



**João Vidal Correia**

**Desenho Lógico com Memristors**

**Logic Design with Memristors**





**João Vidal Correia**

**Desenho Lógico com Memristors**

**Logic Design with Memristors**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Luís Filipe Mesquita Nero Moreira Alves, Professor Auxiliar do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro e do Doutor Ernesto Fernando Ventura Martins, Professor Auxiliar do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro



**o júri / the jury**

presidente / president

**Prof. Doutor Rui Manuel Escadas Ramos Martins**

Professor Auxiliar da Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)

vogais / examiners committee

**Prof. Doutor Luis Filipe Mesquita Nero Moreira Alves**

Professor Auxiliar da Universidade de Aveiro

**Prof. Doutor Vítor Manuel Grade Tavares**

Professor Auxiliar da Universidade do Porto



## **agradecimentos / acknowledgements**

Em primeiro lugar, agradeço aos meus pais e irmã pelo apoio moral que me deram ao longo destes 6 anos e pela pessoa que sou hoje (podia ser pior).

Agradeço aos meus orientadores, o Professor Doutor Luís Nero Alves e o Professor Doutor Ernesto Ventura Martins, por serem o meu guia neste percurso e porque sem os quais este trabalho não seria possível.

Agradeço também aos meus amigos por todos os momentos de diversão que me proporcionaram, pela (muita) paciência para me aturar, mas especialmente por, nas alturas mais difíceis, me lembrarem que "o que vale é que não sou o pika".

Agradeço aos meus colegas do Laboratório de Circuitos e Sistemas Integrados do Instituto de Telecomunicações pelas tardes de trabalho muito bem passadas.

Finalmente, agradeço ao Instituto de Telecomunicações por me acolher durante este percurso.





## Palavras-chave

Memristor, Desenho Lógico, Família Lógica, Porta Lógica

## Resumo

Em 1971, Leon Chua reparou que uma relação entre carga elétrica e fluxo magnético estava em falta e, então, propôs teoricamente um elemento de circuito que estabelece tal relação, ao qual deu o nome de memristor, pois este dispositivo é essencialmente uma "resistência com memória". Mais tarde, em 1976, uma generalização para dispositivos memristivos foi produzida, definindo um dispositivo memristivo como um dispositivo cujo comportamento é definido como uma lei de Ohm dependente de estado controlada por uma variável de estado. Isto permitiu a criação de memristors sintéticos, o primeiro dos quais foi anunciado em 2008 por Stanley Williams da HP Labs.

Tem-se mostrado que os memristors têm algumas propriedades muito interessantes e únicas que permitem aplicações sem precedentes. Estes dispositivos são bastante pequenos e rápidos, com tamanhos da ordem dos nanômetros e tempos de transição da ordem dos nanossegundos, com a capacidade de serem embutidos entre duas camadas de processo, permitindo a fabricação de circuitos integrados de alta densidade e alto paralelismo. Hoje em dia, sabe-se que a diminuição da tecnologia CMOS se torna cada vez mais difícil. As características do memristor, que permitem a criação de portas lógicas, juntamente com o seu tamanho reduzido, fazem deste dispositivo um bom substituto para o transistor tradicional em circuitos lógicos. Para além disso, os materiais que compõem estes dispositivos são compatíveis com a tecnologia CMOS, o que aponta para a hipótese de integração CMOS/Memristor.

Outro aspeto tentador do memristor é que tem, simultaneamente a capacidade de armazenamento de dados e processamento lógico. Isto permite arquiteturas de computadores inovadoras para além da arquitetura de von Neumann tradicional, combinando armazenamento e processamento de dados num único circuito, removendo a necessidade de acesso à memória e, assim, contornando o problema do von Neumann bottleneck. Para este fim, é necessário um estudo sobre como os memristors realizam processamento lógico. O objetivo desta dissertação é precisamente o estudo de como operações de álgebra de Boole podem ser realizadas usando memristors como os componentes chave nestes circuitos lógicos. É apresentada uma revisão de modelos matemáticos de memristors e técnicas de desenho lógico com memristors, com algum suporte matemático, para facilitar as simulações posteriores. Estas simulações são realizadas usando um modelo matemático do memristor, cujos parâmetros são extraídos de medidas feitas num dispositivo disponível comercialmente.



**Keywords**

Memristor, Logic Design, Logic Family, Logic Gate

**Abstract**

In 1971, Leon Chua noticed that some relationship between electric charge and magnetic flux was missing and, so, he theoretically proposed a circuit element that established said relationship, to which he called a memristor, because this device is essentially a "resistor with memory". Later, in 1976, a generalization to memristive devices was produced, defining a memristive device as a device whose behavior is defined by a state-dependant Ohm's law controlled by some state variable. This allowed for the creation of synthetic memristors, the first of which was announced in 2008 by Stanley Williams from HP Labs.

Memristors have been shown to have some very interesting and unique properties which allow for unprecedented applications. These are very small and fast devices, with sizes in the order of nanometers and switching times in the order of nanoseconds, with the ability to be embedded between two process layers, allowing for highly dense chips, capable of high parallelism. Nowadays, it is known that CMOS down-scaling is becoming increasingly difficult. The characteristics of the memristor, which allow for the creation of logic gates, in addition to its small size, make this device a good substitute for the traditional transistor in logic circuits. Additionally, the materials which compose these devices are compatible with CMOS technology, which points to the prospect of CMOS/Memristor integration.

Another appealing feature of the memristor is that it is capable of both data storage and logic processing. This allows for novel computer architectures beyond the traditional von Neumann architecture, combining data storing and processing in a single circuit, removing the need for memory access, thus circumventing the von Neumann bottleneck problem. For this, a study on how memristors may perform logic processing is required. This dissertation's objective is precisely the study of how Boolean algebra operations may be performed using memristors as the key components in these logic circuits. A review on memristor mathematical models and state-of-the-art logic design techniques using memristors is presented, with some degree of mathematical background, to facilitate posterior simulation. These simulations are performed using a memristor mathematical model, the parameters of which were extracted from measurements made on an commercially available device.



# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iii</b>
<b>List of Tables</b>	<b>vii</b>
<b>Nomenclature</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objectives . . . . .	2
1.3 Structure . . . . .	2
1.4 Original Contributions . . . . .	3
<b>2 Memristor Modeling</b>	<b>5</b>
2.1 Introduction to memristors . . . . .	5
2.1.1 What is a memristor? . . . . .	5
2.1.2 Classes of Memristors . . . . .	6
2.1.3 Dynamic Route Map and Power-Off Plot . . . . .	8
2.2 Memristor Models . . . . .	10
2.2.1 Memristor Equivalent Circuit . . . . .	11
2.2.2 $\Phi$ memristor . . . . .	11
2.2.3 HP Labs Model . . . . .	13
2.2.4 Hyperbolic Sine Models . . . . .	18
2.2.5 Metastable Switch Model . . . . .	23
2.2.6 Model Classification . . . . .	26
2.3 Memristor Model Implementation in Cadence . . . . .	26
2.3.1 Model Implementation . . . . .	26
2.3.2 Model Fitting . . . . .	27
2.4 Final Remarks . . . . .	32
<b>3 Implementation of Logic Gates with Memristors</b>	<b>33</b>
3.1 Memristor Switching Dynamics . . . . .	33
3.2 Classification of Memristive Logic Families . . . . .	36
3.3 Out-of Memory computing logic families . . . . .	38
3.3.1 MRL-Memristor Aided Logic . . . . .	38
3.3.2 CMOS/Memristor Threshold Logic . . . . .	39

3.4	Near Memory computing logic families . . . . .	43
3.4.1	CMOS-like Memristor Complementary Logic . . . . .	44
3.4.2	Parallel Input-Processing Memristor Logic . . . . .	47
3.5	In Memory computing logic families . . . . .	50
3.5.1	IMPLY Logic . . . . .	50
3.5.2	MAGIC-Memristor Aided loGIC . . . . .	54
3.6	Final Remarks . . . . .	58
<b>4</b>	<b>Circuit Simulations</b>	<b>59</b>
4.1	Metrics . . . . .	59
4.2	Simulation Circuits . . . . .	60
4.3	Simulation results . . . . .	62
4.3.1	Read Time . . . . .	62
4.3.2	Evaluation Time . . . . .	63
4.3.3	Writing Time . . . . .	64
4.3.4	Reading Stage Dissipated Power . . . . .	65
4.3.5	Evaluation Stage Dissipated Power . . . . .	66
4.3.6	Writing Stage Dissipated Power . . . . .	67
4.3.7	Noise Margin . . . . .	68
4.4	Final Remarks . . . . .	70
<b>5</b>	<b>Conclusions</b>	<b>73</b>
5.1	Work Done . . . . .	73
5.2	Future Work . . . . .	74
	<b>Appendices</b>	<b>77</b>
<b>A</b>	<b>LTSpice Netlist Codes and Simulation Schematics</b>	<b>78</b>
<b>B</b>	<b>Remaining results obtained from the memristor's IV-curve measurements</b>	<b>84</b>
<b>C</b>	<b>MATLAB code used to process .csv data from PicoScope</b>	<b>86</b>
	<b>Bibliography</b>	<b>91</b>

# List of Figures

2.1	How to identify and classify memristors flowchart. . . . .	7
2.2	Venn diagram of the memristor classes . . . . .	8
2.3	Example of a POP . . . . .	9
2.4	(a): Voltage signal applied. (b): Example of a DRM of a memrsitor . . . . .	9
2.5	(a): Equivalent memristor circuit. (b): Examples of POPs with two asymptotically stable equilibrium points . . . . .	10
2.6	Memristor Equivalent Circuit . . . . .	11
2.7	IV characteristic of a $\Phi$ memristor when driven by $I(t) = 0.001 \sin(2\pi t)$ . . . .	12
2.8	IV characteristic of a $\Phi$ memristor when driven a current triangular wave ranging from 0 to 1mA and period of 1s . . . . .	13
2.9	Graph of the Joglekar Window Function . . . . .	14
2.10	IV characteristic of the HP memristor model using Joglekar Window when driven by $V(t) = 0.9 \sin(20\pi t)$ . . . . .	15
2.11	IV characteristic of the HP memrsitor model using Joglekar Window when driven by a triangular wave ranging from 0 to 1V with period of 1s. . . . .	15
2.12	Graph of the Biolek Window Function . . . . .	16
2.13	IV characteristic of the HP memristor model using the Biolek Window when driven by a tr $V(t) = \sin(20\pi t)$ . . . . .	17
2.14	IV characteristic of the HP memristor model using the Biolek Window when driven by a triangular wave ranging from 0 to 1V with period of 1s. . . . .	17
2.15	IV characteristic of the General Sinh Model when driven by $V(t) = 3 \sin(0.2\pi t)$ . . . .	18
2.16	IV characteristic of the General Sinh Model when driven by a triangular wave ranging from 0 to 2.5V with period of 100ms . . . . .	19
2.17	IV characteristic of the U. Michigan memristor model when driven by $V(t) = \sin(2\pi t)$ . . . . .	20
2.18	IV characteristic of the U. Michigan memristor model when driven by a triangular wave ranging from 0 to 1.5V with period of 1s . . . . .	20
2.19	IV characteristic of the Yakopcic model when driven by $V(t) = \sin(20\pi t)$ . . . .	22
2.20	IV characteristic of the Yakopcic model when driven by a triangular wave ranging from 0 to 250 mV with period of 100ms . . . . .	22
2.21	IV characteristic of the Knowm memristor model (blue) and Resistance vs. Voltage curve (orange) when driven by $V(t) = 0.5 \sin(200\pi t)$ . . . . .	25
2.22	IV characteristic of the Knowm memristor model when driven by a triangular wave ranging from 0 to 100mV . . . . .	25

2.23	Current vs. Voltage curve (red) and Resistance vs. Voltage curve (yellow) with $V(t) = 0.5\sin(200\pi t)$ . . . . .	27
2.24	Circuit used to measure a physical memristor's IV-characteristic . . . . .	28
2.25	IV-curves measured with $R_L = 1k\Omega$ : (a) 1Hz, (b) 10Hz, (c) 100Hz, (d) 1kHz . . . . .	28
2.26	IV-curves measured with $R_L = 10k\Omega$ :(a) 1Hz, (b) 10Hz, (c) 100Hz, (d) 1kHz . . . . .	29
2.27	IV-curves measured with $R_L = 10k\Omega$ :(a) 1Hz, (b) 10Hz, (c) 100Hz, (d) 1kHz . . . . .	29
2.28	Comparison between modeled and measured IV-characteristics . . . . .	31
3.1	(a) Forward Polarized Memristor(FPM). (b) Reverse Polarized Memristor . . . . .	33
3.2	(a) IV characteristic of a forward polarized memristor. (b) . . . . .	34
3.3	(a)-(d) Different combinations of memristors . . . . .	34
3.4	IV characteristic of the parallel composition of a FPM and a RPM . . . . .	35
3.5	IV characteristic of the series composition of a FPM and a RPM . . . . .	36
3.6	Illustration of data movement in memristive systems with different proximity of computation. (a) Out-of Memory computing. (b) Near Memory computing. (c) In Memory computing. (d) Legend for the previous block diagrams . . . . .	37
3.7	Illustration of a memristive crossbar . . . . .	37
3.8	(a) MRL OR gate. (b) MRL AND gate. . . . .	38
3.9	(a) Circuit implementation of an LTG. (b) Implementation of the memristors' switching threshold. . . . .	40
3.10	IV characteristics of a programmable memristor with two anti-parallel diodes in series . . . . .	40
3.11	Memristor programming circuit . . . . .	41
3.12	Resistance programming of a memristor . . . . .	42
3.13	Circuit used to evaluate a Threshold Logic gate . . . . .	43
3.14	CMOS-like Memristor Complementary Logic NOT Gate . . . . .	44
3.15	Circuit architecture of a CMOS-like NOR gate . . . . .	45
3.16	Circuit topology of a cmos-like NOT gate . . . . .	46
3.17	IV characteristics of composite memristive circuits . . . . .	47
3.18	Simplification of the circuit Implementation of a Parallel Input-Processing Gate . . . . .	48
3.19	Implementation of a Parallel Input Processing circuit . . . . .	49
3.20	Equivalent circuit when " $\sim ex$ " is active . . . . .	49
3.21	Equivalent circuit when " $\sim read$ " is active . . . . .	50
3.22	IMPLY Logic circuit architecture . . . . .	50
3.23	IMPLY Logic circuit using 3 memristors. . . . .	52
3.24	ImPLY circuit final design . . . . .	54
3.25	Circuit for different MAGIC gates: (a) NAND, (b) NOR, (c) AND, (d) OR, (e) NOT . . . . .	54
3.26	Circuit implementation of a N-input MAGIC NOR gate . . . . .	55
3.27	Sensing amplifier for MAGIC circuits . . . . .	57
3.28	Complete MAGIC crossbar row circuit . . . . .	58
4.1	Circuit used to simulate an MRL OR gate . . . . .	60
4.2	Circuit used to simulate a CMOS-like memristive NOT gate . . . . .	60
4.3	Circuit used to simulate a Parallel Input Processing NOT gate . . . . .	61
4.4	Circuit used to simulate a MAGIC NOT gate . . . . .	61
4.5	Results obtained for the measurements of $t_{read}$ . . . . .	62



