2}} \right]$$  \hspace{1cm} (5.4)

The readout noise margins decrease as \( \Delta_r \) increases until the regions representing the “zero” value and the “one” value overlap. In such case, it is impossible to use the classic readout technique to get an error free output, even with a theoretical absence of thermal noise.

To study the impact of the leakage current, the readout noise margins and the current intervals are extracted from HSPICE simulations. Fig. 5.5 shows the memristor array simulations at different technology nodes. The simulations show the intervals represent-
Figure 5.5: (a-d) HSPICE simulations for the reading current boundaries for different data patterns. (e) Current interval (Δ) and (f) normalized readout noise margins, versus the memristor array length.

In addition, it shows how readout noise margins vanish and the two intervals overlap for smaller transistor sizes. The simulations use two different data patterns. First, the NIST memory data dump [111], which represents a real workload of a memory, where 1080 data points were simulated for each array size. Furthermore, simulations are made for the theoretical boundary conditions (representing the worst case scenario), where the memory is filled up with “all ones” and “all zeros”, since the maximum leakage current is consumed in the case of “all ones” and the minimum in the case of “all zeros”. The simulations show that the readout noise margins vanish and the two intervals overlap for the smaller transistor size, and even for the larger at 1M array. This adds a new dimension to the problem, where good FOM is desirable only given functional memory and high noise margins. It should be noted that the emerging memory technologies hi-
erarchy is assumed be similar to DRAM (e.g. PCRAM [115]). This hierarchy includes banks, blocks, sub-blocks to increase the memory bandwidth and reduce the parasitic effects, such as the capacitive loading of long crossbar [116, 117]. Other techniques used to reduce the effective size of the array, e.g., folding arrays and open digiline [117], can be borrowed from DRAM architectures too. Such hierarchy limits the maximum size of a continuous array to be less than 1Mb [116]. Therefore, we believe that the proposed simulations cover the region of interest. Finally, it should be noted the memristor devices fabricated for memory applications are engineered to have a writing threshold [10]. For threshold-less devices, the small leakage current per cell drifts the stored values, and hence, zero-flux readout and low rate refresh cycles are needed [16].

5.4 Compensated Readout Technique

The leakage current cannot be avoided in the high-density memristor arrays; however, its effect on the readout can be compensated [9]. A new readout technique is needed in order to restore the noise margins and eliminate the leakage current effect. This readout should dynamically sense the leakage current during each readout operation since the parasitic current is data dependent. As discussed earlier, we know that this resistance is composed of two parallel components: the desired cell resistance and the leakage current resistance. To get rid of the parasitic leakage current, we introduce a new two-phase readout mechanism. In the first phase, only the leakage current is sampled by not activating any of the array columns, hence, the desired cell is half-selected as the rest of the row cells. The current of the first phase reflects the leakage current with a tiny positive shift of the current added by the half selected desired cell. The first phase current is given as,

$$I_{p1} = I_{dh} + I_{leakage}$$  \hspace{1cm} (5.5)

where $I_{dh}$ is the current that passes through a half-selected desired cell. However, adding an extra leakage current of a single cell out of hundreds of cells will not have any effect.
In the second phase, the desired row current is sampled while the desired cell is fully selected. This current is given as,

\[ I_{p2} = I_{df} + I_{leakage} \]  

(5.6)

where \( I_{df} \) is the desired current passing through a fully selected desired cell. The difference between the current of the two phases is equivalent to the desired current with a shift of only a single cell leakage current, rather than that of hundreds of cells. It should be noted that the proposed two-phase reading works with any resistive sensing mechanism while we adopt current sensing in our simulations as a proof of concept. The proposed readout can be realized using a simple circuit as shown in Fig. 5.6. The current is sampled from the desired row using a two-transistor current mirror (\( M_1 \) and \( M_2 \)). The mirrored current charges one of the two capacitors \( C_1 \) or \( C_2 \), based on the reading phase. The two transistors \( M_3 \) and \( M_4 \) are used to activate one of the two charging paths. The voltage difference on the two capacitors, corresponding to the difference in current, is the input of a comparator with hysteresis. The two transistors \( M_5 \) and \( M_6 \) are used to

![Figure 5.6: (a) Schematic for the proposed compensated readout circuit. The used transistor sizes for 16nm technology nodes are; Gate Transistors: 16nm/16nm, \( M_1 \): 32nm/32nm, \( M_2 \): 256nm/64nm, \( M_3, M_4 \): 16nm/16nm, and \( M_5, M_6 \): 64nm/16nm. The used capacitors values are \( C_1 = C_2 = 50 \, \text{fF} \). (b) Trace for the readout circuit operation.](image-url)
reset the circuit for the next reading.

The trace of the circuit operation is shown in Fig. 5.6. Initially, the circuit starts at point ‘a’, where the two capacitors are discharged. During the first phase of the operation capacitor, $C_1$ is charged with the leakage current. The applied voltage on the Schmitt-trigger comparator ($V_d = V_{c1} - V_{c2}$) decreases reaching point ‘b’, thereby resetting the

![Diagram of circuit operation](image)

(a) Reading “zero”

![Diagram of circuit operation](image)

(b) Reading “one”

Figure 5.7: HSPICE for the proposed circuit showing the capacitors voltages and the comparator output voltage, where a full memristor array of size 256K is used for this simulation. The different operation stages are; 1: reset, 2: charge $C_1$, 3: charge $C_2$, and 4: compare.
comparator output. The comparator threshold is selected such that $V_{th} < |V_b|$. The second phase of operation has two scenarios. In the case of desired cell storing “zero” (high resistance), the added desired current pulls back the comparator voltage to point ‘c’, which is inside the hysteresis region, and the output remains at $V_{SS}$. While in the case of desired cell storing “one”, we will have a higher current, which moves the operating point to ‘d’. At this point, the output of the comparator will be set to $V_{DD}$. If the capacitors start to discharge, the input voltage of the comparator will start to decrease approaching zero voltage. However, the output will not be affected because of the comparator hysteresis. In order to get balanced noise margins, the threshold voltage need to be selected as,

$$V_{th} = \frac{V_c + V_d}{2} \approx \frac{I'_{on} \cdot T_p}{2 \cdot C_1}$$

where $I'_{on}$ in the mirrored current in case of only desired current is consumed by ON cell, $T_p$ is the time period of either phase one or two, and $V_c$ and $V_d$ are the voltages at points ‘c’ and ‘d’ respectively. Fig. 5.7 shows the transient simulations for the proposed circuit in case of reading “one” or “zero”, using HSPICE. The simulations are made using a fully gated array of 256k size, which represents a typical DRAM sub-array size.

5.5 Summary

The parasitic effect of leakage current ruins the readout process of a high-density MOS gated memristor array. We have introduced a new readout technique and circuitry capable of eliminating the parasitic effect of leakage current while maintaining a low power-area product figure-of-merit.
Chapter 6:
Memristor-Based Reactance-Less Oscillators (MRLO)

6.1 Introduction

Oscillators are essential components in electronic systems since there is always a need for a repetitive signal with a given frequency and waveform for timing, modulation, and test and measurement applications. In addition, nano-oscillators are the essential elements for oscillation-based bio-inspired computing systems. Oscillators can be classified as sinusoidal or relaxation oscillators. Sinusoidal oscillators are based on positive feedback, where a frequency selective network is used to determine the frequency of oscillation of the sinusoidal output [123]. Relaxation oscillators generate square or triangular waveforms based on astable multivibrators [124]. Both oscillators depend on reactive elements (i.e., capacitors and inductors) to realize the oscillator function. It should be noted that even ring oscillators depend on the delay introduced by the intrinsic and extrinsic capacitance at every node in the chain, where the intrinsic capacitance is the parasitic capacitance of the transistor device while the extrinsic capacitance is an intentionally added one [125].

In this chapter, we present for the first time thorough mathematical derivation describing a complete family of memristor-based reactance-less oscillators (MRLOs) alongside a physical realization of an MRLO. The charging and discharging of a reactive element is replaced by the increase and decrease of the memristor resistance. Instead of having an energy storing reactive element, memristors are used as “resistance-storing”
elements. The inherent delay in the memristor response due to the finite dopant drift mobility is exploited to realize the oscillator function.

MRLO shares the same application spectrum as CMOS oscillators. However, MRLO over-performs CMOS oscillators at lower frequencies. The applications of low-frequency oscillators include biomedical circuits and embedded systems through off-chip components are usually used because large capacitors are required [126, 127]. The implementation utilizing memristors proposed in this chapter eliminates the need for capacitors or inductors allowing a fully integrated implementation in a tiny area. In addition, MRLO is a very attractive candidate for bio-inspired computing using synchronized oscillators [35].

### 6.2 Oscillator Structure

The general architecture of the proposed oscillator is shown in Figure 6.1a. The oscillator is composed of two basic elements \((E_1 \text{ and } E_2)\) forming a voltage divider and a transfer function \((F(V_i))\). \(E_1\) and \(E_2\) can be a memristor or a resistor, where at least one element is a memristor. The voltage on \(E_2\) is given by,

\[
V_i(t) = V_o(t) \frac{R(E_2)}{R(E_1) + R(E_2)}
\]  

(6.1)

![Figure 6.1](image)

**Figure 6.1:** (a) General schematic of the MRLO family. (b-d) Schematic of different sub-types of MRLO family connected in the positive configuration. For the type 'A' two identical memristors connected in opposite polarities are used.
where \( R(E_x) \) is the resistance of element \( E_x \). For the sake of simplicity, \( F(V_i) \) is assumed to have infinite input impedance.