4.6	Results obtained for the measurements of $t_{ev}$ . . . . .	63
4.7	Results obtained for the measurements of $t_{wr}$ . . . . .	64
4.8	Results obtained for the measurements of $P_{read}$ . . . . .	65
4.9	Results obtained for the measurements of $P_{ev}$ . . . . .	66
4.10	Results obtained for the measurements of $P_{wr}$ . . . . .	67
4.11	Representation of the Noise Margin . . . . .	68
4.12	Results obtained for the measurements of $NM$ . . . . .	69
A.1	HP Labs model with Joglekar Window Netlist . . . . .	78
A.2	Simulation schematic of HP Labs model with Joglekar Window . . . . .	78
A.3	HP Labs model with Biolek Window Netlist . . . . .	79
A.4	Simulation schematic of HP Labs model with Biolek Window . . . . .	79
A.5	General Hyperbolic Sine model netlist . . . . .	80
A.6	Simulation Schematic of the General Hyperbolic Sine Model . . . . .	80
A.7	U.Michigan model netlist . . . . .	81
A.8	Simulation schematic of the U.Michigan model . . . . .	81
A.9	Yakopcic model netlist . . . . .	82
A.10	Simulation schematic of Yakopcic's Model . . . . .	82
A.11	Mean Metastable Switch model netlist . . . . .	83
A.12	Simulation schematic of the Mean Metastable Switch model . . . . .	83
B.1	IV-curves measured with $R_L = 1k\Omega$ : (a) 10kHz, (b) 100kHz, (c) 1MHz . . . .	84
B.2	IV-curves measured with $R_L = 10k\Omega$ : (a) 10kHz, (b) 100kHz, (c) 1MHz . . .	85
B.3	IV-curves measured with $R_L = 100k\Omega$ : (a) 10kHz, (b) 100kHz, (c) 1MHz . .	85



# List of Tables

2.1	Definition of the four classes of memristors. . . . .	6
2.2	Classification of the presented memristor models . . . . .	26
2.3	Model parameters obtained . . . . .	31
3.1	Switch positions for all operation phases . . . . .	45
3.2	Two variable logic gates implementations . . . . .	48
3.3	Material implication truth table . . . . .	51
3.4	Basic logic operations and equivalents using implies and '0' . . . . .	51
3.5	Steps to perform an OR gate through IMPLY Logic . . . . .	52



# Nomenclature

## Acronyms

CMOS	Complementary Metal Oxide Semiconductor
CRS	Complementary Resistive Switch
DRM	Dynamic Route Map
FPM	Forward Polarized Memrsistor
FPM	Reverse Polarized Memrsistor
GND	Ground
IV	Current/Voltage
LTG	Linear Threshold Gate
MAGIC	Memristor Aided loGIC
MIM	Metal Insulator Metal
MRL	Memristor Ratioed Logic
POP	Power-Off Plot

## Units

$\Omega$	Ohm
$A$	Ampere
$Hz$	Hertz
$K$	Kelvin
$m$	meter
$Oe$	Oersted ( $1Oe = \frac{1000}{4\pi} A/m$ )
$S$	Siemens
$s$	second
$T$	Tesla

$V$  Volt

$W$  Watt

## Symbols

$\phi$	Magnetic Flux
$C$	Capacitance
$G$	Memconductance
$G_{off}$	OFF state conductance
$G_{on}$	ON state conductance
$I$	Current
$M$	Memristance
$NM$	Noise Margin
$P_{ev}$	Evaluation Stage Dissipated Power
$P_{read}$	Reading Stage Dissipated Power
$P_{wr}$	Writing Stage Dissipated Power
$q$	Charge
$R$	Resistance
$R_{off}$	OFF state resistance
$R_{on}$	ON state resistance
$t_{ev}$	Evaluation time
$t_{read}$	Reading time
$t_{wr}$	Writing time
$V$	Voltage
$V_{DD}$	Voltage Drain Drain
$V_{ev}$	Evaluation voltage

$V_{read}$	Reading voltage
$V_{reset}$	RESET voltage
$V_{set}$	SET voltage
$V_{SS}$	Voltage Source Source
$V_{wr}$	Writing voltage
$x$	State Variable

### Constants

$\mu_0$	permeability of vacuum $= 1.25663706 \times 10^{-6} m.kg.s^{-2}.A^{-2}$
$\pi$	Pi $\approx 3.14159265359$
$e$	Euler's number $\approx 2.718281828459$
$k_B$	Boltzmann's constant $= 1.38064852 \times 10^{-23} m^2.kg.s^{-2}.K^{-1}$

# Chapter 1

## Introduction

In 1971, Leon Chua noticed that, out of all 6 possible two-by-two combinations of the four electric variables voltage, current, charge and magnetic flux, the relationship between charge and magnetic flux was unaccounted for. This drove Chua to propose a new circuit element called the memristor, since it is essentially a "resistor with memory". Later, in 1976, some generalizations on memristive systems were produced, defining a memristor as a state-dependant Ohm's law device. This device was pretty much forgotten, until the creation of a memristor in 2008 at HP Labs.

### 1.1 Motivation

Since 2008, when it was announced by HP Labs the discovery of the memristor [1], a newly found interest in these devices bloomed, in particular, for its potential applications in electronics and computation. The properties these devices display are so unique that its arrival brought along with it an amalgam of unprecedented potential. These are very small and fast devices, with size in the order of manometers and switching times in the order of nanoseconds [2]. Memristors can be embedded between two process layers, allowing for high density and high parallelism chips. It is known today, that Moore's law is failing, because of the increasing difficulty in down-scaling CMOS components. The characteristics the memristor presents, which allow the creation of logic gates, in addition to its small size, make this device a potential substitute for the traditional transistor in logic circuits. Also, memristors may be built out of CMOS compatible materials, which allows for CMOS/Memristor 3D integration.

The property of resistance switching of the memristor make it possible for them to be used in reconfigurable devices, such as, on-the-fly configurable amplifiers, filters, sensors and other devices.

Another property of the memristor is that its behavior is similar to the synapses of our brains. This, combined with the high parallelism offers the chance to build neuromorphic systems, systems that mimic the structure of the human brain, which is much more efficient than modern computers. The memristor is capable, simultaneously, of data storage and logic processing. This property of the memristors points to the prospect of novel computer architectures, beyond the traditional von Neumann. By combining data storage and processing all in a single circuit, removing the need for memory access by the CPU, solving the von

Neumann bottleneck problem. For this, a study on how memristors may perform this logic processing is needed. The intent behind this work is to understand the mechanisms behind the strategies to build memristive logic gates.

## 1.2 Objectives

The prime objective of this dissertation is to study how memristors can perform logic computations, by means of common Boolean algebra and some slight modifications to it. To this effect the following work plan was proposed:

- Review on memristors and memristor mathematical models;
- Measure the IV curve of a real memristor;
- Choose a mathematical model and implement it on Cadence, using the measurements previously obtained;
- Study logic design techniques with memristors;
- Design and simulate logic circuit;

## 1.3 Structure

This dissertation is divided into 5 chapters:

- In chapter 2, memristor mathematical modeling is introduced. In the first section of this chapter, the device known as memristor is presented, as well as some tools to better understand the behavior of memristors. In the second section, some memristor mathematical models are presented and discussed. In the third and final section of this chapter, some measurements made on a physical memristor are presented, in order to extract some parameters to insert into a memristor model implemented in the Cadence simulator. The implementation of the model is also discussed.
- Chapter 3 discusses the implementation of logic gates using memristors. The first section discusses the switching dynamics of a single memristor as well as compositions of several memristors, making use of a simplified view of the behavior of a memristor. The second section discusses how memristive logic families may be classified into three different categories. The next three sections present some logic families that belong to these categories.
- Chapter 4's purpose is to show the results of some simulations conducted on some of the circuits presented in chapter 3. In the first section, some metrics are defined to then compare the performance of the different logic families. In the following section, the circuits used for the simulations are presented and the general conditions of the experiments are also presented. In the third section, the results of the simulations performed are presented and discussed.
- Chapter 5 summarizes the work done so far and the conclusions that can be derived from the results obtained. Additionally, some suggestions for future work are also presented.



## 1.4 Original Contributions

This work contributed with a comparative study of the performance of different memristive logic families. A similar work has not been published to this day.



## Chapter 2

# Memristor Modeling

In the following chapter, the memristor is introduced. It is shown that memristors can belong to different classes, characterized by different equations and behaviors. Also some tools required to better understand memristors are also discussed. Next, several mathematical models of memristors are discussed and, finally, their implementation on the Cadence simulator is presented.

### 2.1 Introduction to memristors

This section gives a brief introduction to memristors. Firstly, it discusses how it came to be theoretically proposed. Then, it is shown that several devices which can be identified as memristors fit into different classes. Finally, some tools and theorems useful to the understanding of memristor behavior, such as dynamic route maps the "two is infinite" theorem, are presented.

#### 2.1.1 What is a memristor?

The memristor is a two-terminal device which was theoretically proposed by Leon Chua in 1971 in [3]. Later, in 2008, was physically recognized at HP Labs. This discovery arose when Leon Chua noticed that considering the four fundamental electric variables voltage, current, charge and magnetic flux, it is possible to establish 6 two-by-two relationships. Out of these, five are well-known fundamental definitions. However, no such relationship was known for charge and magnetic flux and so, for the sake of completeness, Chua proposed a new circuit element which established the constitutive relation  $g(\varphi, q) = 0$ . The voltage across this device is:

$$v(t) = M(q(t))i(t) \tag{2.1}$$

$$M(q) = \frac{d\varphi(q)}{dq} \tag{2.2}$$

The equations above show something interesting: the resistance of the device is dependent on the charge (which is the integral of the current) passing through it. In other words, it depends on past values of the current, making it essentially a resistor with memory. Hence the name memristor. The case presented above is for what is called a current-controlled memristor. By the same reasoning, for a voltage-controlled memristor:

$$i(t) = G(\varphi(t))v(t) \quad (2.3)$$

$$G(\varphi) = \frac{dq(\varphi)}{d\varphi} \quad (2.4)$$

Since most circuits presented in this dissertation will be voltage-controlled, only the voltage-controlled case will be treated. However, the same reasoning can be applied to the current-controlled case.

Such a device possesses some interesting properties. One of the most prominent of these properties is that, when a passive memristor is driven by a sinusoidal signal with zero mean, the v-i curve observed is a hysteresis loop which passes through the first and third quadrant, since it is a passive component, and through the origin, and this loop tends to a straight line when the signal's frequency tends to infinity. This and other memristor properties are proved in [4].

### 2.1.2 Classes of Memristors

The mathematical model described above is what is called an ideal memristor. The equations which describe physical memristors can be more complex. So, for the sake of convenience, Leon Chua in [5], introduced four classes of memristors, attending to the two equations that define their behavior: the device's I-V relationship and its state equation. The four classes and the respective definition of each are described in Tab.2.1.

Table 2.1: Definition of the four classes of memristors.

Class	Voltage Controlled	Current Controlled
Extended	$I = G(x, V)V, G(x, 0) \neq \infty$ $\frac{dx}{dt} = g(x, V)$	$V = M(x, I)I, M(x, 0) \neq \infty$ $\frac{dx}{dt} = f(x, I)$
Generic	$I = G(x)V$ $\frac{dx}{dt} = g(x, V)$	$V = M(x)I$ $\frac{dx}{dt} = f(x, I)$
Ideal Generic	$I = G(x)V$ $\frac{dx}{dt} = g(x)V$	$V = M(x)I$ $\frac{dx}{dt} = f(x)I$
Ideal	$I = G(\varphi)V$ $\frac{d\varphi}{dt} = V$	$V = M(q)I$ $\frac{dq}{dt} = I$

In the same paper, Chua also described a sequence of tests designed to identify and classify a memristor. The process of identifying and classifying memristors is described in the flowchart shown in Fig.2.1.

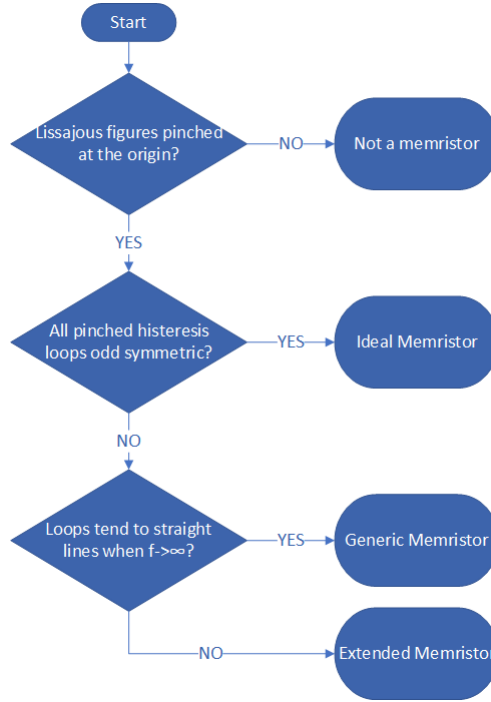


Figure 2.1: How to identify and classify memristors flowchart.

It is important to notice that the state equation in the ideal and ideal generic classes is a separable variables differential equation, which can be directly solved as follows:

- Ideal Memristor

$$\begin{aligned}
 \frac{d\phi(t)}{dt} &= V(t) \Leftrightarrow \\
 \Leftrightarrow \int_{\phi(0)}^{\phi(t)} d\phi(t) &= \int_0^t V(t)dt \Leftrightarrow \\
 \Leftrightarrow \phi(t) &= \phi(0) + \int_0^t V(t)dt
 \end{aligned} \tag{2.5}$$

- Ideal Generic Memristor

$$\begin{aligned}
 \frac{dx(t)}{dt} &= g(x(t))V(t) \Leftrightarrow \\
 \Leftrightarrow \frac{1}{g(x(t))}dx(t) &= V(t)dt
 \end{aligned} \tag{2.6}$$

By definition  $d\phi(t) = V(t)dt$ . Substituting this definition into Eq.2.6 and integrating:

$$\phi(t) = \phi(0) + \int_{x(0)}^{x(t)} \frac{1}{g(x(t))}dx \tag{2.7}$$

By close observation of the equations that describe each memristor class, it is possible to observe that these classes are nested within each other as shown in Fig.2.2.

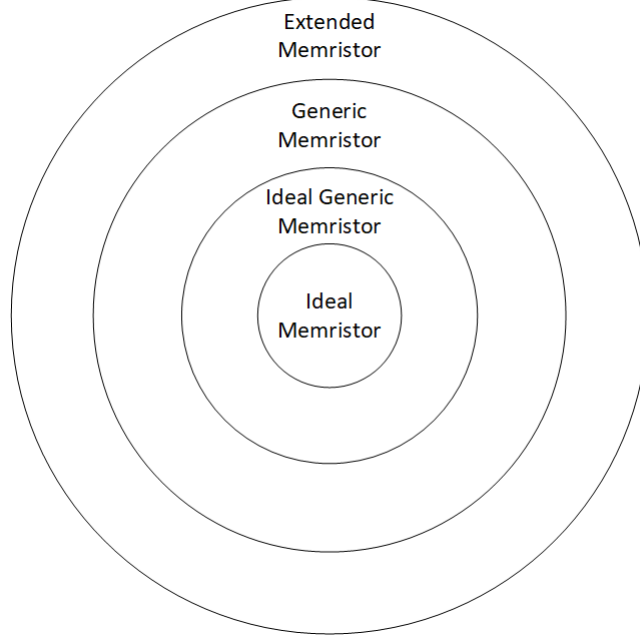


Figure 2.2: Venn diagram of the memristor classes

### 2.1.3 Dynamic Route Map and Power-Off Plot

A very useful tool to study the behavior of a memristive device is called a Dynamic Route, which is the plot of the state equation  $f(x, V) = \frac{dx}{dt}$  as a function of the state variable  $x$ , for a given voltage  $V$  [5]. This plot gives useful information about the evolution of the state variable when some voltage is applied to the memristor. Since this plot has the parameter  $V$ , several plots can be traced for different voltages. The collection of all Dynamic Routes parametrized by the voltage  $V$  is called the Dynamic Route Map (DRM). The plot generated when  $V = 0V$  provides even more information. It shows how the memristor's state variable changes over time when it is not being driven. From this, some information can be deduced regarding the volatility of the device, how many stable states it has and how to switch between them. The plot of  $f(x, 0)$  is called the Power-Off Plot (POP).