The type of oscillator is determined according to the type of the two basic elements \((E_1 \text{ and } E_2)\). \( E_1 \) and \( E_2 \) can be two memristors, a floating memristor, and a grounded resistor, or a floating resistor and a grounded memristor. For each type, the memristor(s) can be connected such that the magnitude of \( V_i \) increases when \( V_o \) is positive and decreases when \( V_o \) is negative, which we will refer to as the positive configuration. The schematics of the three types of MRLO connected in the positive configuration are shown in Figure 6.1. Alternatively, the memristor(s) can be connected such that the magnitude of \( V_i \) decreases when \( V_o \) is positive and increases when \( V_o \) is negative. This will be referred to as the negative configuration. The analysis and performance of the positive and negative configurations of each type of MRLO are very similar, except that \( F(V_i) \) is different. The transfer functions of \( F(V_i) \) for both positive and negative configurations are shown in Figure 6.2. The threshold voltages \( V_p \) and \( V_n \) should be selected such that \( V_n < 0 < V_p \). A simple implementation of \( F(V_i) \) using two comparators and basic gates is also shown in Figure 6.2. It should be noted that other implementations for \( F(V_i) \) are also possible.

For \( E_1 \) and \( E_2 \) being memristors, their resistances increase or decrease according to the direction of current flowing through them. Using the mathematical model of HP memristor in [74], the memristor resistance as a function of time can be described by,

\[
R_m^2(t) = R_i^2 + 2k' \int_0^t V_m(\tau) d\tau 
\]

where \( R_m \) is the memristor resistance, \( R_i \) is the initial resistance of the memristor, \( V_m \) is the voltage across the memristor, and \( k' \) is the memristor constant given by,

\[
k' = \frac{\alpha \mu_v R_{on} (R_{off} - R_{on})}{d^2} \]

(6.3)
where $\mu_v$ is the dopant drift mobility, $R_{on}$ and $R_{off}$ are the minimum and maximum memristor resistances respectively, and $d$ is the length of the device. The dopant drift mobility is the physical limit that determines the response time of the memristor whereas the polarity indicated by $\alpha = -1$ or $1$ describes how it is connected to the circuit. The suitable polarity configuration, positive or negative, and the oscillation conditions depend on $k'_1$ and $k'_2$, where $k'_x$ is memristor constant of the device ‘x’. Resistors are treated by setting their $k'$ to zero, where $k'_1 = 0$ when a resistor ($R_a$) is used as $E_1$ and $k'_2 = 0$ when a resistor ($R_b$) is used as $E_2$. It is useful to define the relation between the constants of the two devices as,

$$k_r = \frac{k'_1}{k'_2} \quad (6.4)$$

For oscillation to occur, $k_r$ must have any defined value other than unity, given that the proper configuration is used. Hence, no oscillation will occur in the case of two resistors or two memristors with the same memristor constant ($k_r = 1$). In this case, the transfer function input voltage ($V_i$), given in (6.1), will be constant with time; thus, no change will occur at the circuit output. Table 1 shows the suitable configuration for each of

![Diagram](attachment:image.png)

(a) Positive configuration

![Diagram](attachment:image.png)

(b) Negative configuration

Figure 6.2: Transfer function of $F (V_i)$ showing transitions between different operating points, in addition to possible circuit implementations for positive and negative configurations. The circuit is made of two comparators and (a) AND gate (b) NAND gate.
the possible combinations of $k'_x$. These combinations of device polarity and their speed determine whether $V_i$ increases or decrease with positive $V_o$ and vice versa.

### 6.3 Circuit Operation

In this section, we present the generic analysis of the oscillator circuit, where two memristors connected in the same polarity is assumed. The two memristors are connected such that their resistances increase with positive $V_o$ and decrease for the negative one as shown by the top-most M-M circuit in Table 6.1. For proper operation, $E_2$ must be faster than $E_1$ such that $k_r < 1$. The operation of the oscillator can be traced as follows, assuming that we start at ‘a’:

- **a → b**: At ‘a’, $V_o$ is positive and is equal to $V_{oh}$. Both $R_a$ and $R_b$ will increase, but $R_b$ will increase in a faster speed. Therefore, $V_i$ will increase until the operating point reaches ‘b’.

- **b → c**: At ‘b’, the value of $V_i$ will just cross $V_p$, thus $V_o$ will switch to $V_{ot}$, and the operating point will jump to ‘c’.

- **c → d**: At ‘c’, the resistances of the two memristors will decrease as $V_o$ is negative ($V_o = V_{ot}$), but $R_b$ will decrease faster. Thus, $|V_i|$ will decrease and the operating point will move to ‘d’.

- **d → a**: At ‘d’, when $V_i$ just crosses $V_n$, $V_o$ will switch from $V_{ot}$ to $V_{oh}$. Consequently, the operating point will instantly move to ‘a’. And the oscillation will continue.

### 6.3.1 Oscillation Conditions

From (6.2), the relation between $R_a$ and $R_b$ and the voltage across them can be written as,

\[ R_a dR_a = k'_a (V_o - V_i) dt \]  \hspace{1cm} (6.5)

\[ R_b dR_b = k'_b V_i dt \]  \hspace{1cm} (6.6)
Table 6.1: All possible combinations of memristor constants and suitable configuration (positive or negative). M abbreviates memristor and R abbreviates resistor.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Configuration</th>
<th>Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-R</td>
<td>$k_r = \infty$</td>
<td>$k'_i &lt; 0$</td>
<td>+ve</td>
<td><img src="image1" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k'_i &gt; 0$</td>
<td>−ve</td>
<td><img src="image2" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td>$1 &lt; k_r &lt; \infty$</td>
<td>$k'_i &lt; 0$</td>
<td>+ve</td>
<td><img src="image3" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k'_i &gt; 0$</td>
<td>−ve</td>
<td><img src="image4" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td>$k_r = 1$</td>
<td></td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>M-M</td>
<td>$0 &lt; k_r &lt; 1$</td>
<td>$k'_i &gt; 0$</td>
<td>+ve</td>
<td><img src="image5" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k'_i &lt; 0$</td>
<td>−ve</td>
<td><img src="image6" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td>$-\infty &lt; k_r &lt; 0$</td>
<td>$k'_i &lt; 0$</td>
<td>+ve</td>
<td><img src="image7" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k'_i &gt; 0$</td>
<td>−ve</td>
<td><img src="image8" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td>R-M</td>
<td>$k_r = 0$</td>
<td></td>
<td>+ve</td>
<td><img src="image9" alt="Circuit Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k'_2 &lt; 0$</td>
<td>−ve</td>
<td><img src="image10" alt="Circuit Diagram" /></td>
</tr>
</tbody>
</table>
Given that the same current flows through the two memristors, the relation between the two memristor resistances is given by,

\[ \frac{1}{k_a'} dR_a = \frac{1}{k_b'} dR_b \]  

(6.7)

By integrating both sides, the relation between \( R_a \) and \( R_b \) and their initial values is given by,

\[ R_a - R_{ai} = k_r (R_b - R_{bi}) \]  

(6.8)

where \( R_{ai} \) and \( R_{bi} \) are the initial resistances of \( R_a \) and \( R_b \) respectively. From (6.1) and by using (6.8), the transition resistances at \( V_i = V_p \) are given by,

\[
R_{ap} = \frac{R_{bi}k_r - R_{ai}}{\left(\frac{V_o}{V_p} \right) k_r - 1}
\]  

(6.9)

\[
R_{bp} = \frac{R_{ai} - R_{bi}k_r}{\left(\frac{V_o}{V_p} \right) - k_r}
\]  

(6.10)

where \( R_{ap} \) and \( R_{bp} \) are the values of \( R_a \) and \( R_b \) at the transition point \( (V_i = V_p) \) respectively. Similarly, the transition resistances at \( V_i = V_n \) are given by,

\[
R_{an} = \frac{R_{bi}k_r - R_{ai}}{\left(\frac{V_o}{V_n} \right) k_r - 1}
\]  

(6.11)

\[
R_{bn} = \frac{R_{ai} - R_{bi}k_r}{\left(\frac{V_o}{V_n} \right) - k_r}
\]  

(6.12)

where \( R_{an} \) and \( R_{bn} \) are the values of \( R_a \) and \( R_b \) at the transition point \( (V_i = V_n) \) respectively. Besides the necessary conditions for \( k_r \) given in Table 6.1, the memristor resistances must not reach saturation in order to sustain oscillation,

\[ R_{on,a} < R_a < R_{off,a} \]  

(6.13)

\[ R_{on,b} < R_b < R_{off,b} \]  

(6.14)
where $R_{on,a}$ and $R_{off,a}$ are the ON and the OFF values of $R_a$, and $R_{on,b}$ and $R_{off,b}$ are the ON and the OFF values of $R_b$. By substituting (6.13) and (6.14) in (6.9), (6.10), (6.11), and (6.12), the oscillation condition can be given as,

$$
R_{bi}k_r - R_{ai} < \min \begin{cases} R_{off,a} \left[ \left( \frac{V_p}{V_{oh} - V_p} \right) k_r - 1 \right] \\ R_{off,b} \left[ k_r - \left( \frac{V_{oh} - V_p}{V_p} \right) \right] \end{cases} 
$$

(6.15)

$$
R_{bi}k_r - R_{ai} > \max \begin{cases} R_{on,a} \left[ \left( \frac{V_a}{V_{ol} - V_n} \right) k_r - 1 \right] \\ R_{on,b} \left[ k_r - \left( \frac{V_{ai} - V_n}{V_n} \right) \right] \end{cases} 
$$

(6.16)

### 6.3.2 Oscillation Frequency

The oscillation frequency can be calculated by rewriting (6.5) and (6.6) as,

$$
dt = \frac{1}{V_ohk'_a} R_a dR_a + \frac{1}{V_ohk'_b} R_b dR_b
$$

(6.17)

By integrating (6.17) from $V_i = V_n$ to $V_i = V_p$, the time of the positive half cycle is given by,

$$
T_H = \frac{1}{2V_ohk'_a} (R_{ap} - R_{an}^2) + \frac{1}{2V_ohk'_b} (R_{bp}^2 - R_{bn}^2)
$$

(6.18)

Similarly, the negative half cycle can be calculated by integrating (6.17) from $V_i = V_p$ to $V_i = V_n$,

$$
T_L = \frac{1}{2V_ohk'_a} (R_{an}^2 - R_{ap}^2) + \frac{1}{2V_ohk'_b} (R_{bn}^2 - R_{bp}^2)
$$

(6.19)

Therefore, the oscillation frequency is given by,

$$
f = \frac{2V_ohV_{ol} (k'_a V_n + k'_b V_n - k'_b V_{ol})^2 (k'_d V_p + k'_b V_p - k'_b V_{oh})^2}{(V_{oh} - V_{ol}) (V_n V_{oh} - V_{ol} V_p) (k'_b R_{ai} - k'_a R_{bi})^2 [(V_n V_{oh} + V_p V_{ol}) (k'_a + k'_b) - 2k'_b V_{oh} V_{ol}]}
$$

(6.20)
This expression is the general case for all the possible values of $k_r$ except $k_r = 1$, since no oscillation occurs at the unity value. Equation (6.20) can be simplified for $V_{ol} = -V_{oh}$,

$$f = \frac{(k'_a V_n + k'_b V_n + k'_b V_{oh})^2 (k'_a V_p + k'_b V_p - k'_b V_{oh})^2}{V_{oh} (V_p + V_n) (k'_a R_{bi} - k'_b R_{ai})^2 [(V_n - V_p) (k'_a + k'_b) + 2k'_b V_{oh}]}$$

(6.21)

The minimum and the maximum frequencies of the system can be calculated by substituting (6.15) and (6.16) in (6.20). The derived frequency equations are general and hold for any proposed MRLOs introduced in the next sections.