In the POP shown in Fig.2.3, three points are highlighted: Q0, Q1 and Q2. These are called the equilibrium points, because the motion of  $x$  is 0 in those points. However, the motion of  $x$  diverges from the equilibrium point Q1. This means that a infinitesimal perturbation to  $x$  causes it to converge to one of the other two equilibrium points. Q1 is then classified as an unstable equilibrium point whereas the points Q0 and Q2 are classified as asymptotically stable equilibrium points. They are called asymptotically stable because, while  $x$  may converge to these points, it never equals them in finite time. Generally speaking, whenever the Dynamic Route intersects the  $x$ -axis with a negative slope, the intersect point is an asymptotically stable equilibrium point. Conversely, if it intersects the  $x$ -axis with a positive slope, the intersect point is an unstable equilibrium point.

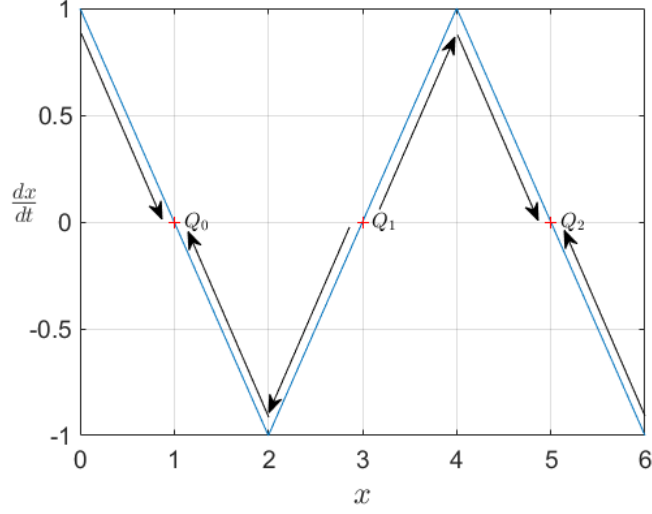


Figure 2.3: Example of a POP

These asymptotically stable equilibrium points present in the POP are what allow the memristor to behave like a memory, since the state variable tends to their value when the memristor is not being driven.

To switch between states, a voltage pulse of a certain duration and amplitude should be applied, making the state variable converge to a new equilibrium point, such that, when  $V = 0V$  again, the state variable tends to the desired point. An example of this can be seen in Fig.2.4. In this example, the initial state of the memristor is represented by  $x(t_0)$ . Since the POP is always equal to 0, this state doesn't change while  $V = 0V$ . However, when a positive voltage is applied in  $t_1$ , the motion of  $x$  is positive, so  $x$  must move in the positive direction until  $V = 0V$  again at  $t_2$ . At  $t_3$ , a negative voltage is applied and the derivative of  $x$  is negative so  $x$  must move backwards until  $t_4$  when  $V = 0V$  once again.

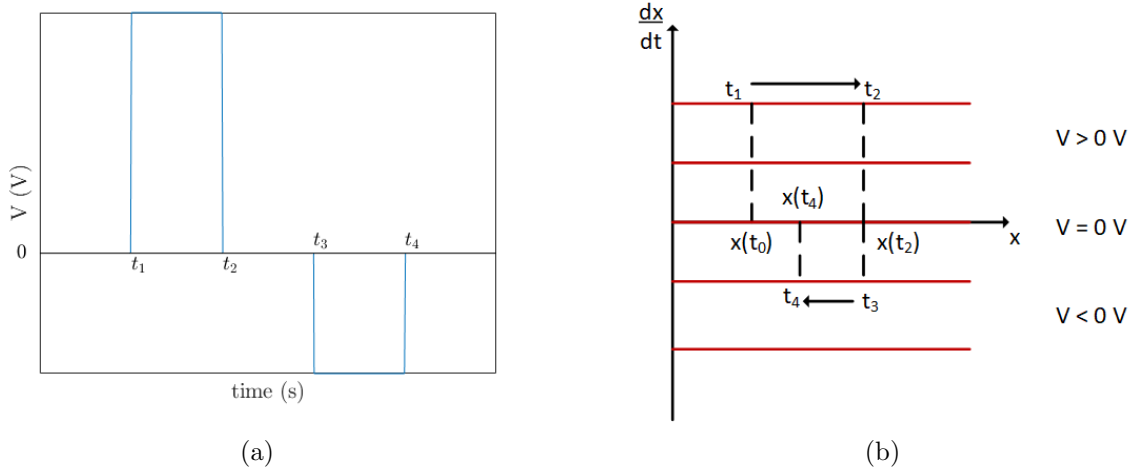


Figure 2.4: (a): Voltage signal applied. (b): Example of a DRM of a memrsitor

Two very important rules arise when studying the DRM of memristors, which were presented first in [6]. These rules are important because they can be used to determine the feasibility of the device in question. One of these rules is called the "two is infinite" theorem, which states that the sufficient condition for a passive memristor to be non-volatile over the entire  $x$ -axis, i.e, its POP is  $\frac{dx}{dt} = 0$ , is that it has two asymptotically stable equilibrium points. The proof of this theorem follows by considering the circuit shown in Fig.2.5(a), which is equivalent to an extended passive memristor by assuming that  $v_1$  is the state variable,  $i_1 = -\frac{dv_1}{dt}$  is the state equation and  $v_2$  and  $i_2$  are the voltage and current through the memristor. Then the  $(v_1, i_1)$  plot is the POP of the memristor when  $v_2 = 0V$ . Assuming that this POP has, at least, two asymptotically stable equilibrium points, it will always intersect the even quadrants, as shown in Fig.2.5(b), except if it coincides with the  $x$ -axis. Since the memristor is passive, its POP can not intersect the even quadrants, because then  $G_{11} = \frac{I_1}{V_1} \Big|_{V_2=0V} < 0S$ , making the device an active one. Therefore, it can be concluded that the POP has to coincide with the  $x$ -axis.

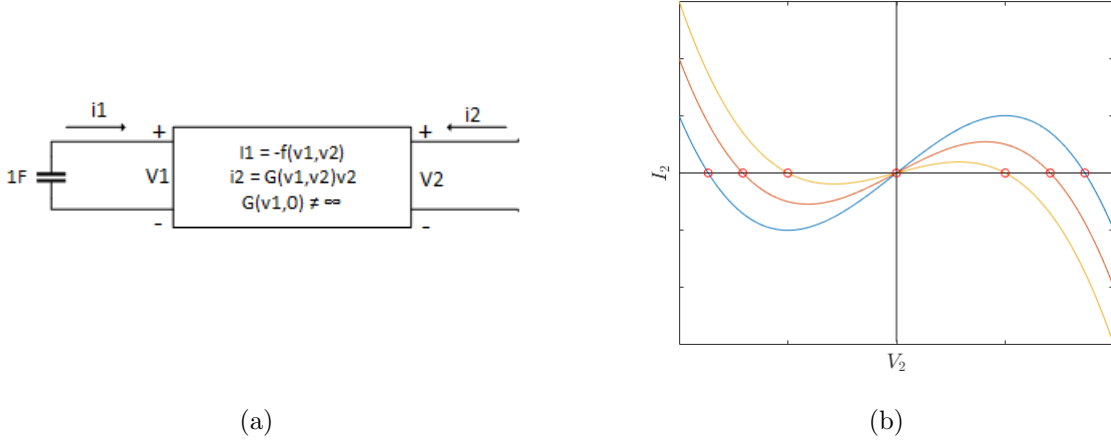


Figure 2.5: (a): Equivalent memristor circuit. (b): Examples of POPs with two asymptotically stable equilibrium points

The other rule is called the "no backtracking rule". This is an empirical rule which states that the dynamic routes are isolated in the upper plane for  $V > 0$  and lower plane for  $V < 0$  or vice-versa.

## 2.2 Memristor Models

In 2012, Chris Yakopcic et al. discussed LTSpice modeling of memristors in 2012 [7]. First by reviewing previously existing techniques and ending with the proposal of his own model. The present subsection summarizes these findings for the reader's convenience. Additionally, the model used at Knowm and the recently discovered  $\Phi$  memristor are also presented. It should be noted that some models are not shown here, because of their inherent specificity, reported in the literature.



### 2.2.1 Memristor Equivalent Circuit

Before reviewing any models, their implementation in the simulator will be discussed. For that, the equivalent memristor circuit shown in Fig.2.6 is used, where TE and BE are the top and bottom electrodes of the device, respectively.

$G(x, v)$  is the conductance of the device, and is modeled as a current source dependent on the voltage across the memristor. The state variable is calculated using a current source in parallel with a capacitor. Since  $i = C \frac{dv}{dt}$ , if  $I = \frac{dx}{dt}$  then  $CV$  is the implementation of the state variable  $x$ .

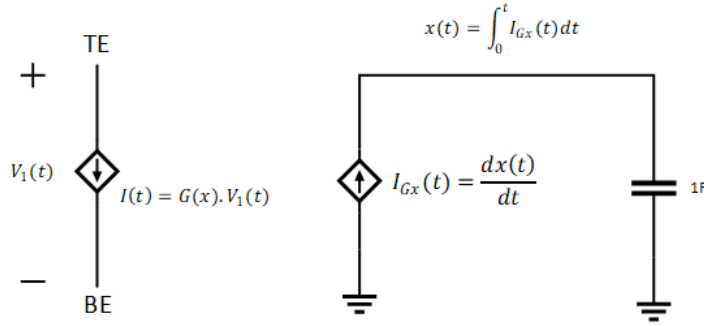


Figure 2.6: Memristor Equivalent Circuit

So, in order to implement a model into a simulator, such as LTSpice, one simply needs to define  $G(x)$  and  $\frac{dx(t)}{dt}$ , and create the equivalent circuit as shown in Fig.2.6. For more detail on the implementation of the models netlists on a simulation or on their actual simulation, refer to appendix A.

### 2.2.2 $\Phi$ memristor

In 2019, a paper was published describing the  $\Phi$  memristor [8], a memristor that directly relates magnetic flux with charge, making it an ideal memristor. This device is composed of a wire which passes through a magnetic core. The current passing through the wire generates a magnetic field, which induces a voltage across the conductor as it flows through the magnetic core. Eq.(2.8) describes the relationship between the magnetic flux and charge for this device, which was derived from the Landau–Lifshitz–Gilbert equation. In Eq.(2.8),  $\mu_0$  is the permeability of vacuum,  $S$  is the cross-sectional area,  $M_S$  is the saturation magnetization and  $m_0$  is the initial normalized magnetization.  $S_w = \frac{2\pi r(1+g^2)}{g|\gamma|}$  is called the switching coefficient, where  $\gamma$  is the electron gyromagnetic ratio,  $g$  is the Gilbert damping parameter and  $r$  is the radius of the device.

$$\phi = \mu_0 S M_S [\tanh(\frac{1}{S_w} q + \text{arctanh}(m_0)) - m_0] \quad (2.8)$$

$$\frac{d\phi}{dt} = \frac{\mu_0 S M_S}{S_w} \text{sech}^2(\frac{1}{S_w} q + \text{arctanh}(m_0)) \frac{dq}{dt} \quad (2.9)$$

By definition,  $\frac{d\phi}{dt} = V$  and  $\frac{dq}{dt} = I$ . Substituting in Eq.(2.9) yields:

$$V(t) = M(q)I(t) \quad (2.10)$$

$$M(q) = \frac{\mu_0 S M_S}{S_w} \text{sech}^2\left(\frac{1}{S_w}q + \text{arctanh}(m_0)\right) \quad (2.11)$$

A voltage controlled version of this model can be obtained by calculating the inverse function of Eq.(2.8) and its derivative. The inverse of Eq.(2.8) is represented in Eq.(2.12) and its derivative in Eq.(2.13), where  $G(\phi) = \frac{S_w}{\mu_0 S M_S} \frac{1}{1 - (\frac{1}{\mu_0 S M_S} \phi + m_0)^2}$

$$q = S_w(\text{arctanh}(\frac{1}{\mu_0 S M_S} \phi + m_0) - \text{arctanh}(m_0)) \quad (2.12)$$

$$I(t) = G(\phi)V(t) \quad (2.13)$$

However, since the domain of the arctanh function is  $]-1,1[$ , this model is only valid for  $|\frac{\phi}{\mu_0 S M_S} + m_0| < 1$ , while the current controlled version is valid across the entire real line.

The IV characteristics of this model can be seen in Figs.2.7 and 2.8.

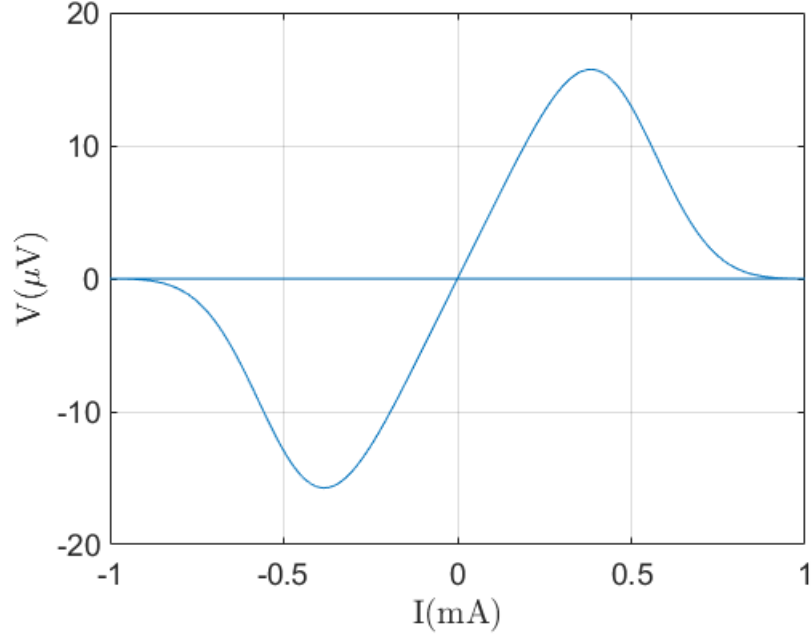


Figure 2.7: IV characteristic of a  $\Phi$  memristor when driven by  $I(t) = 0.001 \sin(2\pi t)$ . (Model Parameters:  $S = 1\mu m$ ,  $S_w = 0.3Oe$ ,  $M_s = 1\mu T$ ,  $m_0 = 0$ )

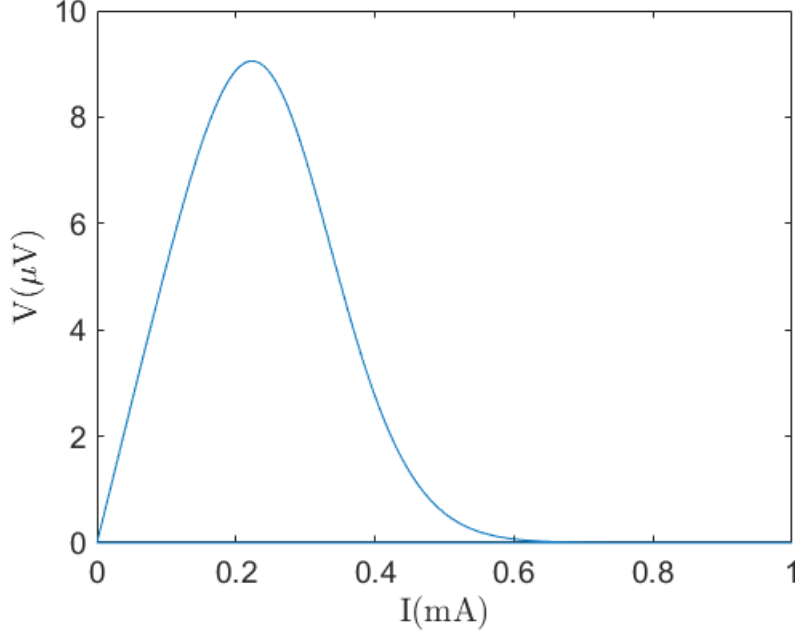


Figure 2.8: IV characteristic of a  $\Phi$  memristor when driven a current triangular wave ranging from 0 to 1mA and period of 1s. (Model Parameters:  $S = 1\mu m$ ,  $S_w = 0.3Oe$ ,  $M_s = 1\mu T$ ,  $m0 = 0$ )

### 2.2.3 HP Labs Model

In 2008, results published by HP Labs described the behavior of the memristor as it is presented in the following equations [1].  $V(t)$  represents the voltage across the memristor at time  $t$ ,  $I(t)$  the current through it,  $R_{off}$  and  $R_{on}$  are constants that represent the maximum and minimum resistances that the device may present, respectively,  $D$  is the device length and  $x(t)$  is the length of the oxygen deficient layer of the device and  $\mu_D$  is the oxygen ion mobility.  $\frac{dx(t)}{dt}$  is called the drift velocity of oxygen deficiencies in the device and defines the state equation of an ideal generic memristor.

$$V(t) = [R_{on} \frac{x(t)}{D} + R_{off}(1 - \frac{x(t)}{D})]I(t) \quad (2.14)$$

$$\frac{dx(t)}{dt} = \frac{\mu_D R_{on}}{D} I(t) \quad (2.15)$$

The results given by any input are calculated by integrating Eq.(2.15) and plugging  $x(t)$  in Eq.(2.14). It is noteworthy that, in Eq.(2.15), everything before  $I(t)$  is a constant. Therefore,  $x(t)$  is proportional to the integral of the current through the device, which means that, in the absence of current, the charge on the device is constant,  $x(t)$  is constant and the device's resistance is constant, which lets us conclude that the model describes a non-volatile device, as expected from HP Labs' implementation of the memristor.