### 6.3.3 Validation

In order to verify memristor-based circuits, researchers utilize SPICE, Verilog-A, and/or Matlab models [18,76,86,128], or emulate the memristor model using active circuitry [54, 129]. The proposed circuit was verified using Cadence Virtuoso 6 Spectre transient circuit simulations employing the memristor model given in [74]. The parameters $R_{on}$, $R_{off}$, $d$, and $\mu_v$ were selected to be $100\Omega$, $38k\Omega$, $10nm$, and $10^{-10}cm^2s^{-1}V^{-1}$ respectively, which are the same values reported by HP Labs in [4]. The circuit was simulated for $V_{oh} = 1V$, $V_{ol} = -1V$, $V_p = 0.75V$, and $V_n = -0.5V$. Figure 6.3 compares

![Figure 6.3: Tuning curve for the oscillation frequency using (6.20) and circuit simulation on Cadence Spectre for different values of $k_r$, where $R_{on} = 100\Omega$, $R_{off} = 38k\Omega$, $d = 10nm$, $\mu_v = 10^{-10}cm^2s^{-1}V^{-1}$, $R_{ai} = 4k\Omega$, $V_{oh} = 1V$, $V_{ol} = -1V$, $V_p = 0.75V$, $V_n = -0.5V$, $k'_1 = -3.79 \times 10^8$ and $k'_2 = 3.79 \times 10^8$.](image)
frequency of oscillation from transient simulation with the derived expression as \( R_{bi} \) is swept from \( 1 \, k\Omega \) to \( 12 \, k\Omega \). For \( (k_r < 1) \), the circuit oscillates faster as the ratio \( k_r \) decreases, i.e., for larger difference in the speeds of the two memristors.

In the following subsections, three particular types of the MRLO are given. These oscillators are easier to tune and have simpler governing equations. In Section 6.7, a comparison between these types is introduced.

### 6.4 MRLO Type ‘A’: Two Opposite Memristors

MRLO type ‘A’ is a special case where both \( E_1 \) and \( E_2 \) are two identical memristors of opposite polarities. Thus, there will be two oppositely varying resistances.

The analysis will be done for the positive configuration shown in Figure 6.1, noting that the analysis of the negative configuration is similar. The memristors are connected in a polarity such that \( R_a \) decreases for positive \( V_o \) and increases for negative \( V_o \) and the opposite applies for \( R_b \). It should be noted that \( V_o \) and \( V_i \) always have the same polarity and the magnitude of \( V_o \) is always higher than the magnitude of \( V_i \).

#### 6.4.1 Mathematical Analysis

From (6.2), the relation between \( R_a \) and the voltage across it can be written as,

\[
R_a dR_a = k' (V_i - V_o) \, dt
\]  

(6.22)

Thus, the current flowing through \( R_a \) can be written as,

\[
i_a = -\frac{1}{k'} \frac{dR_a}{dt}
\]  

(6.23)
Similarly, the relation between $R_b$ and the voltage across it is given by,

$$R_b dR_b = k' V_i dt$$  \hspace{1cm} (6.24)$$

and the current flowing through $R_b$ is,

$$i_b = \frac{1}{k'} \frac{dR_b}{dt}$$  \hspace{1cm} (6.25)$$

We assume that $F(V_i)$ has infinite input resistance. Thus, from (6.23) and (6.25),

$$\frac{dR_a}{dt} = - \frac{dR_b}{dt}$$  \hspace{1cm} (6.26)$$

By integrating both sides,

$$\int_{R_{ai}}^{R_a} dR_a = - \int_{R_{bi}}^{R_b} dR_b$$  \hspace{1cm} (6.27)$$

where $R_{ai}$ and $R_{bi}$ are the initial values of $R_a$ and $R_b$ respectively. Thus, the relation between $R_a$ and $R_b$ and their initial values is given by,

$$R_a + R_b = R_{ai} + R_{bi}$$  \hspace{1cm} (6.28)$$

To derive the oscillation condition we need to write the expressions of $R_a$ and $R_b$ at the transition points, i.e., $V_i = V_p$ and $V_i = V_n$. From (6.1), the transition resistances are given by,

$$R_{ap} = \frac{V_{oh} - V_p}{V_{oh}} (R_{ai} + R_{bi})$$  \hspace{1cm} (6.29)$$

$$R_{bp} = \frac{V_p}{V_{oh}} (R_{ai} + R_{bi})$$  \hspace{1cm} (6.30)$$

$$R_{an} = \frac{V_{ol} - V_n}{V_{ol}} (R_{ai} + R_{bi}), \text{ and}$$  \hspace{1cm} (6.31)$$

$$R_{bn} = \frac{V_n}{V_{ol}} (R_{ai} + R_{bi})$$  \hspace{1cm} (6.32)$$
where \( R_{ap}, R_{bp}, R_{an}, \) and \( R_{bn} \) are the values of \( R_a \) and \( R_b \) at the transition points \( V_i = V_p \) and \( V_i = V_n \) respectively. The transition resistances depend on the sum of the initial memristor resistances; thus the oscillation period will also depend on this sum. The memristor resistance oscillates in the time domain between the transition resistances, and as the rails of oscillation change, the oscillation period will also change.

Based on the circuit tracing given in the previous section, the oscillation will occur if \( V_p \) and \( V_n \) are selected such that,

\[
V_p > V_n \frac{V_{oh}}{V_{ol}} \tag{6.33}
\]

The transition resistances should fall within the minimum and maximum memristor resistances,

\[
R_{on} < \{R_{an}, R_{ap}, R_{bn}, R_{bp}\} < R_{off} \tag{6.34}
\]

By substituting with the formulas of the transition resistances given in (6.29), (6.30), (6.31), and (6.32) the oscillation condition can be rewritten as,

\[
(R_{ai} + R_{bi}) > \max \left( \frac{V_{ol}}{V_n}, \frac{V_{oh}}{V_{oh} - V_p} \right) R_{on} \tag{6.35}
\]

\[
(R_{ai} + R_{bi}) < \min \left( \frac{V_{oh}}{V_p}, \frac{V_{ol}}{V_{ol} - V_n} \right) R_{off} \tag{6.36}
\]

To derive an expression for the oscillation frequency, from (6.22) and (6.24), we can write the time differential as,

\[
dt = \frac{1}{k'V_o} (R_0 dR_b - R_a dR_a) \tag{6.37}
\]

Oscillation frequency can be derived in the same technique shown in the previous section. However, the same equation can be given by substituting \( k'_2 = -k'_1 = k' \) in (6.20),

\[
f = \frac{k'V_{ol}^2V_{oh}^2}{(V_{oh} - V_{ol}) (V_n V_{oh} - V_p V_{ol}) (R_{ai} + R_{bi})^2} \tag{6.38}
\]
Figure 6.4: Spectre transient simulation results for $I(R_m)$, $R_m$, $V_i$, and $V_o$ respectively for $R_{ai} = R_{bi} = 3 \, k\Omega$ and other parameters are the same as Figure 6.3.

At $V_{ol} = -V_{oh}$, the expression of the oscillation frequency can be further simplified to,

$$f = \frac{k'V_{oh}^2}{2 (V_p + V_n) (R_{ai} + R_{bi})^2}. \quad (6.39)$$

The valid range for the oscillation frequency can be derived by substituting the oscillation condition given in (6.35) and (6.36) into (6.39). Substituting, the minimum and
maximum oscillation frequencies are given by,

\[
    f_{\text{min}} = \frac{k' \max \left(V_p^2, (V_{oh} + V_n)^2\right)}{2R_{\text{eff}}^2 (V_p + V_n)}
\]

\[
    f_{\text{max}} = \frac{k' \min \left(V_n^2, (V_{oh} - V_p)^2\right)}{2R_{\text{on}}^2 (V_p + V_n)}
\]

### 6.4.2 Validation

The two memristor resistances change in a linear fashion in opposite directions and their sum is always constant. The transition resistances depend on the sum of the initial resistances and define the frequency of oscillation. By substituting the circuit parameters into (6.29), (6.30), (6.31), (6.32), and (6.39): \( R_{ap} = 1.5 \, k\Omega \), \( R_{bp} = 4.5 \, k\Omega \), \( R_{an} = 3 \, k\Omega \), \( R_{bn} = 3 \, k\Omega \), and \( f = 21.06 \, Hz \), which shows an excellent match with the simulation results (see Figure 6.4). For further verification of the mathematical analysis presented, the oscillation frequency was tuned by sweeping \( R_{bi} \) from 1 k\( \Omega \) to 12 k\( \Omega \) (see Figure 6.5).

Figure 6.4 shows transient simulation results for \( (k_r = -1) \) description for the model given in [74] running on Cadence Virtuoso 6 Spectre, where \( R_{ai} = 4 \, k\Omega \) and \( R_{bi} = 3 \, k\Omega \). A comparison between simulation results and the derived expression is shown in Figure 6.5. The comparison shows an excellent match between the simulation results and the derived expression, with a maximum error less than 0.16%.