It can be observed in Eqs.(2.14) and (2.15) that  $x(t)$  does not have any bounds, which means, according to the model, the length of the oxygen deficient layer could be larger than the actual device. It can be concluded then, that the above equations are not enough to accurately describe the memristor's behavior. So, a window function  $f(x) = \frac{x(D-x)}{D^2}$  for the state equation is also proposed, which models a nonlinearity in ionic transport as  $x$  approaches 0 or  $D$ . In the next two subsections some modifications to this model are presented.

### 2.2.3.1 Joglekar Window

The modification proposed by Joglekar and Wolf to the HP Labs memristor model [9] is presented in the following equations, where  $\eta$  is a parameter to model the direction of the motion of  $x(t)$  relative to the applied voltage. If a positive voltage applied to the device increases the state variable,  $\eta = 1$ . If it decreases the value of the state variable,  $\eta = -1$ . The parameter  $p$  models the magnitude of the windowing effect of  $F(x(t))$ , as  $p$  becomes larger, the windowing effect becomes more pronounced. The graph of this window function can be seen in Fig.2.9 and the IV characteristics of this model when driven by a sine wave and an offset triangular wave are presented in Figs.2.10 and 2.11, respectively.

$$\frac{dx(t)}{dt} = \eta \frac{\mu_D R_{on}}{D^2} F(x(t)) I(t) \quad (2.16)$$

$$F(x(t)) = 1 - (2x(t) - 1)^{2p} \quad (2.17)$$

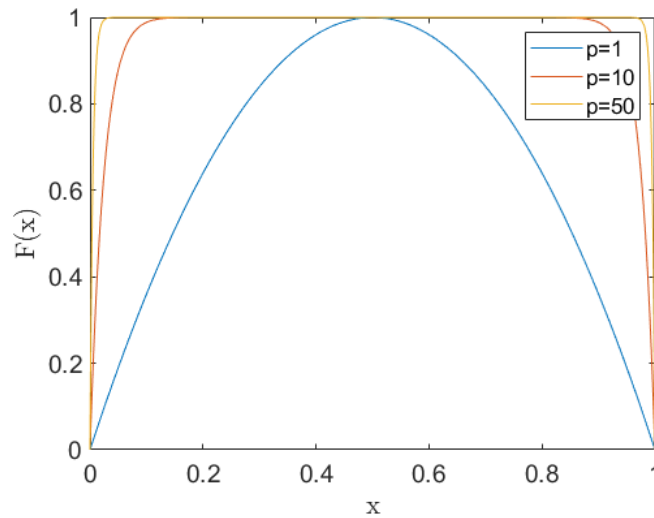


Figure 2.9: Graph of the Joglekar Window Function

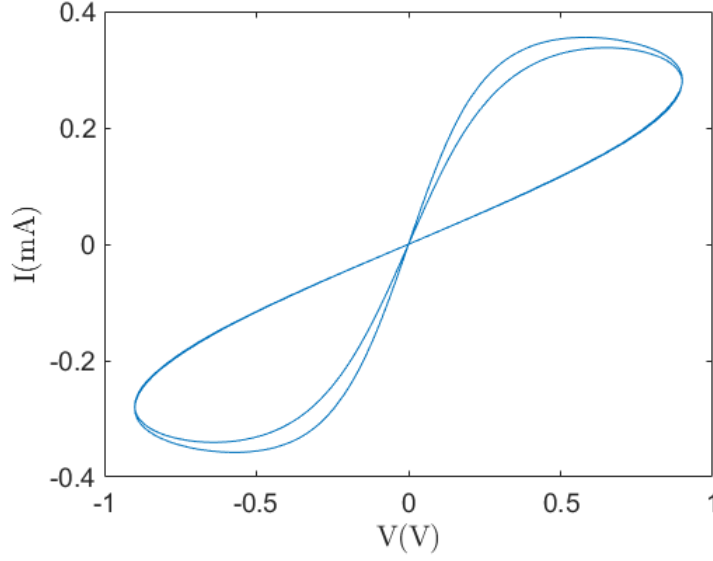


Figure 2.10: IV characteristic of the HP memristor model using Joglekar Window when driven by  $V(t) = 0.9\sin(20\pi t)$  (Model Parameters:  $R_{on} = 100\Omega$ ,  $R_{off} = 10k\Omega$ ,  $D = 12nm$ ,  $\mu_D = 50 \times 10^{-15}m^2s^{-1}V^{-1}$ ,  $p = 7$ ,  $\eta = 1$ )

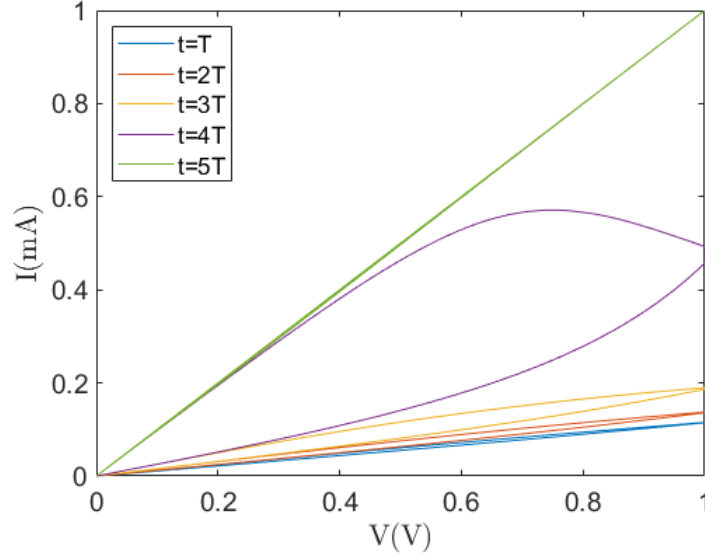


Figure 2.11: IV characteristic of the HP memristor model using Joglekar Window when driven by a triangular wave ranging from 0 to 1V with period of 1s. (Model Parameters:  $R_{on} = 1k\Omega$ ,  $R_{off} = 10k\Omega$ ,  $\mu_D = 20 \times 10^{-15}m^2s^{-1}V^{-1}$ ,  $D = 85nm$ ,  $p = 2$ ,  $\eta = 1$ )

The new IV relationship is the same as Eq.(2.14), except that  $x(t)$  is now a normalized quantity between 0 and 1. When it is 0, the device's resistance is  $R_{off}$ , and when it is 1, the resistance is  $R_{on}$ . This also allows for a generalization on the structure of the device, meaning

that this model represents more devices than just the titanium dioxide device developed by HP Labs.

However, this model produces a problem: the motion of the state variable is reduced near the boundaries whether it is traveling toward or away from it, while the data presented by HP Labs shows that the motion of the state variable reduces only when traveling towards the boundary. The windowing function presented next solves this problem.

### 2.2.3.2 Biolek Window

The next windowing function, which was proposed by Biolek et al. [10], aims at solving the problem brought by the Joglekar window, which is the reduction of the motion of the state variable being independent of the direction it is traversing towards relative to the boundaries. This function is shown in Eq.(2.18), where  $u(t)$  is the Heaviside step function, which has value 0 if  $t < 0$  and 1 otherwise. The graph of this function is shown in Fig.2.12, and the new IV characteristics can be seen in Figs.2.13 and 2.14.

$$F(x(t)) = 1 - (x(t) - u(-I(t)))^{2p} \quad (2.18)$$

The state equation and IV relationship are the same as the Joglekar model. If  $I(t) > 0$ ,  $u(-I(t)) = 0$  and  $\frac{dx(t)}{dt} > 0$ , which means that  $x(t)$  increases with time and  $F(x(t))$  decreases until  $x(t) = 1$ , at which point  $F(x(t)) = 0$  and, therefore,  $\frac{dx(t)}{dt} = 0$ , imposing the boundary effect on  $x(t) = 1$ . On the other hand, if  $I(t) < 0$ ,  $u(-I(t)) = 1$  and  $\frac{dx(t)}{dt} < 0$ , making  $x(t)$  decrease with time and  $F(x(t))$  decreases until  $x(t) = 0$  and  $F(x(t)) = 0$ , bounding the state variable at  $x(t) = 0$ .

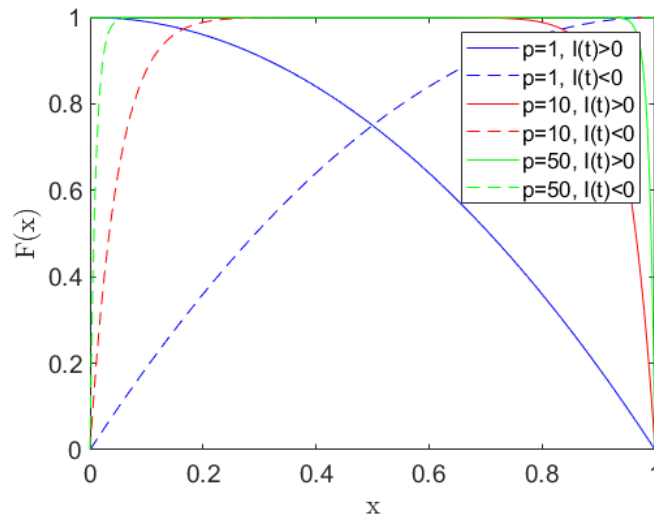


Figure 2.12: Graph of the Biolek Window Function

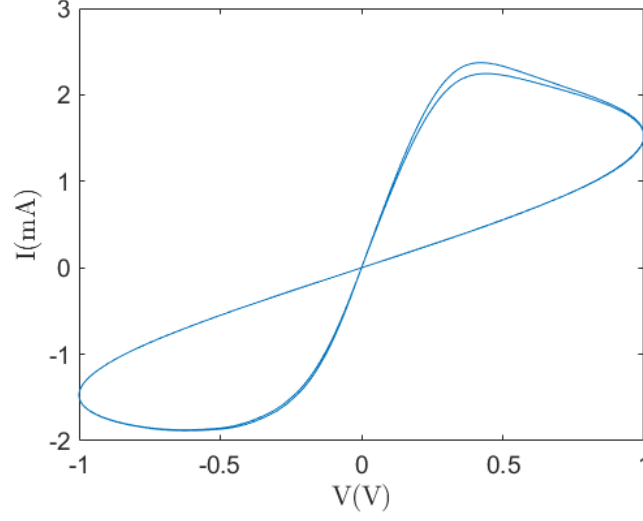


Figure 2.13: IV characteristic of the HP memristor model using the Biolek Window when driven by a  $\sin V(t) = \sin(20\pi t)$  (Model Parameters:  $R_{on} = 100\Omega$ ,  $R_{ff} = 1k\Omega$ ,  $D = 10nm$ ,  $\mu_D = 40fm^2s^{-1}V^{-1}$ ,  $p = 7$ )

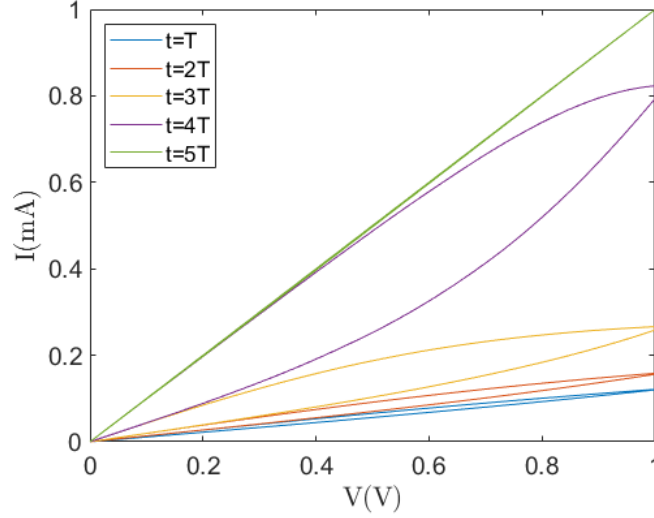


Figure 2.14: IV characteristic of the HP memristor model using the Biolek Window when driven by a triangular wave ranging from 0 to 1V with period of 1s. (Model Parameters:  $R_{on} = 1k\Omega$ ,  $R_{off} = 10k\Omega$ ,  $\mu_D = 20 \times 10^{-15}m^2s^{-1}V^{-1}$ ,  $D = 80nm$ ,  $p = 2$ )

The models previously presented show inconsistencies in comparison with actual physical data. When driving the device with a non-zero mean periodic pulse, the IV curve presents several loops of increasing area, but the published data presents the opposing effect [11, 12]. Another problem is that actual memristors present threshold voltages for which the conductivity changes abruptly, but the models presented thus far do not include this effect. Finally, it was also observed that the motion of the state variable is not equivalent when

moving in either the positive or negative direction, but the models presented thus far do not incorporate this effect.

## 2.2.4 Hyperbolic Sine Models

The next three models presented are called hyperbolic sine models, because they are largely based on the hyperbolic sine function, which has been observed to model MIM (metal-insulator-metal) junctions very well. Since memristors are usually made by depositing an oxide, such as titanium dioxide, between two metal electrodes, forming a MIM junction, it is reasonable to approximate a memristor's behavior through a hyperbolic sine. These models return considerably better results when driving the device with a non-zero mean pulsed signal.