For the negative configuration, the analysis is similar. But it should be noted that for this case, the oscillation will occur when \( V_p \) and \( V_n \) are selected such that

\[
    V_p < V_n \frac{V_{oh}}{V_{ol}}
\]

Thus, an additional negative sign will appear in the expressions of oscillation frequency and frequency range, such that the overall expression is positive.
Figure 6.5: Tuning curve for the oscillation frequency of type ‘A’ using (6.39) and circuit simulation on Cadence Spectre for $R_{ai} = 5\, k\Omega$. Other parameters are the same as in Figure 6.3.

### 6.5 MRLO Type ‘B’: Floating Memristor

For MRLO type ‘B’, $E_1$ and $E_2$ are a memristor and a resistor respectively. The memristor is connected between $V_o$ and $V_i$ nodes, where both have varying voltage. For the positive configuration shown in Figure 6.1c, the memristor is connected such that its resistance decreases when $V_o$ is positive and increases when $V_o$ is negative.

#### 6.5.1 Mathematical Analysis

From (6.1), the transition resistances at $V_i = V_p$ and $V_i = V_n$ are given by

$$R_{mp} = R_a \frac{V_{oh} - V_p}{V_p} \quad (6.43)$$

$$R_{mn} = R_a \frac{V_{ol} - V_n}{V_n} \quad (6.44)$$

The transition resistances do not depend on the memristor initial resistance. From the previous circuit tracing, the oscillation will occur if $V_p$ and $V_n$ are selected such that

$$V_p > V_n \frac{V_{oh}}{V_{ol}} \quad (6.45)$$
The transition resistances have to satisfy

\[ R_{on} < \{ R_{mn}, R_{mp} \} < R_{off} \]  \hspace{1cm} (6.46)

By substituting with (6.43) and (6.44) into (6.46), the oscillation condition can be rewritten as

\[ R_{on} \left( \frac{V_p}{V_{oh} - V_p} \right) < R_a < R_{off} \left( \frac{V_n}{V_{ol} - V_n} \right) \]  \hspace{1cm} (6.47)

The frequency can be calculated by taking the limit \((k'_2 = k'_6 \to 0)\) of (6.20). This yields

\[ f = \frac{2k'V_n^2V_p^2V_{ol}V_{oh}}{R_a^2(V_n^2V_{oh}^2 - V_p^2V_{ol}^2)(V_{oh} - V_{ol})} \]  \hspace{1cm} (6.48)

It should be noted that (6.20) is the general case for all possible values of \(k_r\) except \(k_r = 1\). For the case \(V_{ol} = -V_{oh}\), the expression of the frequency of oscillation is simplified to be

\[ f = \frac{k'V_n^2V_p^2}{R_aV_{oh}(V_p^2 - V_n^2)} \]  \hspace{1cm} (6.49)

The frequency of oscillation depends on the memristor constant \(k'\) and the values of \(V_{oh}\), \(V_{ol}\), \(V_p\), \(V_n\), and \(R_a\). The most direct way to tune the frequency of oscillation is varying the resistance \(R_a\). By substituting the simplified frequency expression into the oscillation condition, the minimum and maximum oscillation frequencies are given by

\[ f_{\text{min}} = \frac{k'V_p^2(V_{oh} + V_n)^2}{R_{off}^2V_{oh}(V_p^2 - V_n^2)} \]  \hspace{1cm} (6.50a)

\[ f_{\text{max}} = \frac{k'V_n^2(V_{oh} - V_p)^2}{R_{on}^2V_{oh}(V_p^2 - V_n^2)} \]  \hspace{1cm} (6.50b)

6.5.2 Validation

Using \(R_a = 3 \, \text{k}\Omega\) and the same parameters used in validation of type ‘A’, the derived expressions were compared to transient simulation results. Figure 6.6 shows transient
simulation results. The memristor resistance changes non-linearly and oscillates between two fixed rails which do not depend on the memristor initial state. By substituting parameter values in (6.43), (6.44) and (6.49), the memristor resistance oscillates between $R_{mp} = 1 \, k\Omega$ and $R_{mn} = 3 \, k\Omega$ and $f_o = 18.95 \, Hz$, which matches the simulation results. The frequency of oscillation tuning curve using $R_a$ as a parameter is plotted against simulation results in Figure 6.7, showing excellent match between simulation and analysis.
6.6 MRLO Type ‘C’: Single Grounded Memristor

In MRLO type ‘C’, $E_1$ is a resistor and $E_2$ is a memristor. The positive configuration of this type, shown in Figure 6.1d, presented by the authors in [36]. The same analysis can be applied to the family of circuits presented in this work but was omitted for simplicity.

6.6.1 Mathematical Analysis

Using a similar analysis to type ‘B’, it can be that the transition resistances at $V_i = V_p$ and $V_i = V_n$ are,

$$R_{mp} = R_a \frac{V_p}{V_{oh} - V_p} \quad (6.51)$$

$$R_{mn} = R_a \frac{V_n}{V_{ol} - V_n} \quad (6.52)$$

The oscillation condition in terms of design parameters is given by,

$$R_{on} \left( \frac{V_{ol} - V_n}{V_n} \right) < R_a < R_{off} \left( \frac{V_{oh} - V_p}{V_p} \right) \quad (6.53)$$
The frequency can be calculated by taking the limit ($k'_1 = k'_a \to 0$) of (6.20). This yields:

$$f = \frac{-2k'V_{oh}V_{oh} (V_{oh} - V_p)^2 (V_n - V_{oh})^2}{R_a^2 (V_{oh} - V_{oh}) (V_p V_{oh} - V_n V_{oh}) (2V_{oh} V_{oh} - V_{oh} V_{oh} V_n)}$$  (6.54)

It should be noted that (6.20) is the general case for all possible values of $k_r$ except $k_r = 1$. For $(V_{oh} = -V_{oh})$, the oscillation frequency can be further simplified to,

$$f = \frac{k' (V_{oh} - V_p)^2 (V_{oh} + V_n)^2}{R_a^2 V_{oh} (V_p + V_n) (2V_{oh} - V_p + V_n)}$$  (6.55)

The valid range for the oscillation frequency can be derived by substituting the oscillation condition given in (6.53) into (6.55). The minimum and maximum oscillation frequencies are,

$$f_{\text{min}} = \frac{k'V_p^2 (V_{oh} + V_n)^2}{R_{\text{off}}^2 V_{oh} (V_p + V_n) (2V_{oh} - V_p + V_n)}$$  (6.56)

$$f_{\text{max}} = \frac{k'V_n^2 (V_{oh} - V_p)^2}{R_{\text{oh}}^2 V_{oh} (V_p + V_n) (2V_{oh} - V_p + V_n)}$$  (6.57)

### 6.6.2 Validation

Using same parameters as type ‘B’, the oscillator was simulated using SPICE and Verilog-A models, both giving similar results. For further verification of the derived formulas, the oscillation frequency was tuned by sweeping $R_a$ from 1 $k\Omega$ to 12 $k\Omega$. Both simulation results and the derived expression are plotted in Figure 6.8.

### 6.7 Comparison and Discussion

Table 6.2 shows a comparison between the three types of MRLO. For each type, two configurations that yield identical performance are possible, except that a different transfer function ($F(V_i)$) is required. The transfer function that has simpler implementation
will determine the configuration of choice. For type ‘B’ and type ‘C’, the transition resistances do not depend on the initial state of the memristor, thus the oscillation condition and the oscillation frequency are independent of the memristor initial resistance. However, for the type ‘A’ the transition resistances, and consequently the oscillation frequency, depend on the sum of memristor initial resistances. Dependence on initial re-

Table 6.2: Comparison between the different types of MRLOs family.

<table>
<thead>
<tr>
<th>First element ($E_1$)</th>
<th>Type A</th>
<th>Type B</th>
<th>Type C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memristor</td>
<td>Memristor</td>
<td>Resistor</td>
<td></td>
</tr>
<tr>
<td>Second element ($E_2$)</td>
<td>Memristor</td>
<td>Resistor</td>
<td>Memristor</td>
</tr>
<tr>
<td>Depends on IC</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Constant current</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Oscillation freq.</td>
<td>$\frac{k'V_{oh}^2}{2(V_p+V_n)(R_{on}+R_{bi})^2}$</td>
<td>$\frac{k'V_{n}^2V_p}{R_{n}^2V_{oh}(V_p^2-V_{n}^2)}$</td>
<td>$\frac{k'(V_{oh}-V_{p})^4(V_{oh}+V_{n})^2}{R_{on}^2V_{oh}(V_p+V_n)(2V_{oh}-V_p+V_{n})}$</td>
</tr>
<tr>
<td>Minimum freq.</td>
<td>$\frac{k'}{2R_{on}^2(V_p+V_n)}$</td>
<td>$\frac{k'V_{n}^2(V_{oh}+V_{n})^2}{R_{n}^2V_{oh}(V_p+V_n)}$</td>
<td>$\frac{k'V_{oh}^2(V_{oh}+V_{n})^2}{R_{on}^2V_{oh}(V_p+V_n)(2V_{oh}-V_p+V_{n})}$</td>
</tr>
<tr>
<td>Maximum freq.</td>
<td>$\frac{k'min(V_{n}^2(V_{oh}-V_{p})^2)}{2R_{on}^2(V_p+V_n)}$</td>
<td>$\frac{k'V_{n}^2(V_{oh}-V_{p})^2}{R_{n}^2V_{oh}(V_p+V_n)}$</td>
<td>$\frac{k'V_{oh}^2(V_{oh}-V_{p})^2}{R_{on}^2V_{oh}(V_p+V_n)(2V_{oh}-V_p+V_{n})}$</td>
</tr>
<tr>
<td>Oscillation cond. 1</td>
<td>$R_{on,i} &lt; R_{m,i} &lt; R_{off,i}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oscillation cond. 2*</td>
<td>$(-1)^l V_p V_{ol} &gt; (-1)^l V_n V_{oh}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* $l$ equals ‘1’ for positive transfer function and ‘-1’ for negative transfer function.
sistances may be as a disadvantage. However, this can also be considered as an additional degree of freedom to control the frequency of oscillation. An external circuit can be used to adjust the initial resistances of the memristors and thus tune the oscillation frequency. One important note for type ‘A’ is that the sum of the two memristor resistances is always constant. Thus, the magnitude of the current flowing through them is constant as well while its polarity depends on the output voltage. This can be an advantage while designing $F(V_i)$ on the circuit level.