### 2.2.4.1 General Hyperbolic Sine Model

The memristor model described by the following two equations was developed by Mika Laiho et al. [13], in which,  $a_1$ ,  $a_2$ ,  $b_1$  and  $b_2$  are parameters used to shape the IV response of the device,  $d_1$  and  $d_2$  are parameters used to adjust the threshold voltages of the device and  $c_1$  and  $c_2$  are used to define the intensity of the state variable motion. Lastly,  $F(x(t))$  is the Biolek Window function. The graphs of this model's IV characteristics, when driven by a sine wave and a non-zero mean triangular wave can be seen in Figs. 2.15 and 2.16.

$$I(t) = \begin{cases} a_1 x(t) \sinh(b_1 V(t)) & , V(t) \geq 0 \\ a_2 x(t) \sinh(b_2 V(t)) & , V(t) < 0 \end{cases} \quad (2.19)$$

$$\frac{dx(t)}{dt} = \begin{cases} c_1 \sinh(d_1 V(t)) F(x(t)) & , V(t) \geq 0 \\ c_2 \sinh(d_2 V(t)) F(x(t)) & , V(t) < 0 \end{cases} \quad (2.20)$$

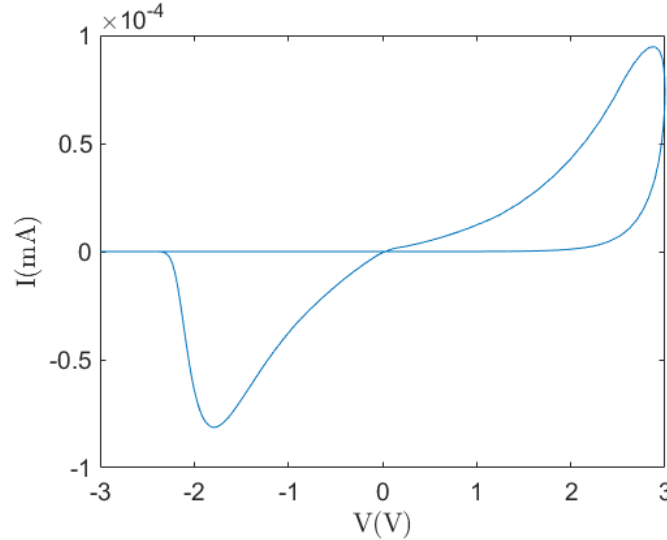


Figure 2.15: IV characteristic of the General Sinh Model when driven by  $V(t) = 3\sin(0.2\pi t)$  (Model Parameters:  $a_1 = 4 \times 10^{-8}$ ,  $a_2 = 1.25 \times 10^{-7}$ ,  $b_1 = 1.2$ ,  $b_2 = 1.2$ ,  $c_1 = 6 \times 10^{-4}$ ,  $c_2 = 6.60 \times 10^{-4}$ ,  $d_1 = 2$ ,  $d_2 = 3.8$ ,  $p = 1$ )



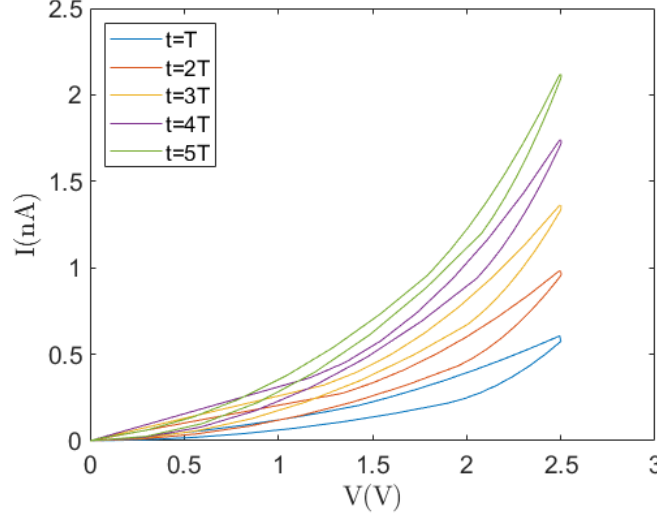


Figure 2.16: IV characteristic of the General Sinh Model when driven by a triangular wave ranging from 0 to 2.5V with period of 100ms (Model Parameters:  $a_1 = 4 \times 10^{-8}$ ,  $b_1 = 1.2$ ,  $a_2 = 1.25 \times 10^{-7}$ ,  $b_2 = 1.2$ ,  $c_1 = 6 \times 10^{-4}$ ,  $d_1 = 2$ ,  $c_2 = 6.6 \times 10^{-4}$ ,  $d_2 = 3.8$ ,  $p = 1$ )

#### 2.2.4.2 University of Michigan Model

The following alternative model was proposed by Ting Chang et al [14]. As it can be seen in Eq.(2.21), the current is made up of two terms: the first term is a current produced by a Schottky barrier due to the oxide layer and the bottom electrode. The second term is due to tunneling through the MIM junction. In this model the state variable  $x(t)$  is no longer the length of the doped region, but is the area through which ions pass through tunneling effect in the MIM junction. When  $x(t) = 0$ , the devices conductivity is due completely to the Schottky barrier, and when  $x(t) = 1$ , it is dominated by the tunneling effect through the MIM junction. In this model,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ ,  $\eta_1$  and  $\eta_2$  are positive valued parameters that describe material properties, and can be read as fitting parameters.  $\lambda$  represents the magnitude of the state variable motion. The state equation Eq.(2.22) also has two terms, where the second term was added later to include the overlapping of several hysteresis loops, when the device was driven with a periodic pulse. This effect was said to be due to ion diffusion in the device. The IV characteristics of this model can be seen in Figs.2.17 and 2.18. For the purpose of this simulation the window function shown in Eq.(2.23) was used to bound  $x$  between 0 and 1.

$$I(t) = (1 - x(t))\alpha(1 - e^{\beta V(t)}) + x(t)\gamma \sinh(\delta V(t)) \quad (2.21)$$

$$\frac{dx(t)}{dt} = \lambda[\eta_1 \sinh(\eta_2 V(t)) - \frac{x(t)}{\tau}] \quad (2.22)$$

$$f(x, V) = \frac{1 + \text{sign}(V)}{2} \frac{1 + \text{sign}(1 - x)}{2} + \frac{1 + \text{sign}(-V)}{2} \frac{1 + \text{sign}(x)}{2} \quad (2.23)$$

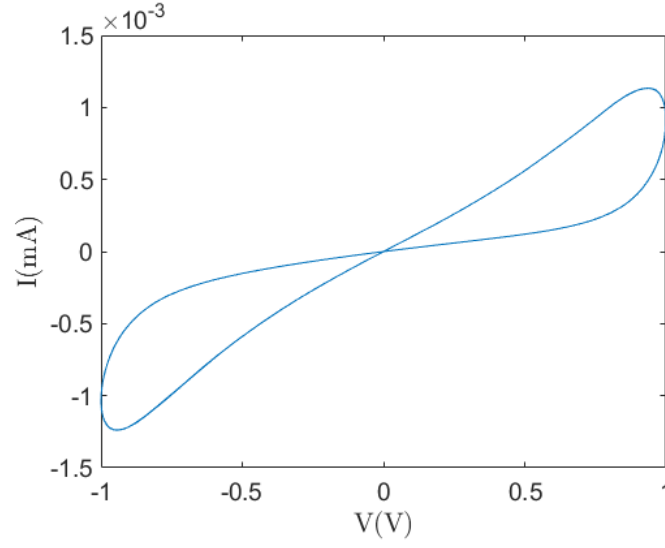


Figure 2.17: IV characteristic of the U. Michigan memristor model when driven by  $V(t) = \sin(2\pi t)$  (Model Parameters:  $\alpha = 0.5 \times 10^{-6}$ ,  $\beta = 0.5$ ,  $\gamma = 4 \times 10^{-6}$ ,  $\delta = 2$ ,  $\lambda = 4.5$ ,  $\eta_1 = 4 \times 10^{-3}$ ,  $\eta_2 = 4$ ,  $\tau = 10$ )

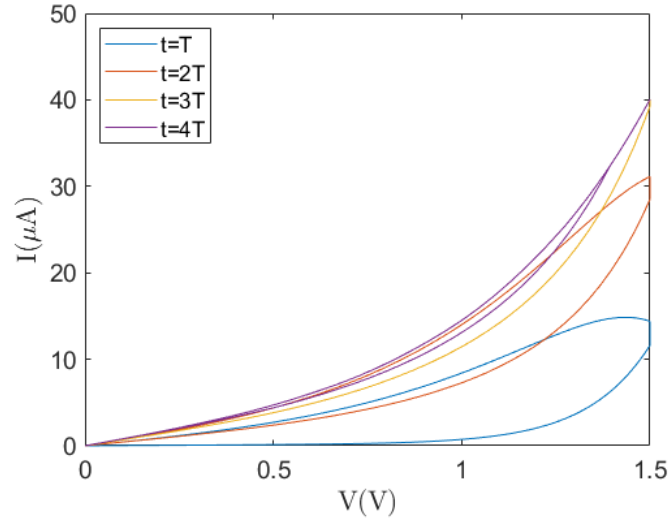


Figure 2.18: IV characteristic of the U. Michigan memristor model when driven by a triangular wave ranging from 0 to 1.5V with period of 1s. (Model Parameters:  $\alpha = 0.5 \times 10^{-6}$ ,  $\beta = 0.5$ ,  $\gamma = 4 \times 10^{-6}$ ,  $\delta = 2$ ,  $\lambda = 4.5$ ,  $\eta_1 = 4 \times 10^{-3}$ ,  $\eta_2 = 4$ ,  $\tau = 10$ )

The two previous models model a memristors characteristics well for various signal inputs, by using the sinh function. Also, additional properties are modeled, which translates into strong correlation with measured data from many physical devices.

### 2.2.4.3 Yakopcic's Model

Like in the previous models, this model makes use of the sinh function to shape the IV relation of the device [15]. However, unlike in the general sinh model, a single  $b$  parameter is used, which controls the intensity of the threshold relating conductivity to voltage amplitude and is independent of voltage polarity, but two  $a$  parameters are still needed to be consistent with measured data, which shows that memristors are more conductive in the positive region.

$$I(t) = \begin{cases} a_1 x(t) \sinh(bV(t)) & , V(t) \geq 0 \\ a_2 x(t) \sinh(bV(t)) & , V(t) < 0 \end{cases} \quad (2.24)$$

The state equation Eq.(2.25) is the product of two functions, namely  $g(V(t))$  and  $f(x(t))$ , which are shown in Eqs.(2.26) and (2.27), respectively. The  $\eta$  variable is simply the direction of the state variable motion and it behaves just like in Eq.(2.16).

$$\frac{dx(t)}{dt} = \eta g(V(t)) f(x(t)) \quad (2.25)$$

$g(V(t))$  allows the programming of the threshold of the device. This threshold is the minimum energy required to alter the physical structure of the device, as it is shown in several publications [11, 12, 16]. In this function, the parameters  $V_p$  and  $V_n$  are the positive and negative thresholds, respectively, and  $A_p$  and  $A_n$  are the speeds with which the state variable evolves after overcoming the positive and negative thresholds, respectively.

$$g(V(t)) = \begin{cases} A_p(e^{V(t)} - e^{V_p}) & , V(t) > V_p \\ -A_n(e^{-V(t)} - e^{V_n}) & , V(t) < -V_n \\ 0 & , -V_n \leq V(t) \leq V_p \end{cases} \quad (2.26)$$

$f(x(t))$  is a windowing function that represents the increasing difficulty in changing the state variable as it approaches the boundaries. It also represents how the state variable motion is different depending on the polarity of the input voltage polarity. It can be seen that up until  $x_p$  or  $x_n$  the function is constant, and then it decays at the rate of  $\alpha_p$  or  $\alpha_n$ . This function is also defined by the functions  $w_p(x(t), x_p)$  and  $w_n(x(t), x_n)$ , which ensure that  $f(x(t))$  is 0 when  $x(t) = 1$  and  $x(t) = 0$ , respectively.

$$f(x(t)) = \begin{cases} \begin{cases} e^{-\alpha_p(x(t)-x_p)} w_p(x(t), x_p) & , x(t) \geq x_p \\ 1 & , x(t) < x_p \end{cases} & , \eta V(t) > 0 \\ \begin{cases} e^{\alpha_n(x(t)+x_n-1)} w_n(x(t), x_n) & , x(t) \leq 1-x_n \\ 1 & , x(t) > 1-x_n \end{cases} & , \eta V(t) \leq 0 \end{cases} \quad (2.27)$$

$$w_p(x, x_p) = \frac{x_p - x}{1 - x_p} + 1 \quad (2.28)$$

$$w_n(x, x_n) = \frac{x}{1 - x_n} \quad (2.29)$$

The IV characteristics resulting from this model for a voltage consisting of a sine wave and an offset triangular wave can be seen in Figs.2.19 and 2.20.

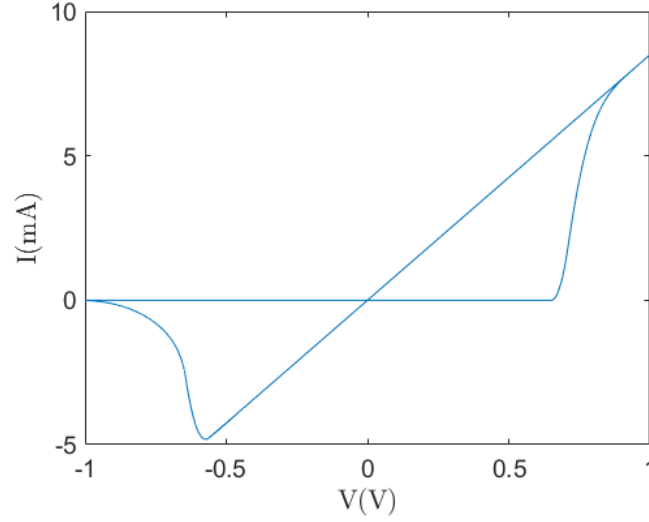


Figure 2.19: IV characteristic of the Yakopcic model when driven by  $V(t) = \sin(20\pi t)$  (Model Parameters:  $a_1 = 0.17$ ,  $a_2 = 0.17$ ,  $b = 0.05$ ,  $V_p = 0.65V$ ,  $V_n = 0.56V$ ,  $A_p = 4000$ ,  $A_n = 4000$ ,  $x_p = 0.3$ ,  $x_n = 0.5$ ,  $\alpha_p = 1$ ,  $\alpha_n = 5$ ,  $x_0 = 0$ ,  $\eta = 1$ )

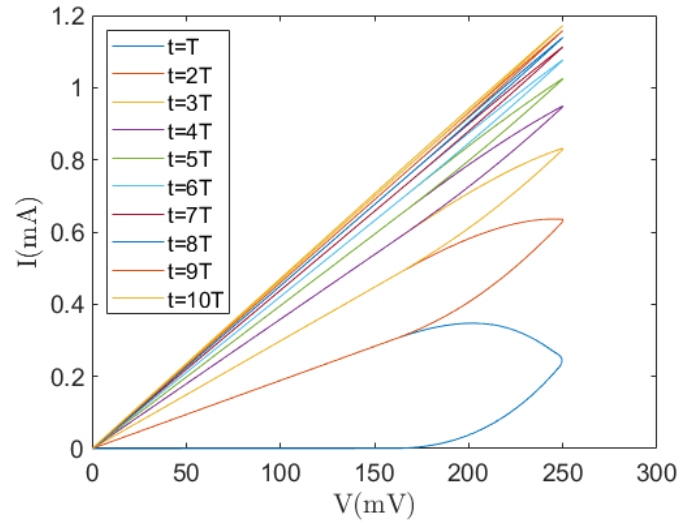


Figure 2.20: IV characteristic of the Yakopcic model when driven by a triangular wave ranging from 0 to 250 mV with period of 100ms (Model Parameters:  $a_1 = 0.097$ ,  $a_2 = 0.097$ ,  $b = 0.05$ ,  $V_p = 0.16$ ,  $V_n = 0.15$ ,  $A_p = 4000$ ,  $A_n = 4000$ ,  $x_p = 0.3$ ,  $x_n = 0.5$ ,  $\alpha_p = 1$ ,  $\alpha_n = 5$ ,  $\eta = 1$ )

While the previous model correlates to physical devices, it was observed at Knowm that memristors exhibit some stochastic behavior and, up until now, all models presented are deterministic. The semi-empirical model that follows was shown to model a wide range of

devices and considers the stochastic behavior of the memristor.