In addition to tuning the frequency of oscillation, the oscillation range can be also tuned. Figure 6.9 shows a comparison of the minimum and maximum oscillation frequencies for the three types of MRLO using $V_p$ as a tuning parameter. For the values of $V_p$ and $V_n$ used in this comparison, type ‘A’ provides the highest maximum frequency. This is not always the case, as the speed of the oscillator depends on the rate of change of the memristor resistance, which in turn depends on the voltage drop across the memristor. The voltage drop across the memristor depends on the choice of $V_p$ and $V_n$ and

![Figure 6.9](image)

Figure 6.9: A Comparison of the maximum and minimum frequencies of the three types of MRLO family, where $R_{ai} = R_{bi} = 3k\Omega$ and other device and circuit parameters are the same as in Figure 6.3. For type ‘B’ and type ‘C’, $R_a$ is selected to be equal to $0.5(R_{ai} + R_{bi})$ of type ‘A’.
the circuit configuration. To further illustrate the effect of $V_p$ and $V_n$, a plot showing the type of MRLO giving the highest oscillation frequency in the $(V_p, |V_n|)$ plane is shown in Figure 6.10, where positive configuration is used for the area of $V_p > |V_n|$ while the negative one is used elsewhere. The three presented types of MRLO can be seen as a voltage-controlled oscillator (VCO). The control voltage can be to one of the voltage parameters that control the oscillation frequency, e.g., $V_p$ and $V_n$. Alternatively, the resistance $R_a$ can be as a voltage controlled resistance (VCR) for types ‘B’ and ‘C’.

The other main parameter controlling the oscillation frequency is the speed of the memristor device itself, which is modeled by the memristor constant $k'$. Figure 6.11 shows that the oscillation frequency increases linearly as the memristor constant increases. Very high oscillation frequencies can be achieved using recently introduced fast memristors [100, 130, 131]. However, slower memristors have their useful domain of applications.
Figure 6.11: Maximum frequency of the three types of MRLO versus the memristor constant \((k')\) at \(V_p = 0.65\). Where \(R_{ai} = R_{bi} = 3\, k\Omega\) and other device and circuit parameters are the same as in Figure 6.3. For type ‘B’ and type ‘C’, \(R_a\) is selected to be equal to \(0.5 \times (R_{ai} + R_{bi})\) of type ‘A’.

### 6.8 Experimental Results

MRLO Type ‘C’ circuit was built using off-the-shelf discrete electronic components mounted on a PCB and one of our fabricated memristor devices, as shown in Figure 6.12. The window comparator is built using two comparators and an AND gate. We used voltage regulators to supply \(V_{oh} = 2.5\, V\) and \(V_{ol} = -2.5\, V\). \(V_p\) and \(V_n\) were set using voltage dividers since they are not driving any loads. The same circuit can be used to build any of the MRLO family members.

Figure 6.12: Snapshot of the oscillator PCB. (a) Voltage regulators and voltage dividers. (b) Window function circuit made of two comparators and an AND gate. (c) Resistor \(R_b = 560\, k\Omega\). (d) Memristor die connected to the PCB.
Figure 6.13a shows the measured response of the window transfer function to a ramp waveform. The transfer function is with a frequency of $200\, kHz$, which is in the same order of the oscillation frequency we got from the circuit. The measured square wave output of the oscillator is in Figure 6.13. The oscilloscope snapshot shows that the circuit frequency of oscillation $\approx 151\, kHz$. This frequency is at $V_p = V_{REF} + 0.9V$ and $V_n = V_{REF} - 0.6V$.

![Figure 6.13: Measurement results of (a) the window function circuit response to a ramp input and (b) square wave output of the implemented MRLO circuit.](image)

### 6.9 Summary

A family of MRLO oscillators that can be implemented without capacitors or inductors was presented. The proposed oscillators use one or two memristors instead of reactive elements. Types ‘A’, ‘B’, and ‘C’ of the MRLO family were analyzed and simulated. Comparison between derived expressions and simulation results shows an excellent match. Type ‘A’ has a constant current magnitude advantage, but it depends on the initial state of the memristors. Types ‘B’ and ‘C’ do not depend on the initial conditions, but a resistor does. The three types can be used as a VCO and can offer a substantial area advantage over traditional oscillators in the low-frequency range. Physical realization and measurement results of a type ‘C’ MRLO were also introduced.
Chapter 7:
Conclusion & Future Work

7.1 Conclusion

In this work, we joined the research community’s efforts to enable the memristor technology, in order to advance the computing and electronic systems beyond the CMOS era. On the system level, we introduced new structures and readout techniques for the high-density memristive crossbar. The new strategies enable much faster and power efficient access to high-density memristive crossbar as compared to other work presented in the literature. This was not possible without our new highly descriptive memristive arrays simulation platform, which is capable of simulating realistic size arrays filled with real memory data while accounting for various crossbar nonidealities. On the circuit level, we proposed a novel memristor reactance-less oscillators family for embedded systems and bio-inspired computing. The proposed concept was proved through theory, simulations, and experimental results. On the device level, we fabricated memristor devices and arrays at our local clean room for circuit implementation purposes. The fabricated devices enriched our understanding of the memristive behavior. In addition, we introduced analytical study and mathematical modeling for memristors. The presented theoretical analysis is essential for the first stages of memristor-based circuit design. All in all, we believe that the presented work should help in paving the road for better memristor-based computing and electrical systems.

For the future work effort, we will be working on associative processing, using content addressable memories as presented in the next section.
7.2 Future Work: Associative Processing using CAMs

Computer memory and storage sizes are growing rapidly, and occupying larger silicon share. In addition, in memory computing provides a promising alternative to the conventional Von Neumann architecture. This can be realized using associative processing (AP) on content addressable memories (CAMs). The in-memory AP concept was initially presented in the seventies and eighties era of the last century [132, 133], but its area and power consumption were not competitive to compete with the mainstream technologies. The recent advances in resistive memory technologies offers much higher density and power efficient CAM architecture [134]. This shows a high potential for AP of memristor-based memories, which is the motivation behind our future work plans.

In general, accessing the data stored in the memory could be achieved using one of two primary strategies, as shown in Fig. 7.1. The first, and the most common, way is by providing an address to the memory and getting the data stored at this address, a process named Random Access Memory (RAM). The other way is to search if a particular data word is stored in the memory and get in return its corresponding address(es), which is called Content Addressable Memory (CAM).

7.2.1 CAM Operation

Our goal is to use memory for data storage and processing altogether. This sets the main guidelines for our circuit design. While being able to search the memory in a parallel

![Figure 7.1: Memory accessing strategy for (a) Random Access Memory (RAM) vs. (b) Content Addressable Memory (CAM).](image)
fashion is the sole requirement for CAMs, it is not the case for associative computing. As highlighted by the previous examples, there are other mandatory properties to enable on-memory data processing. These features are masked search, and parallel column write. Such requirement can be achieved using various CAM cells, as presented in Table 7.1. The best candidate for a CAM primary cell is the CAM cell presented in [135], which is made of two memristors and two transistors. This cell is a REDOX compatible cell that complies with associative computing requirements. It should be noted here that the cell presented in [136] would not work with the high-density bipolar REDOX devices.

The essential requirement for a CAM array is to sort out the location of matches with a search word. This is typically achieved by pre-charging all the rows of the array, and then a search word is applied to the columns. During the evaluation phase, only rows carrying matching data to the search word will keep their charges; otherwise the charge will leak to the ground, as shown in Fig. 7.2a. Therefore, a CAM cell should connect a path to the ground in case of a mismatch between the data it is holding and the data assigned to its column.

Binary data are stored on the memristor device in the form of “High” and “Low” resistances respectively. Therefore, the device can work as a storage element and a switch at the same time, as in the “2T2M” cell. The charges on the row stray capacitance leak the mismatched cell, where the memristor and the series transistor will be of low resistances creating a path to the ground, as shown in Fig. 7.2b. The data are stored in a “2T2M” cell in a complimentary mode, since a high resistance device will not leak charges to the ground even in case of a mismatch, contrary to its complementary device. Fig. 7.2c

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>12</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Diodes</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Memristors</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AP Ready</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 7.1: Various CAM cells candidates.
Figure 7.2: (a) Typical evaluation phase of a CAM array and (b-d) the various states of a “2T2M” CAM cell. In (b-d), the “light gray” color represents an open switch and the “dark blue” color represents a closed switch.

shows the state of the CAM cell in the event of a match, where no paths to the ground are available. A “Don’t” care state can be stored in the cell by setting its two memristors to “High” resistance, where no path is created to the ground independent of the search bit, as shown in Fig. 7.2d.

### 7.2.2 Driving Circuitry

Searching a CAM array is done in two phases, “Pre-charge” and “Evaluate”. During the “Pre-charge” phase, all the rows are connected to “VDD” to charge their stray capacitance, where the entire cells selector are switched off. The search word is applied to the array column in the “Evaluate” phase. The stored charges will leak through the mismatched cells, and only a row with all match cells will keep its match output at logic ‘1’.
Fig. 7.3 shows the proposed driving circuitry. During the “Pre-charge” phase, transistors $M_{ri}$ will connect the source voltage to the rows to charge their stray capacitance, where ‘i’ is the row number. At the same time, transistors $M_{aj}$ will force the cell transistors to the “OFF” state, by connecting their gates to the ground, where ‘j’ is the row number. After the stray capacitors are fully charged, the “Evaluate” phase will start by switching of transistors $M_{ri}$ and $M_{aj}$. During the “Evaluate” phase, the search word will be connected to the array columns through the transmission gate of transistors $M_{bj}$ and $M_{cj}$.

Writing to the CAM memory in an AP system is done in a one column at a time.
scheme. However, this is translated into two writing steps since a complimentary data column is selectively made of two columns of the CAM array. In general, the “Write” operation is a hybrid between the “Pre-charge” and “Evaluate” operations. The bits to write are loaded to the match lines of the rows, and a search word of ‘1’ or ‘0’ at the column of interest and “Don’t Care” everywhere else is written in the columns to activate the column of interest. The active bit will be written as ‘1’ and ‘0’ in turn to write in the column and its complement respectively. This will eliminate the need for any modification to the column driving circuitry used for reading.
REFERENCES


Appendix A:
Device Models

A.1 MATLAB Model

```matlab
function resistance = memristor(time, voltage, r_i, r_off, r_on, d, u_v)
    % MEMRISTOR Numerical solution for memristor device
    
    % MEMRISTOR(time, v)
    % MEMRISTOR(time, voltage, r_i)
    % MEMRISTOR(time, voltage, r_i, r_off)
    % MEMRISTOR(time, voltage, r_i, r_off, r_on)
    % MEMRISTOR(time, voltage, r_i, r_off, r_on, d)
    % MEMRISTOR(time, voltage, r_i, r_off, r_on, d, u_v)
    
    % MEMRISTOR(time, voltage, ...) returns the numerical solution of
    % the resistance of the memristor for a arbitrary voltage and time
    % vectors.
    %
    % note: time and the voltage vectors have to be of the same length.
    %
    % Function Parameters:
    % time: time vector
    % v: voltage vector
    % r_i: initial resistance of the device (default: 1K)
    % r_off: off (maximum) resistance of the device (default: 38K)
    % r_on: on (minimum) resistance of the device (default: 0.1K)
    % d: device length in nm (default: 10nm)
    % u_v: dopant drift mobility of the device material
    % (default: 10^−14)
    
    Example:
    %
    t = 0:0.001:4;
    v = 1.5*sin(2*pi*t); % f = 1Hz
    r = memristor(t, v, 2*10^3);
    plot(t, r)
    %
    i = v./r;
    figure
    plot(i, v)
    %
    % Authors:
    % M. Affan Zidan, A. G. Radwan, and K. N. Salama
    % King Abdullah University of Science and Technology
    % Email: {mohammed.zidan,ahmed.radwan,khaled.salama}@kaust.edu.sa
    %
    % Copyright (c) 2011, M. Affan Zidan, A. G. Radwan and K. N. Salama
    % All rights reserved.
    %
```
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions
% are met:
% * Redistributions of source code must retain the above copyright
% notice, this list of conditions and the following disclaimer.
% * Redistributions in binary form must reproduce the above
% copyright notice, this list of conditions and the following
% disclaimer in the documentation and/or other materials provided
% with the distribution
% * Neither the name of the King Abdullah University of Science and
% Technology (KAUST) nor the names of its contributors may be
% used to endorse or promote products derived from this software
% without specific prior written permission.

% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
% "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
% NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
% FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
% COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
% INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
% (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
% SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
% STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
% IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.

%%%%% Error checking
if nargin < 2
    error('Function expects at least time and voltage vectors');
end

% time and voltage have to be of the same length
if ~isequal(length(time), length(voltage))
    error('Time and voltage vectors have to be of the same length.');
end

%%%%% Default values
if nargin < 7
    u_v = 10^-14;
end
if nargin < 6
    d = 10*10^-9;
end
if nargin < 5
    r_on = 0.1*10^3;
end
if nargin < 4
    r_off = 38*10^3;
end
if nargin < 3
    r_i = 2*10^3;
end

%%%%% Memristor resistance
k = u_v * r_on / d^2;
rd  = r_off - r_on;
resistance(1) = r_i;
area(1) = 0;
i1 = 1;
Figure A.1: GUI interface of the MATLAB memristor behavioral simulator.

for i = 2: length(time)
    area(i) = area(i1) + 0.5*(voltage(i)+voltage(i1))*(time(i)−time(i1))
    resistance(i) = sqrt(resistance(1)^2 + 2 * k * r_d * area(i));
i1 = i;
end

% saturation condition
resistance(resistance>r_off) = r_off;
resistance(resistance<r_on) = r_on;
%eof

Fig. A.1 shows a snapshot from the GUI interface of the MATLAB code.

A.2 Verilog-A Model

阊CambiumVerilogA

module memristor (p, n);
    input p, n;
    input p, n;
    parameter real uv = 10f;
    parameter real d = 10n;
    parameter real ron = 100;
    parameter real roff = 38k;
    parameter real rin = 5k;
    real k, r1, r2, R;
    analog begin
        k = 2 * uv * ron * (roff - ron) / pow(d, 2);
        r1 = pow(rin, 2) + k * idt(V(p,n), 0);
\[
\begin{align*}
    r_2 &= \min(\text{pow}(r_0, 2), \max(r_1, \text{pow}(r_0, 2))); \\
    R &= \sqrt{r_2}; \\
    V(p, n) &\lessapprox R = I(p, n);
\end{align*}
\]

\text{end module}

// Copyright (c) 2011, M. Afan Zidan, H. Omran, A. G. Radwan
// and K. N. Salama
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or
// without modification, are permitted provided that the followin
// conditions are met:
//   » Redistributions of source code must retain the above
//     copyright notice, this list of conditions and the
//     following disclaimer.
//   » Redistributions in binary form must reproduce the above
//     copyright notice, this list of conditions and the
//     following disclaimer in the documentation and/or other
//     materials provided with the distribution
//   » Neither the name of the King Abdullah University of
//     Science and Technology (KAUST) nor the names of its
//     contributors may be used to endorse or promote products
//     derived from this software without specific prior written
//     permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
// CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
// INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
// MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
// DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
// CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
// INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
// (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
// OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
// INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
// WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
// NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
// THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
// DAMAGE.

// eof
Appendix B:
Memory Simulation Platform –
SPICE Memristor Array Creator (SMAC)

# smac_14.py
# SPICE Memristor Array Creator (SMAC)
# Version 14.0
#
# Python 3.3
# HSPICE or Cadence APS
#
# Mohammed Affan Zidan
# Sensors Lab, KAUST
# mohammed.zidan@kaust.edu.sa
#
# Created: 13 April 2013
# Last update: 18 March 2015

# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#   Redistributions of source code must retain the above
#   copyright notice, this list of conditions and the following
#   disclaimer.
#   Redistributions in binary form must reproduce the above
#   copyright notice, this list of conditions and the following
#   disclaimer in the documentation and/or other materials
#   provided with the distribution
#   Neither the name of the King Abdullah University of Science
#   and Technology (KAUST) nor the names of its contributors may
#   be used to endorse or promote products derived from this
# software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
# CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
# INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
# BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
# EXEMPLARY, OR CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT LIMITED TO,
# ( PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
# OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
## Theory of Liability

The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

---

### Print welcome message

```python
version = "V14.0"
print("\n# SPICE Memristor Array Creator (SMAC) - " + version)
print("# Mohammed Affan Zidan")
print("# Sensors Lab, KAUST")
print("# mohammed.zidan@kaust.edu.sa\n")
```

---

### Parameters Selection

#### Device

- **1**: Res
- **2**: HP

#### Resistive device

- Resistor ON resistance (Ron) = "1X" Ohms
- Resistor OFF resistance (Roff) = "1G" Ohms

#### HP SINH device

- Conduction ON (kon) = "10N"
- Conduction OFF (koff) = "10P"
- a = "3"

#### Crossbar resistance and capacitance per unit cell

- Crossbar resistance (Reb) = "5" Ohms
- Crossbar capacitance (Ccb) = "0.1f" (capacitance is not effective in OP simulation)
- Cell capacitance (Ccell) = "10p""}

#### Switching resistance

- Switching resistance (Rsw) = "7K" (For gateless, while for gated real transistor model is used)

#### Sweep

- Start size = 512
- Step = 128
- End size = 1025

#### Data Patterns:

- **1**: All Ones
- **2**: All Zeros
- **3**: Checkerd
- **4**: Interleaved
- **5**: Random
- **6**: File

#### Cell Type

- **1**: Gateless
- **2**: Gated

```python
cell_type = 1
cell_type_name = ["Gateless","Gated"]
```
# VDD Gateless
VDD_gateless = "3"
VDD_force = "1.5"

# VDD gated
VDD_row = "0.8"
VDD_col = "0.8"
VDD_cont = "0.8"

# Transistor Model for Gated
tran_len = 22
transistor_lib = tran_len.__str__() + "nm_HP.pm"

# Gateless Connection Strategy
#1: Multipoint (Default)
#2: 3 Nodes (Multipoint)
#3: Forced Node
#4: Floating Rows and Columns
#5: Grounded Rows and Floating Columns
#6: Floating Rows and Grounded Columns
#7: Grounded Rows and Columns
strategy = 3

# Multipoint Reading Nodes
# desired_row = n1
# desired_col = n2
# nrs = n3
# ncs = n4
# VDD GND
# 1: n1 n2
# 2: n1 n3
# 3: n1 n4
# 4: n2 n3
# 5: n2 n4
# 6: n3 n4

# 3 Nodes
# VDD GND
# 1: n1 n2
# 2: n1 n3
# 3: n2 n3
read_nodes = 1

# Naming arrays
pattern_name = ["All_Ones", "All_Zeros", "Checkerd", "Interleaved", "Random", "File"]
strategy_name = ["MP", "3N", "FN", "FRC", "GR_FC", "FR_GC", "GRC"]
strategy_text = ["Multipoint", "3 Nodes (Multipoint)", "Forced Node", "Floating Rows and Columns", "Ground Rows and Floating Columns", "Grounded Rows and Floating Columns"]

if strategy == 1:
    read_name = ["n1, n2", "n1, n3", "n1, n4", "n2, n3", "n2, n4", "n3, n4"]
elif strategy == 2:
    read_name = ["n1, n2", "n1, n3", "n2, n3"]
# Device type
if device == 1:
on = Ron
off = Roff
elif device == 2:
on = kon
off = koff