### 2.2.5 Metastable Switch Model

In this model, proposed in [17] and [18], it is assumed that the memristor is composed of metastable switches (MSS) which vary over time. A MSS is defined as an ideal device of two states and changes states with a probability which is dependent in applied voltage and temperature. The current across the memristor is described by the following equation, where  $I_s$  is a current due to a Schottky barrier formed by a metal-oxide junction,  $I_m$  is a current dependent on the MSS and  $\Phi$  quantifies the impact of each term.

$$I = \Phi I_m(V, t) + (1 - \Phi) I_s(V) \quad (2.30)$$

The current  $I_s$  is given by the equation below, in which,  $\alpha_f$ ,  $\alpha_r$ ,  $\beta_f$  and  $\beta_r$  are positive valued parameters.

$$I_s = \alpha_f e^{\beta_f V} - \alpha_r e^{-\beta_r V} \quad (2.31)$$

$I_m$  represents the idea that a memristor is a set of conducting channels that switch between different states of different resistances. This change of resistance is obtained through an external voltage. Each of these channels is a MSS.

A MSS has two states A and B separated by a potential barrier, and it switches from state B to A with probability  $P_A$  and from A to B with probability  $P_B$ , which are defined as:

$$P_A = \alpha \frac{1}{1 + e^{-\beta(V-V_A)}} = \alpha \Gamma(V, V_A) \quad (2.32)$$

$$P_B = \alpha(1 - \Gamma(V, V_B)) \quad (2.33)$$

where  $\beta = \frac{q}{k_B T} = v_t^{-1}$  is the inverse of the thermal voltage and  $\alpha = \frac{\Delta t}{\tau}$  is the ratio between the time step period and the device's time constant. From Eqs. (2.32) and (2.33), it can be seen that a positive voltage increases  $P_A$  and decreases  $P_B$  and vice-versa. It should be noted that the logistic function  $\frac{1}{1+e^{-x}}$  is similar to the sinh that was used in previous models.

The memristor is modeled as a set of  $N$  MSS that evolve in discrete time  $\Delta t$ . The total conductance is the sum of each MSS, where  $N_B$  and  $N_A$  are the number of MSS in state B and A, respectively.

$$G_m = N_A G_A + N_B G_B = N_B (G_B - G_A) + N G_A \quad (2.34)$$

The probability that out of  $n$  MSS,  $k$  will change state follows a binomial distribution, where  $p$  is the state transition probability  $P_A$  or  $P_B$ :

$$P(n, k) = \binom{n}{k} p^k (1 - p)^{n-k} \quad (2.35)$$

And, as  $n$  becomes larger,  $P(n,k)$  can be approximated by the normal distribution, where  $\mu = np$  and  $\sigma^2 = np(1-p)$ .

$$\lim_{n \rightarrow \infty} P(n, k) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} = \mathcal{N}(\mu, \sigma^2) \quad (2.36)$$

The change in conductance of the memristor is, therefore, modeled as a probabilistic process where the number of MSS that change state is picked out of a normal distribution with average  $np$  and variance  $np(1-p)$ . The update of the conductance is the sum of two random variables:

$$\Delta N_B = \mathcal{N}(N_A P_A, N_A P_A(1-P_A)) - \mathcal{N}(N_B P_B, N_B P_B(1-P_B)) \quad (2.37)$$

$$\Delta G_m = \Delta N_B (G_B - G_A) \quad (2.38)$$

To make this model have the conventional form and be usable in a simulator, the state variable must be defined as  $x$  in the interval  $[0,1]$  and not  $N_B$ , and it must be defined in terms of  $\frac{dx(t)}{dt}$ .

The change in the number of MSS,  $dx$ , is:

$$dx = N_{off \rightarrow on} - N_{on \rightarrow off} \quad (2.39)$$

The number of switches switching state is:

$$N_{off \rightarrow on} = P_{off \rightarrow on}(1-x) \quad (2.40)$$

$$N_{on \rightarrow off} = P_{on \rightarrow off}x \quad (2.41)$$

Substituting Eqs. (2.40) and (2.41) in Eq. (2.39):

$$dx = P_{off \rightarrow on}(1-x) - P_{on \rightarrow off}x \quad (2.42)$$

And finally, substituting Eqs. (2.32) and (2.33) in (2.42):

$$dx = \frac{dt}{\tau} \frac{1}{1 + e^{-\beta(V-V_{on})}}(1-x) - \frac{dt}{\tau} \left(1 - \frac{1}{1 + e^{-\beta(V+V_{off})}}\right)x \quad (2.43)$$

$$\frac{dx}{dt} = \frac{1}{\tau} \left[ \frac{1}{1 + e^{-\beta(V-V_{on})}}(1-x) - \left(1 - \frac{1}{1 + e^{-\beta(V+V_{off})}}\right)x \right] \quad (2.44)$$

Eq. (2.44) is now in the conventional form of a memristor state equation. By defining the conductance of the device as:

$$G = \frac{x}{R_{on}} + \frac{1-x}{R_{off}} \quad (2.45)$$

And by applying Ohm's law:

$$I = GV \quad (2.46)$$

The model is complete, and its IV characteristics can be observed in Figs.2.21 and 2.22.

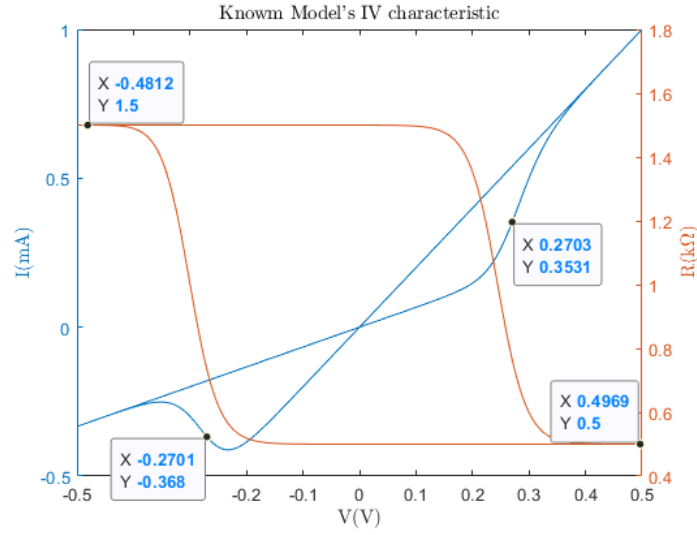


Figure 2.21: IV characteristic of the Known memristor model (blue) and Resistance vs. Voltage curve (orange) when driven by  $V(t) = 0.5\sin(200\pi t)$  (Model Parameters:  $R_{on} = 500\Omega$ ,  $R_{off} = 1.5k\Omega$ ,  $V_{on} = 0.27V$ ,  $V_{off} = 0.27V$ ,  $\tau = 100\mu s$ ,  $T = 298.5K$ )

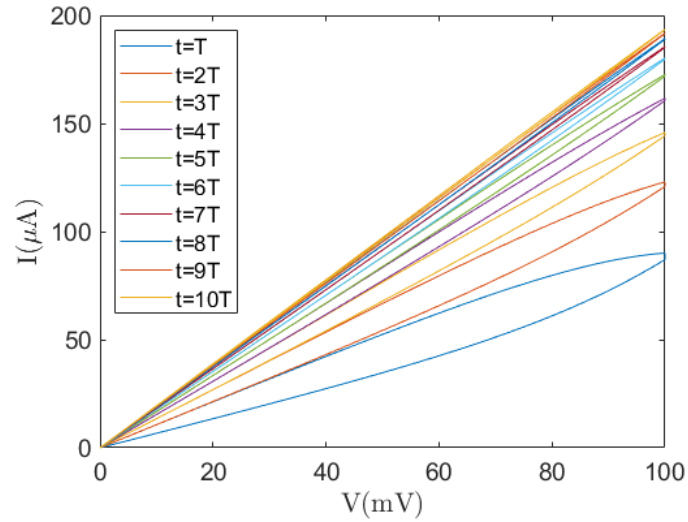


Figure 2.22: IV characteristic of the Known memristor model when driven by a triangular wave ranging from 0 to 100mV (Model Parameters:  $R_{on} = 500\Omega$ ,  $R_{off} = 1.5k\Omega$ ,  $V_{on} = 0.27V$ ,  $V_{off} = 0.27V$ ,  $\tau = 100\mu s$ ,  $T = 298.5K$ )

## 2.2.6 Model Classification

As discussed in subsection 2.1.2, it is possible to divide all mathematical models previously described into these four classes, as represented by Tab.2.2.

Table 2.2: Classification of the presented memristor models

Model	Class
$\Phi$ memristor	Ideal
HP Labs	Ideal Generic
Joglekar	Ideal Generic
Biolek	Ideal Generic
General Sinh Model	Extended
U.Mich	Extended
Yakopcic	Generic
Metastable Switch	Generic

In conclusion, given all the possibilities for memristor models, the model which will be used for simulation throughout the rest of this dissertation will be the MSS model, which is fairly simple compared to some others, especially when shaping the IV relationship, and also takes into account the stochastic behavior of the memristor.

## 2.3 Memristor Model Implementation in Cadence

This section discusses the implementation of a memristor model in Cadence. The model chosen was the Metastable Switch Model, which was described in subsection 2.2.5.

### 2.3.1 Model Implementation

Following the instructions in [19], a new library was created where the model is kept. Inside this library a new cell named “MEM\_KNOWM” was created.

Then, a cellview named “schematic” of the Pspice type was created and edited with the code shown bellow. After the editing, a symbol was created for the new component.

```
* "pspice" description for "memristive", "MEM_KNOWM", "schematic"

.SUBCKT MEM_KNOWM TE BE XSV

.PARAM Ron = 440 Roff = 8.2k Voff = 265.00m Von = 300.00m
+TAU = 1m x0 = 0 T = 273.15+27 boltz=1.38064852*(10** 23 )
echarge = 1.60217662*(10**-19)

* Function G(V(t)) - Describes the device threshold

.func G(V) = V/Ron+(1-V)/Roff

* Function F(V(t),x(t)) - Describes the SV motion
.func F(V1,V2) = (1/TAU)*((1/(1+exp(-1/(T*boltz/(echarge)))*(V1-Von))))*(1-V2)-
```



```

+ (1 - (1 / (1 + exp(-1 / (T * boltz / (e * charge)) * (V1 + Voff)))))) * V2)

* Memristor I-V Relationship
.func IVRel(V1,V2) = V1 * G(V2)

* Circuit to determine state variable
* dx/dt = F(V(t), x(t)) * G(V(t))
Cx XSV 0 1
.ic V(XSV) = x0
Gx 0 XSV CUR=F(V(TE,BE), V(XSV,0))

* Current source for memristor IV response
Gmem TE BE CUR=IVRel(V(TE,BE), V(XSV,0))

.ENDS MEM_KNOWM

```

As can be seen in the code above, the sub-circuit implemented has three terminals: TE and BE are the Top and Bottom Electrodes, respectively. The XSV terminal is a terminal that outputs the state variable as the device is operated. This terminal has no influence in the rest of the circuit, and should not be connected anywhere. Its sole purpose is to plot the state variable, which will be useful to measure certain metrics.

For testing the newly created component, a simple simulation was made in order to obtain its IV curve. The resulting IV plot is presented in Fig.2.23. The test conditions applied were the same as in subsection 2.2.5 for comparison.

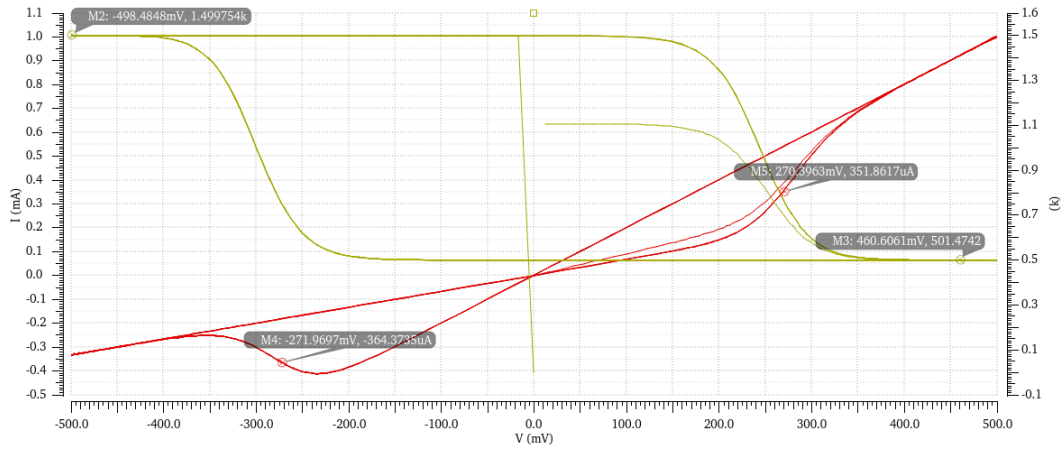


Figure 2.23: Current vs. Voltage curve (red) and Resistance vs. Voltage curve (yellow) with  $V(t) = 0.5\sin(200\pi t)$  (Model Parameters:  $R_{on} = 500\Omega$ ,  $R_{off} = 1.5k\Omega$ ,  $V_{on} = 0.27V$ ,  $V_{off} = 0.27V$ ,  $\tau = 100\mu s$ ,  $T = 298.5K$ )

The simulation results presented above show the expected model behavior when compared to Fig.2.21, meaning that the sub-circuit created in Cadence is fully operational.

## 2.3.2 Model Fitting

In order to approximate the model implemented in Cadence to that of a physical memristor, the device was experimentally measured and compared with the simulation results in Cadence.

The test setup is depicted in Fig.2.24, where the memristor used was the BS-AF-W from Knowm. The measurements were made using a PicoScope 2208B, with the AWG channel as the AC voltage source and the oscilloscope channels as indicated in the figure.

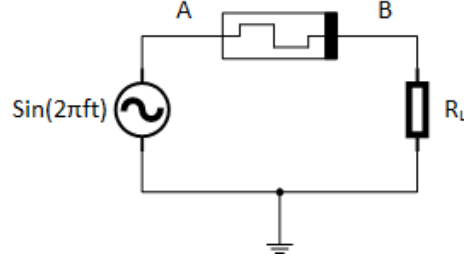


Figure 2.24: Circuit used to measure a physical memristor's IV-characteristic

The measurements consisted in 32 cycles of the signal, with 10000 samples each. This was made for frequencies from 1 Hz to 1 MHz, with increments of 1 decade and for load resistances of 1 k $\Omega$ , 10k $\Omega$  and 100k $\Omega$ . After obtaining the data for each frequency and load the average of the 32 cycles was calculated. The signal processing consisted of a moving mean filter applied to that average to smooth the signal. Below are shown some of the results obtained which are representative of the evolution of the dynamic of the memristor. The rest of the results obtained can be consulted in appendix B and the code used can be consulted in appendix C.