# Initialize
import os
import gc
import random

# Table file
table_f = open("Table_"+strategy_name[strategy -1]+"_"+read_nodes._str__()+"_"+start_size._str__()+"_"+step._str__()+"_"+end_size._str__()+"_"+.txt","w")

# Print Header
table_f.write("# SPICE Memristor Array Creator (SMAC) — "+version+"\n")
table_f.write("# Mohammed Affan Zidan\n")
table_f.write("# Sensors Lab, KAUST\n")
table_f.write("# mohammed.zidan@kaust.edu.sa\n")
table_f.write("#\n")
table_f.write("# Ron : "+Ron._str__()+" MOhm\n")
table_f.write("# Roff : "+Roff._str__()+" MOhm\n")
table_f.write("# Rcb : "+Rcb._str__()+" Ohm\n")
table_f.write("# Strategy : "+strategy_name[strategy -1]+"\n")
if strategy < 3:
table_f.write("# Nodes: "+read_name[read_nodes -1]+"\n")
table_header = "\n\nlength\tRow\tCol"
for pattern_from_set in test_set:
    table_header = table_header + "\t" + pattern_name[pattern_from_set -1] + " (Ron)" + "\t" + pattern_name[pattern_from_set -1] + " (Roff)"
table_f.write( table_header + "\n")

# Counter file
counter_f = open("Counter_"+strategy_name[strategy -1]+"_"+read_nodes._str__()+"_"+start_size._str__()+"_"+step._str__()+"_"+end_size._str__()+"_"+.txt","w")

# Print Header
counter_f.write("# SPICE Memristor Array Creator (SMAC) — "+version+"\n")
counter_f.write("# Mohammed Affan Zidan\n")
counter_f.write("# Sensors Lab, KAUST\n")
counter_f.write("# mohammed.zidan@kaust.edu.sa\n")
counter_header = "\n\nlength"
for pattern_from_set in test_set:
counter_header = counter_header + "\tR\tC\tT"
counter_f.write( counter_header + "\n")
# Loop on the array size
for l in range(start_size, end_size, step):

    # Desired Cell Location
    # 1: Diagonal
    # 2: Sweep XY
    sweep = 2
    sweep_step = 1−2  # Sweep XY

    if sweep == 1:
        location_set_x = diagonal_set
    else:
        location_set_x = range(1, l, sweep_step)
        location_set_y = location_set_x

    # Loop on the desired cells
    for desired_col in location_set_x:
        if sweep == 1:
            location_set_y = [desired_row]

        for desired_row in location_set_y:

            random.seed(111)  # imp

            data_file_number = 1;  # Initial file number
            table_line = l.__str__(l) + """"+desired_row.__str__(l)+"""
            desired_col.__str__(l)
            counter_line = l.__str__(l) + """"+desired_row.__str__(l)+"""
            desired_col.__str__(l)

            # Loop on the test patterns
            for pattern in test_set:

                print("\n\nArray Length: "+l.__str__(l))
                print("[row, col]: "+desired_row.__str__(l)+"", "+
                desired_col.__str__(l)
                print("Pattern: "+pattern_name[pattern−1])

                file_name = "array"+l.__str__(l)"_"+l.__str__(l)+".sp"

                arrayLength = l

    # Open output file
    f = open(file_name, "w")
# Print Header
f.write(file_name+"\n")
f.write("* File generated using SPICE Memristor Array
Creator (SMAC)\n")
f.write("* +version+"\n")
f.write("=n")

f.write("* Mohammed A. Zidan\n")
f.write("* Sensors Lab, KAUST\n")
f.write("* mohammed.zidan@kaust.edu.sa\n")

f.write("* Array Size: "+arrayLength.__str__() x "+
arrayLength.__str__()\n")
f.write("* Ron : "+Ron.__str__() +" MΩ\n")
f.write("* Roff : "+Roff.__str__() +" MΩ\n")
f.write("* Rcb : "+Rcb.__str__() +" Ω\n")
pattern_name_index = pattern -1
if pattern > 5:
    pattern_name_index = 5
f.write("* Date Pattern : "+pattern_name[pattern_name_index]+"\n")

f.write("* Cell Type : "+cell_type_name[cell_type -1]+"\n")

f.write("* Connection Strategy : "+strategy_text[strategy -1]+"\n")
if strategy < 3:
    f.write("* Read Node: "+read_name[read_nodes -1]+"\n")

f.write("* Desire Cell Location: ("+desired_row.__str__()
")

# Memristor Device Subckt
f.write("\n\n* Memristor Device Subckt\n")
f.write(".subckt memr_dev p n mv = 1\n")
if device == 1:
    f.write("R_mem_cell p n mv\n")
elif device == 2:
    f.write("G_mem_cell p n CUR = 'mv * sinh ( "+a+" + v(p,n)')\n")
f.write(".ends\n")

# Memory Cell Subckt
if cell_type == 2:
    f.write("\n\n* Memory Cell Subckt\n")
    f.write(".include "+transistor_lib+"\n")
    f.write(".subckt mem_cell nd ng ndn mc = 1\n")
    f.write("X_mem_dev nd ni memr_dev mv = mc\n")
    #f.write("= \d g s b\n")
    f.write("=nnl ni ng ngd ndn nmw ="+tran_len._str_()+"\n")
    f.write("=n l="+tran_len._str_() +"n\n")
f.write(".ends\n")
# haspice64 has a limitation of max 8 char per name
res_counter = 1
tran_counter = -1
tones_t_counter = 0  # Total number of ones
tones_r_counter = 0  # Number of ones per row
tones_c_counter = 0  # Number of ones per colmn

# Row resistances and capacitances
f.write("\n\n\nRow Resistances\n")

# Rows nodes are named as ,
# m_1_1  -------  m_1_2  -------  m_1_3  -------  ...  -------  m_1_L
# m_2_1  -------  m_2_2  -------  m_2_3  -------  ...  -------  m_2_L

for i in range(1, arrayLength+1):  # Loop on rows (i : row number)
    for j in range(1, arrayLength):  # Loop on columns (j : col number)
        start_node = "m"+(j + (i-1) * arrayLength).
        end_node = "m"+(j+1 + (i-1) * arrayLength).
        f.write("R"+res_counter.__str__() + "+start_node
" +"+end_node= "+Rcb.__str__() +"\n"")
        f.write("C"+cap_counter.__str__() + "+start_node
" +" 0 "+Ccb.__str__() +"IC=0\n")
        res_counter = res_counter +1
cap_counter = cap_counter +1

# Column resistances and capacitances
f.write("\n\n\nColumn Resistances\n")

# Column nodes are named as ,
# n_1_1  -------  n_1_2  -------  n_1_3  -------  ...  -------  n_1_L
# n_2_1  -------  n_2_2  -------  n_2_3  -------  ...  -------  n_2_L

for j in range(1, arrayLength+1):  # Loop on columns (j : col number)
    for i in range(1, arrayLength):  # Loop on rows (i : row number)
        start_node = "n"+(j+(i-1) * arrayLength).
        end_node = "n"+(j + i * arrayLength).
        f.write("R"+res_counter.__str__() + "+start_node
" +"+end_node= "+Rcb.__str__() +"\n"")  # Res
        f.write("C"+cap_counter.__str__() + "+start_node
" +" 0 "+Ccb.__str__() +"IC=0\n")  # Res
        res_counter = res_counter +1
cap_counter = cap_counter +1
# Cells Resistances
f.write("\n\n# Cells Resistances\n")

# Open data file
if pattern == 6:
    data_file = open( data_file_number.__str__() + " .txt"
)
    print("File Name: " + data_file_number.__str__() + " .txt"
)

for i in range(1, arrayLength + 1):  # Loop on rows (i: row number)
    for j in range(1, arrayLength + 1):  # Loop on columns (j: col number)
        cell_val = off
        if pattern == 1:
            cell_val = on
        elif pattern == 2:
            cell_val = off
        elif pattern == 3:
            if i%2 == j%2:
                cell_val = on
        elif pattern == 4:
            if j%2 == 0:
                cell_val = on
        elif pattern == 5:
            if random.randint(2) == 1:
                cell_val = on
        elif pattern == 6:
            if data_file.read(1) == "1":
                cell_val = on

if i==desired_row and j==desired_col:
    cell_val = "-9";

# Count ones
if cell_val == on:
    ones_t_counter = ones_t_counter + 1
    if i==desired_row:
        ones_r_counter = ones_r_counter + 1
    if j==desired_col:
        ones_c_counter = ones_c_counter + 1

# Start and end nodes
start_node = "n"+(j+(i-1)*arrayLength).__str__()
end_node = "n"+(j+(i-1)*arrayLength).__str__()

if cell_type == 1:
    # Gate-less Cell
    f.write("X"+res_counter.__str__() + " + start_node+"+end_node+" memr_dev mv ="
            +cell_val+"\n")
else:
    # Transistor Cell
    f.write("X"+res_counter.__str__() + " + start_node+"+end_node+" 0 mem_cell mc="
            +cell_val+"\n")

# Cell Capacitance
f.write("C" + cap_counter.__str__() + " +start_node + " +end_node+ " +Ccell+ " IC=0\n")

res_counter = res_counter + 1
cap_counter = cap_counter + 1

# Close data file
if pattern == 6:
    data_file.close()
data_file_number = data_file_number + 1

# Write counters output
counter_line = counter_line + "t" + ones_r_counter.__str__() + "t" + ones_c_counter.__str__() + "t" + ones_t_counter.__str__()

# Gated Switching
if cell_type == 2:
    # Voltage Source
    f.write("\n\nVoltage Sources\n")
    f.write("Vrow n_vrow 0 +VDD_row+ "\n")
    f.write("Vcol n_vcol 0 +VDD_col+ "\n")
    f.write("Vcont n_vcont 0 +VDD_cont+ "\n")