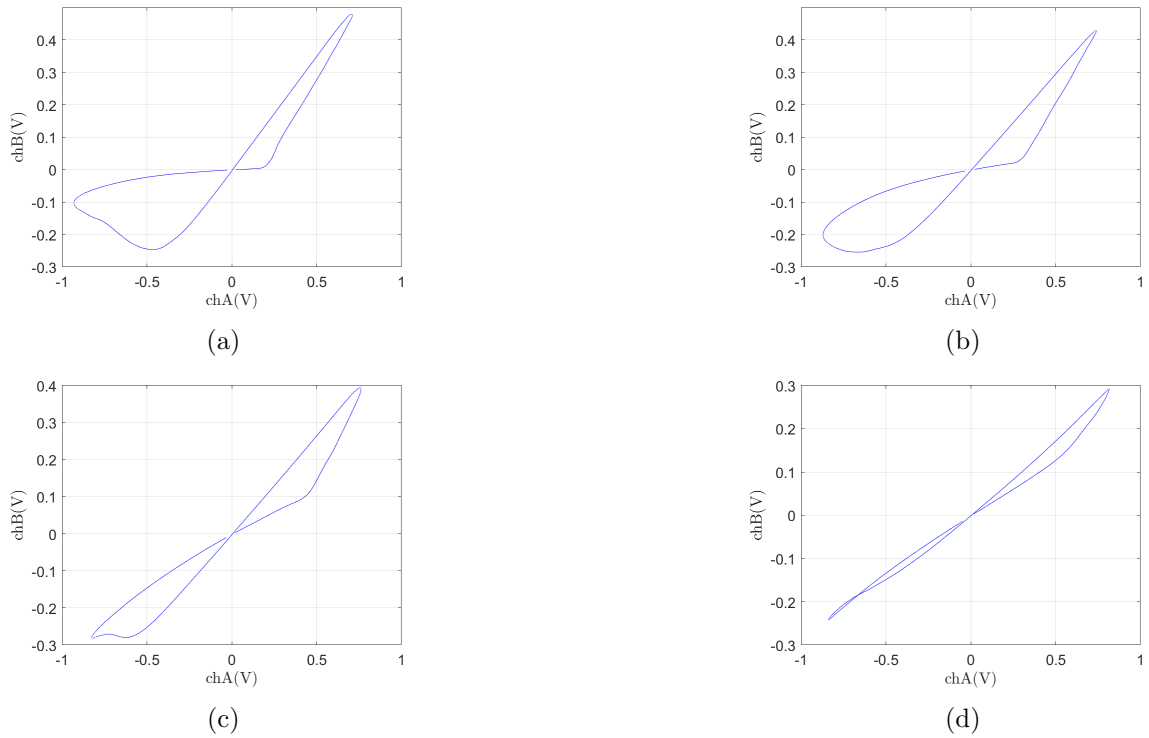
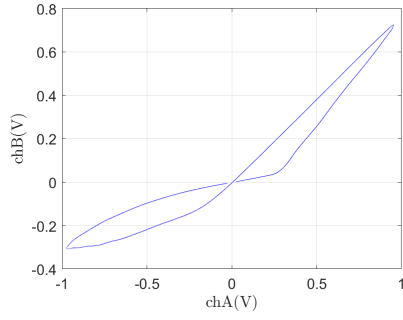
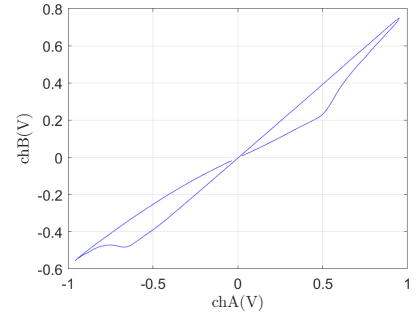


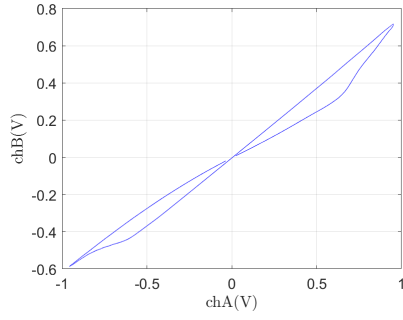
Figure 2.25: IV-curves measured with  $R_L = 1k\Omega$ : (a) 1Hz, (b) 10Hz, (c) 100Hz, (d) 1kHz



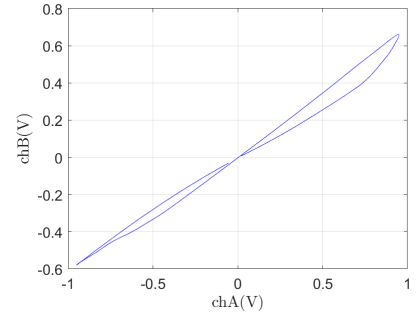
(a)



(b)

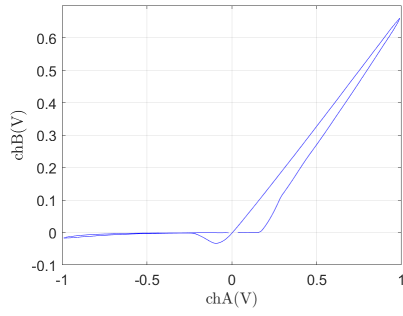


(c)

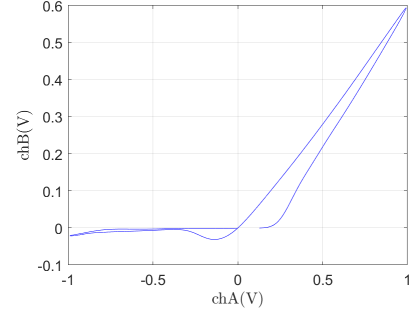


(d)

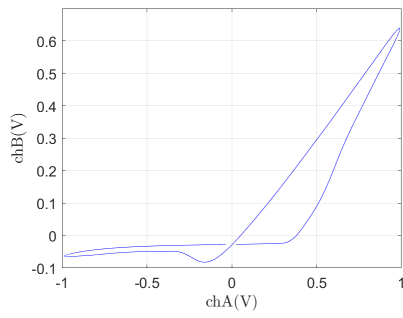
Figure 2.26: IV-curves measured with  $R_L = 10k\Omega$  : (a) 1Hz, (b) 10Hz, (c) 100Hz, (d) 1kHz



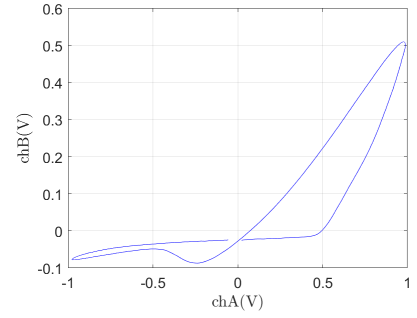
(a)



(b)



(c)



(d)

Figure 2.27: IV-curves measured with  $R_L = 10k\Omega$  : (a) 1Hz, (b) 10Hz, (c) 100Hz, (d) 1kHz

It should be apparent that the IV-characteristic of the memristor tends to a straight line as the frequency increases. This is a known property of memristors which can be consulted more in-depth in [4]. It should also be apparent from the measurements taken with lower frequencies that the values of the high and low state resistances vary with the load resistance. This effect was also reported by the manufacturer in the device's datasheet. However it is not considered in the model proposed by Knowm. So, as a simplification, the model will be tuned to fit the 1 k $\Omega$  case, and the other cases will be neglected, ensuring all switching of memristors will be done with a 1 k $\Omega$  series resistor, when required.

The tuning was done following the procedure shown below:

1. Calculate the high and low states' resistances of the memristor. The slope at a given point of the measured IV curve is given by Eq.(2.47).

$$m = \frac{V_B}{V_A} \quad (2.47)$$

The voltage at B is given by the voltage divider formed by the memristor  $M$  and the load  $R_L$ :

$$V_B = \frac{R_L}{R_L + M} \quad (2.48)$$

Substituting Eq.(2.48) into Eq.(2.47) and solving for  $M$  results in Eq.(2.49).

$$M = \frac{R_L}{m} - R_L \quad (2.49)$$

Since a positive voltage switches the memristor to a low resistance state and a negative voltage switches it to a high resistance state, Eqs.(2.50) and (2.51) can be used to determine the resistances mentioned above.

$$R_{on} = \left. \frac{R_L}{m} \right|_{x=\max(V_A)} - R_L \quad (2.50)$$

$$R_{off} = \left. \frac{R_L}{m} \right|_{x=\min(V_A)} - R_L \quad (2.51)$$

2. Tune  $\tau$ . The modeled IV-characteristic must tend to a straight line roughly at same frequency as the measured one.
3. Tune  $V_{on}$  and  $V_{off}$ . The knee voltages at which the modeled IV-characteristic changes its slope must coincide with the ones observed in the measured case.
4. Check if the IV-characteristics tend to a straight line at the same frequency. If they do not, go to step 2.

The above procedure yielded the following parameters values:

Table 2.3: Model parameters obtained

$R_{on}(\Omega)$	$R_{off}(\Omega)$	$V_{on}(V)$	$V_{off}(V)$	$\tau(s)$
440	8.2 k	300 m	265 m	1 m

Introducing these parameters in the cadence model gives the IV-characteristics as shown in Fig.2.28. The  $x_0$  parameter of the model was set to 0.5 so it is easier to see the IV-characteristics at higher frequencies, since the state of the memristor will converge to this value when driving with higher frequency signals. Setting the initial state to 0 or 1 would cause the simulation to show several loops until the state converges to the aforementioned 0.5 value.

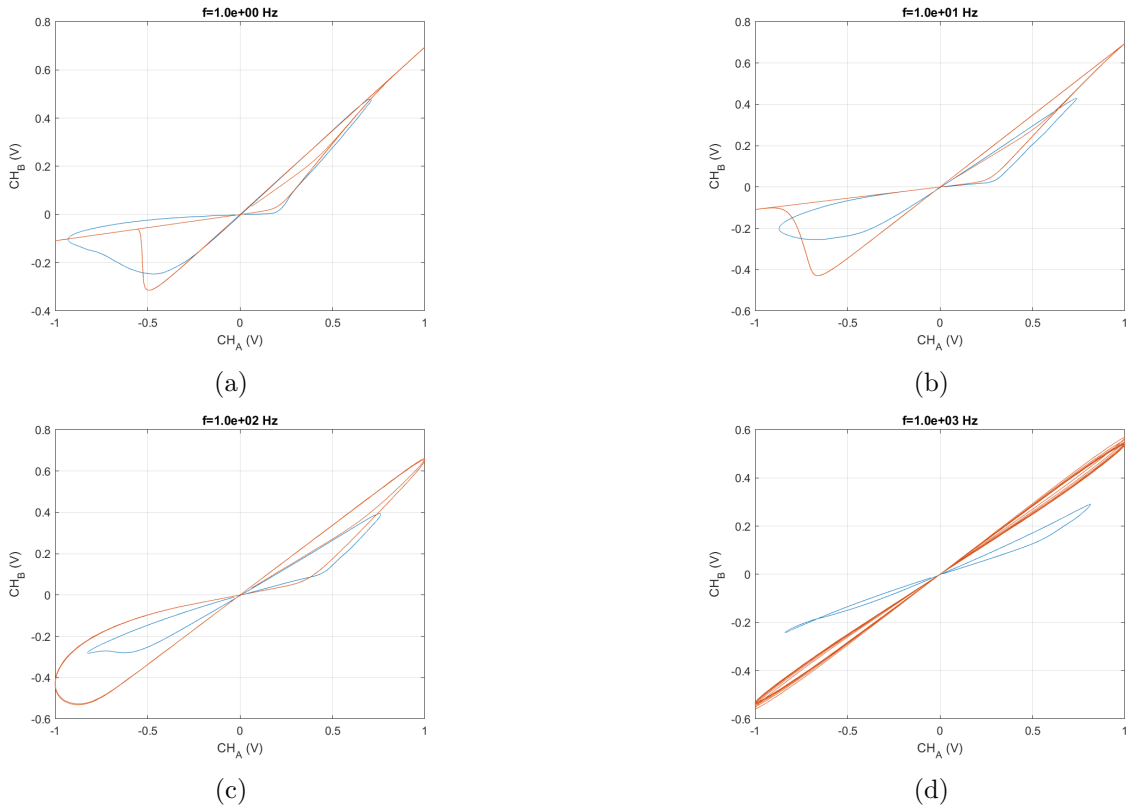


Figure 2.28: Comparison between modeled and measured IV-characteristics

As it can be seen, at the lower frequencies the IV-characteristics are reasonably similar, and they tend to a straight line at roughly the same frequency, even though its slope is considerably different. However, the memristor is not to be operated in this regime, since there is no change in its state and, therefore, would be useless in a logic circuit.

It should be noted that, the circuits will be simulated at lower frequencies than the ones one would expect from a real life product, such as processors and memories, if they were to be

made using memristors. Nevertheless, the results obtained in the simulations performed in this dissertation should be easily scalable to faster devices.

## **2.4 Final Remarks**

This chapter gave a brief introduction to what a memristor is. It was shown that the idea of a memristor originated from a necessity for completeness. It was also concluded that each device identified as a memristor belongs to one of four classes. Then, the DRM and the POP were introduced along with the "Two is infinite" theorem and the no backtracking rule. Then, several memristor models were presented and one of them was implemented on the Cadence simulator.

## Chapter 3

# Implementation of Logic Gates with Memristors

In this chapter, a simplified view on the switching dynamics of memristors is shown. Then, various techniques for designing logic circuits with memristors are presented. These techniques are also summarized in [20], but are presented here for the reader's convenience.

### 3.1 Memristor Switching Dynamics

In order to understand the behavior of memristive circuits, one must understand the switching dynamics of a single memristor and, from that knowledge, one is able to deduce the switching behavior of networks of memristors. In the circuits that will be presented, if not otherwise specified, it is assumed that a memristor polarized as in Fig.3.1(a) behaves as shown in Fig.3.2(a) and a memristor polarized as in Fig.3.1(b) behaves as shown in Fig.3.2(b). The terminal signaled by a black rectangle is called the bottom terminal and the other terminal is called the top terminal. When a memristor is polarized as shown in Fig.3.1(a), it is referred to as a Forward Polarized Memristor(FPM). If it would be polarized as shown in Fig.3.1(b), it would be referred to as a Reverse Polarized Memristor(RPM).

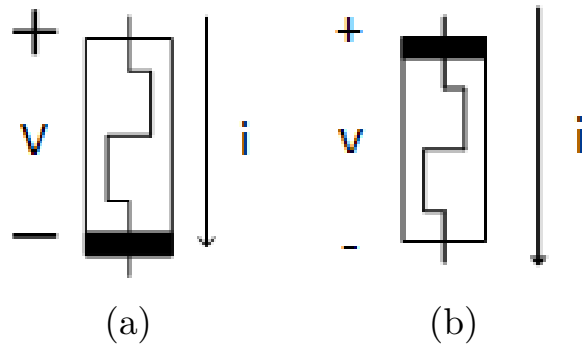


Figure 3.1: (a) Forward Polarized Memristor(FPM). (b) Reverse Polarized Memristor

As it can be seen in Fig.3.2, the conductance (resistance) of the memristor changes from a low (high) value, henceforth referred to as  $G_{off}$  ( $R_{off}$ ), to a high (low) value, henceforth referred to as  $G_{on}$  ( $R_{on}$ ), when the voltage across the device is higher than the threshold voltage  $V_{set}$ . The reverse change from  $G_{on}$  ( $R_{on}$ ) to  $G_{off}$  ( $R_{off}$ ) happens when the voltage across the device is lower than the threshold voltage  $V_{reset}$ . For a RPM, the IV characteristic is simply the symmetric of the one shown in Fig.3.2, with respect to the origin. It is assumed hereafter, that  $G_{on} \gg G_{off}$  and  $R_{off} \gg R_{on}$ , and that the initial states of a RPM and a FPM are ON ('1') and OFF ('0'), respectively.

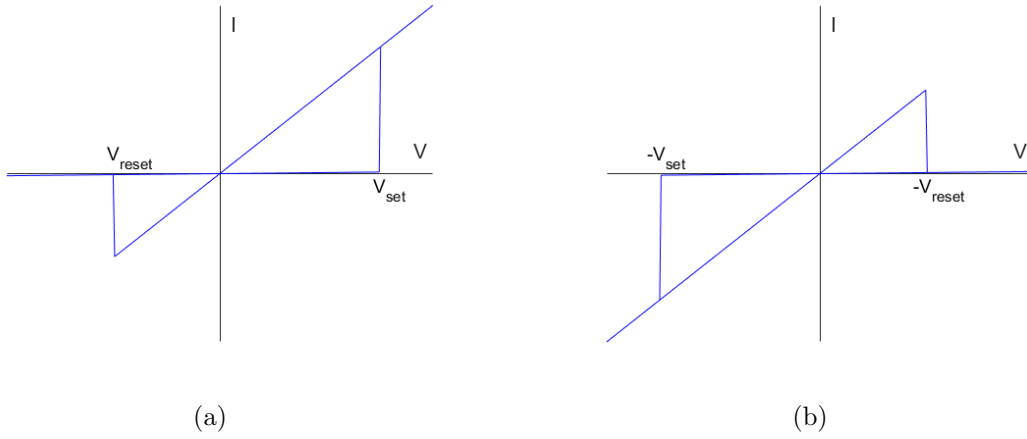


Figure 3.2: (a) IV characteristic of a forward polarized memristor. (b) IV characteristic of a reverse polarized memristor

Two memristors can be combined in four different ways, as shown in Fig.3.3 [21].

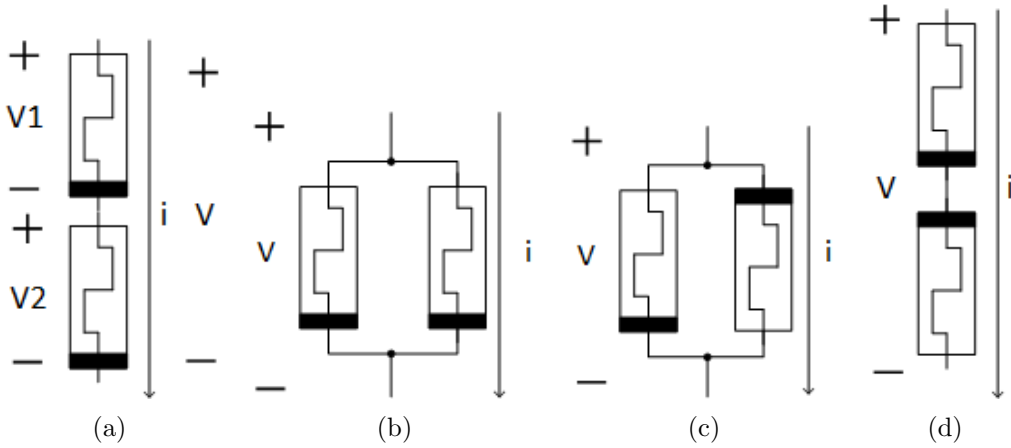


Figure 3.3: (a)-(d) Different combinations of memristors

If two memristors are combined in series as shown in Fig.3.3(a), the total voltage across the device,  $V$ , is equal to the sum of the voltages  $V_1$  and  $V_2$ . If both memristors are in the off state, the voltage in the central node is  $V_{central} = \frac{R_{off}}{2R_{off}} V = \frac{1}{2}V$ . So, to change the total



resistance of the composite device, from the off to the on state, a voltage  $V = 2 \times V_{set}$  must be applied, making the voltage drop across each memristor equal to  $V_{set}$ . Following the same reasoning, to change the overall state of the device from on to off,  $V = 2 \times V_{reset}$ . It can then be concluded that, the threshold voltages of the composite device are  $\{2 \times V_{reset}, 2 \times V_{set}\}$ , and the overall resistance lies in the range  $[2 \times R_{on}, 2 \times R_{off}]$ .

If two memristors are in parallel, as shown in Fig.3.3(b), the voltage across the two memristors is the same, so the threshold voltages of the compound device are equal to that of a single memristor. However, because the memristors are in parallel, the total resistance is  $[\frac{1}{2}R_{off}, \frac{1}{2}R_{on}]$ .

In figure 3.3(c), a RPM and a FPM are combined in parallel. Assuming that the initial states of the FPM and the RPM are, respectively, OFF and ON the resistance of the compound device is  $R_{on} \parallel R_{off} \approx R_{on}$ .

When the voltage  $V$  reaches  $V_{reset}$ , the RPM switches state and the overall resistance is  $R_{off} \parallel R_{off} = \frac{R_{off}}{2} > R_{on}$ . When the voltage  $V$  reaches  $V_{set}$ , the FPM's state switches to ON and the composite resistance is  $R_{on} \parallel R_{off} \approx R_{on}$ . Since the IV characteristic of the FPM and the RPM are symmetric, the IV characteristic of the parallel of said devices is also symmetric with respect to the origin. The IV characteristic of this composite structure is depicted in Fig.3.4.

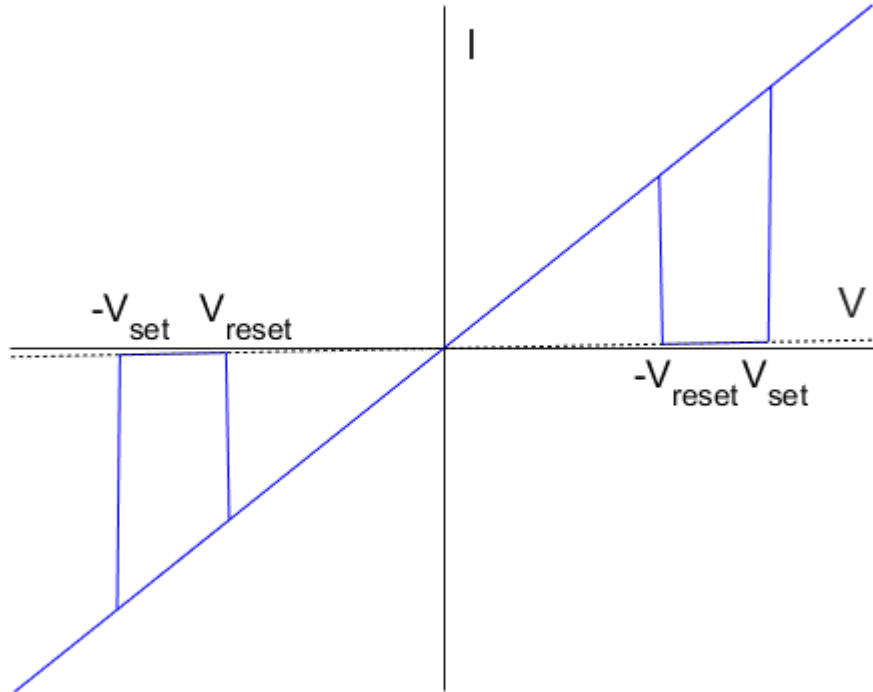


Figure 3.4: IV characteristic of the parallel composition of a FPM and a RPM

Finally, a RPM and a FPM can also be combined in series as shown in Fig.3.3(d). Again, assuming the initial states of the FPM and RPM are, respectively, OFF and ON, the total resistance is  $R_{off} + R_{on} \approx R_{off}$  and the voltage in the central node is  $V_{center} = \frac{R_{on}}{R_{off} + R_{on}} V \approx 0, R_{off} \gg R_{on}$ . If the voltage increases, the voltage across the FPM reaches  $V_{set}$  and it switches its state to ON. The voltage in the central node is  $V_{center} = \frac{R_{on}}{2 \times R_{on}} V = \frac{V}{2}$ . Further increasing the voltage  $V$  causes the RPM's voltage to reach  $V_{reset}$  and the RPM changes state to OFF. For negative voltages, since the FPM and RPM IV characteristics are symmetric with respect to the origin, the IV characteristic of the series association of these devices is also symmetric, as shown in Fig.3.5.

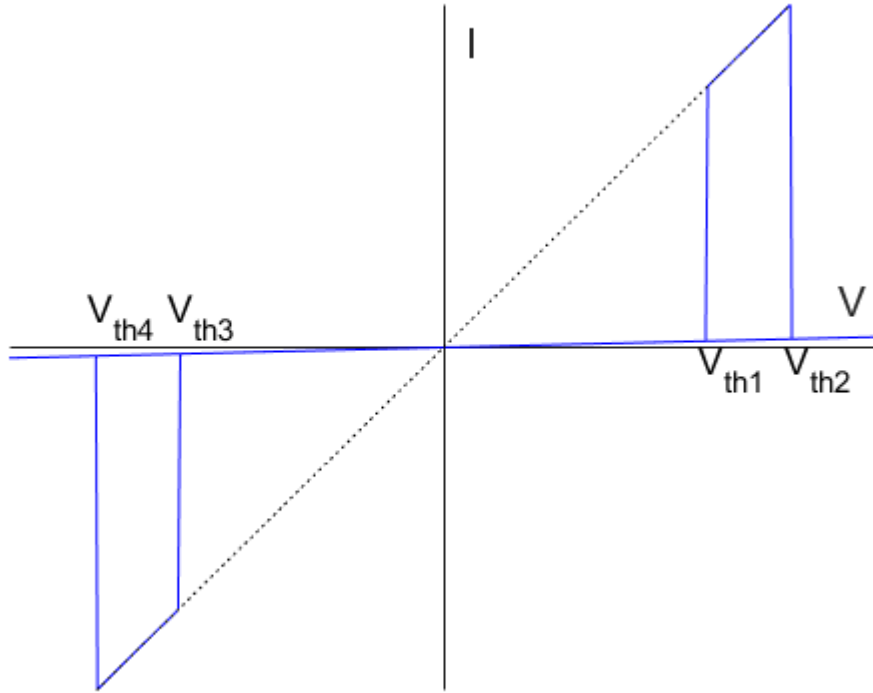


Figure 3.5: IV characteristic of the series composition of a FPM and a RPM

### 3.2 Classification of Memristive Logic Families

As suggested in [22], memristive logic families can be categorized according to a property called proximity of computation, which refers to how the data moves in the overall system. In Out-of memory computing, the data is read out of the memory element, processed by some dedicated element and then written back. In Near Memory computing, data moves in and out of memory, but is processed by a peripheral circuit, as opposed to Out-of Memory, where the data is processed by a completely external circuit. Finally, In Memory computing consists of processing data inside the memory element, without any need to read or write into it. Fig.3.6 illustrates the data movement of memristive systems with different proximity of computation.

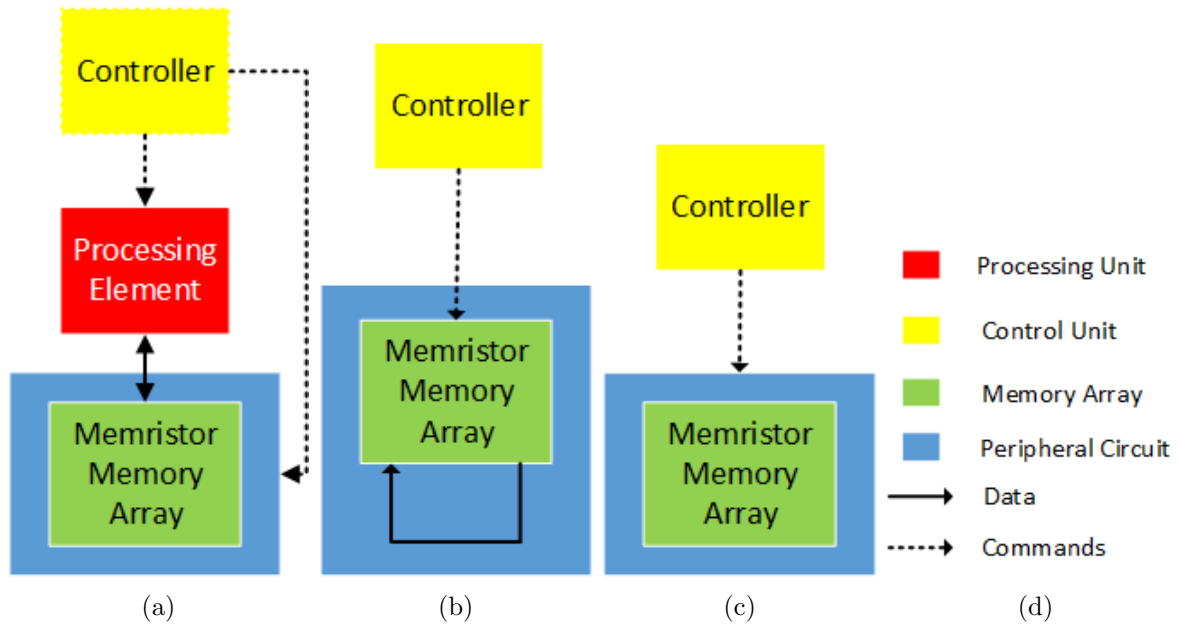


Figure 3.6: Illustration of data movement in memristive systems with different proximity of computation. (a) Out-of Memory computing. (b) Near Memory computing. (c) In Memory computing. (d) Legend for the previous block diagrams

The Memristive Memory Array and its peripheral circuit are usually implemented through a crossbar architecture such as the one shown in Fig.3.7, where the blue circles represent a memristive crossbar cell. This is done to take advantage of the 3D integration capabilities of the memristor by placing the row lines, column lines and the devices in 3 separate planes.

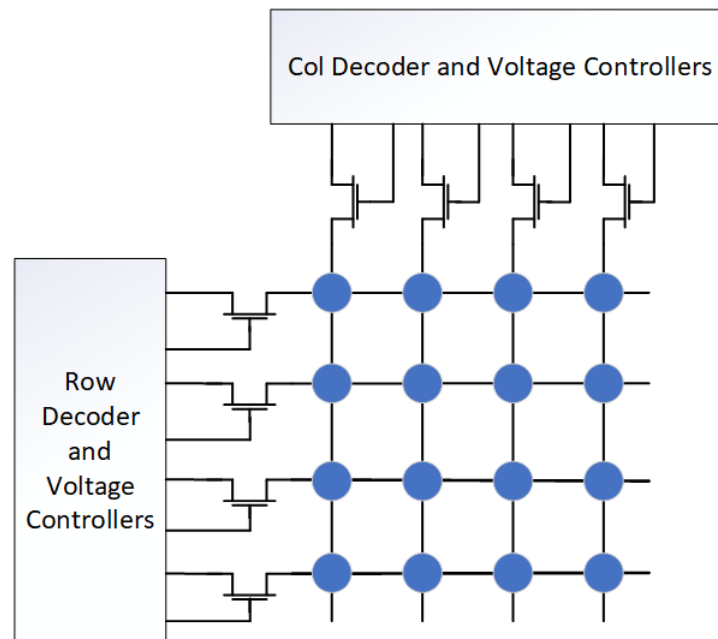


Figure 3.7: Illustration of a memristive crossbar

This proximity of computation property of the memristive system is somewhat correlated with a property of memristive logic families called the statefulness. A memristive logic family is said to be stateful if both the input and output of the logic gate are represented exclusively by the state of memristors. If the input or the output are represented by something other than the memristor's state, like a voltage, the logic family is said to be stateless.

These two properties are correlated because considering that all computing systems have a memory and a processing element, when both the input and output are represented as voltages, the values required must be read from the memory, processed as voltages and written back. This falls into the definition of an Out-of-Memory logic gate. If only one of the input or output is represented as voltage, some auxiliary circuit is required to act as a bridge between voltage and memristor's state, which is defined as Near Memory computing. If both the input and output are states of memristors, then all computing can be done inside the memory, as what happens with In-Memory computing.

### 3.3 Out-of-Memory computing logic families

This section presents some Out-of-Memory computing logic families. These logic families are characterized by only serving as a processing unit, without memory capabilities. However, as an upside, no conversion between memristors' states and voltage is required.

#### 3.3.1 MRL-Memristor Aided Logic

Memristors can be easily integrated within CMOS technology process, allowing higher device densities. The aim of the following design technique, which was proposed by Kvaternik