    # Row Switches
    f.write("\n\nRows Switches\n")
    for i in range(1, arrayLength + 1):
        start_node = "m" + ((i - 1) * arrayLength + 1).__str__()
        tran_counter = tran_counter + 2
        if i == desired_row:
            #
            g = s = b
d
        f.write("mr" + tran_counter.__str__() + " +start_node+ " n_vrow n_vrow pmos w=" +
        tran_len * 4).__str__() + "|n l=" + tran_len.__str__() + "n\n")

        f.write("mr" + (tran_counter + 1).__str__() + " +start_node+ " 0 0 0 nmos w=" +
        tran_len * 2).__str__() + "n l=" + tran_len.__str__() + "n\n")

    else:
        #
        g = s = b
d
        f.write("mr" + tran_counter.__str__() + " +start_node+ " n_vcont n_vcont n_vcont
        pmos w=" + (tran_len * 4).__str__() + "n l=" +
        tran_len.__str__() + "n\n")

        f.write("mr" + (tran_counter + 1).__str__() + " +start_node+ " n_vcont 0 0
        nmos w=" + (tran_len * 2).__str__() + "n l=" +
        tran_len.__str__() + "n\n")

    # Column Switches
    f.write("\n\nColumn Switches\n")
    for j in range(1, arrayLength + 1):
        start_node = "n" + (j + (arrayLength - 1) * arrayLength).__str__()
        tran_counter = tran_counter + 2
if j == desired_col:
    #
    s = b
d
    f.write("mc"+trans_counter.__str__(" + "
    start_node+" n_vcol n_vcol nmos w="+(trans_len+4).__str__(" + "
    n l="+trans_len.
    __str__(" + "
    n n"")
    f.write("mc"+(trans_counter+1).__str__(" + "
    start_node+" n_vcont n_vcont n_vcont
    nmos w="+(trans_len+4).__str__(" + "
    n l="+trans_len.
    __str__(" + "
    n n"")
else:
    
    s = b
d
    f.write("mc"+trans_counter.__str__(" + "
    start_node+" n_vcont n_vcont n_vcont
    nmos w="+(trans_len+4).__str__(" + "
    n l="+trans_len.
    __str__(" + "
    n n"")
    f.write("mc"+(trans_counter+1).__str__(" + "
    start_node+" n_vcont n_vcont n_vcont
    nmos w="+(trans_len+2).__str__(" + "
    n l="+trans_len.
    __str__(" + "
    n n"")

# Gateless Switching
if cell_type == 1:
    
    # Gateless Connection Strategy
    # 1: Multipoint
    # 2: 3 Nodes
    # 3: Forced Node
    # 4: Floating Rows and Columns
    # 5: Grounded Rows and Floating Columns
    # 6: Floating Rows and Grounded Columns
    # 7: Grounded Rows and Columns
    
    # Shorting Resistances
    f.write("\n\n# Shorting Rows Resistances

# Shorting rows
short_node = \"nrs\"
if strategy == 1 or strategy == 2 or strategy == 3
    or strategy == 5 or strategy == 7:
        for i in range(1, arrayLength+1):
            if i != desired_row:
                start_node = \"m\"+(i-1)*arrayLength+1.
                __str__("
                f.write("R"+res_counter.__str__(" + "
                start_node+" +short_node+" +Rsw+\n                n"

                res_counter = res_counter +1

    f.write("\n\n# Shorting Column Resistances\n")
# Sorting columns
short_node = "ncs"
if strategy == 1 or strategy == 2 or strategy == 3
    or strategy == 6 or strategy == 7:
        for j in range(1, arrayLength + 1):
            if j != desired_col:
                start_node = "n" + (j + (arrayLength - 1) * arrayLength).__str__()
f.write("R" + res_counter.__str__() + " +
    start_node + " +short_node + " +Rsw + \
    n")
res_counter = res_counter + 1

# Voltage Supply and Ground

# desired_row = n1
# desired_col = n2
# nrs = n3
# ncs = n4

# Multipoint

# VDD  GND
# 1: n1 n2
# 2: n1 n3
# 3: n1 n4
# 4: n2 n3
# 5: n2 n4
# 6: n3 n4

# 3 Nodes

# VDD  GND
# 1: n1 n2
# 2: n1 n3
# 3: n2 n3

# VDD
f.write("\n\n\n\nVoltage Supply\n")

# Readable Code
if strategy == 1:
    if read_nodes == 1 or read_nodes == 2 or read_nodes == 3:
        f.write("Vsupply m" + (desired_row = arrayLength
            ).__str__() + " 0 " +VDD_gateless)
    elif read_nodes == 4 or read_nodes == 5:
        f.write("Vsupply n" + (desired_col =
            arrayLength - 1) * arrayLength).__str__() + " 0 " +VDD_gateless)
    else:
        f.write("Vsupply nrs 0 " +VDD_gateless)
    elif strategy == 2:
        if read_nodes == 1 or read_nodes == 2:
            f.write("Vsupply m" + (desired_row = arrayLength
                ).__str__() + " 0 " +VDD_gateless)
        else:
            f.write("Vsupply n" + (desired_col =
                arrayLength - 1) * arrayLength).__str__() + " 0 " +VDD_gateless)
else:
    f.write("Vsupply m"+(desired_row+arrayLength).__str__()\n"+"0"+VDD_gateless)

# GND
f.write("\n\nGound (Probe)\n")

# Readable Code
if strategy == 1:
    if read_nodes == 1:
        f.write("Vcurrent n"+(desired_col+(arrayLength-1)+arrayLength).__str__()\n"+"0 0")
    elif read_nodes == 2 or read_nodes == 4:
        f.write("Vcurrent nrs 0 0")
    else:
        f.write("Vcurrent ncs 0 0")
elif strategy == 2:
    if read_nodes == 1:
        f.write("Vcurrent n"+(desired_col+(arrayLength-1)+arrayLength).__str__()\n"+"0 0")
    else:
        f.write("Vcurrent nrs 0 0")
else:
    f.write("Vcurrent ncs 0 0")

# Forced Nodes
f.write("\n\nForced Nodes\n")
if strategy == 2:
    # f.write("\n\nShort Rows and Columns\n")
    f.write("R"+res_counter.__str__()\n"+"nrs ncs 0 0.01 \n")
    res_counter = res_counter + 1
elif strategy == 3:
    f.write("R"+res_counter.__str__()\n"+"nrs ncs 0 0.01 \n")
    res_counter = res_counter + 1
f.write("Vforce nrs 0 "+VDD_force)
elif strategy == 5:
    f.write("R"+res_counter.__str__()\n"+"n"+(desired_col+(arrayLength-1)+arrayLength).\n"+"nrs 0.01 \n")
    res_counter = res_counter + 1
elif strategy == 6:
    f.write("R"+res_counter.__str__()\n"+"n"+(desired_col+(arrayLength-1)+arrayLength).\n"+"nrs 0.01 \n")
    res_counter = res_counter + 1
elif strategy == 7:
    f.write("R"+res_counter.__str__()\n"+"n"+(desired_col+(arrayLength-1)+arrayLength).\n"+"nrs 0.01 \n")
    res_counter = res_counter + 1
f.write("R"+res_counter.__str__()\n"+"n"+(desired_col+(arrayLength-1)+arrayLength).\n"+"nrs 0.01 \n")
    res_counter = res_counter + 1
# Analysis
f.write("\n\n  Operating Point Analysis\n")
f.write(".OP\n")
f.write(".PRINT OP I(Vcurrent)\n")
f.write("\n.END")
f.close()

# Desired cell binary cases
for desired_cell in [on, off]:
    temp_f_name = "array_" + l.__str__() + "_" + l.__str__() + "+desired_cell.__str__()"
    temp_f = open(temp_f_name + ".sp", "w")
    output_file_name = temp_f_name + ".print"

    f = open(file_name, "r")
    for line in f:
        temp_f.write(line.replace("-9", desired_cell.__str__() + ")")
    f.close()
    temp_f.close()

    # HSPICE
    os.system("hspice64 "+temp_f_name+".sp > "+output_file_name+" -mt 8 -hpp")

    # Cadence APS
    os.system(="/aps "+temp_f_name+".sp +spice +aps= 
moderate +mt=8 > log.txt")

# Brace Output File

# Open File
out_f = open(output_file_name, "r")

# Find the line containing the current values
k = 0
current_line_number = -1
for line in out_f:
    if "i(Vcurrent)" in line:
        current_line_number = k + 1
    if current_line_number == k:
        current_line = line
\[ k = k + 1 \]

```
current_line = current_line.replace("-","")
current_line = current_line.replace(" m","e-3")
current_line = current_line.replace(" u","e-6")
current_line = current_line.replace(" n","e-9")
current_line = current_line.replace(" p","e-12")
current_line = current_line.replace(" f","e-15")

words = current_line.split()

# HSPICE
if cell_type == 1:
    table_line = table_line +"\t"+words[2]
else:
    table_line = table_line +"\t"+words[1]

# Cadence APS (Gateless)
#table_line = table_line +"\t"+words[1]

out_f.close()

# Cleaning
os.system("rm -f -r " + output_file_name)
os.system("rm -f -r " +temp_f_name+"_dp_lis")
os.system("rm -f -r " +temp_f_name+".print")
os.system("rm -f -r " +temp_f_name+".st0")
os.system("rm -f -r " +temp_f_name+".ic0")
os.system("rm -f -r " +temp_f_name+".sp")

gc.collect()

# Cleaning
gc.collect()

os.system("rm -f -r "+array_"+l._str_()"+l._str_()

# Flush the output
counter_f.write(counter_line+"\n")
counter_f.flush()

table_f.write(table_line+"\n")
table_f.flush()
gc.collect()

# Cleaning
gc.collect()
table_f.flush()
gc.collect()
```
table_f.close()

# eof #####################################################################
Appendix C:
List of Publications

C.1 Memristor & Memory Related Publications

Journals:


Journals Under Preparation:


Peer Reviewed Conferences:


Patents:


Posters:


3 M. Affan Zidan, and K. N. Salama, “Memristor-Based Memories: Challenges and Opportunities,” The Second Saudi International Electronics, Communications and Photonics Conference (SIECPC), Riyadh, Saudi Arabia, April 2013 (Best Graduate Poster Award)

C.2 Other Publications

Journals:


Peer Reviewed Conferences:


Patents:


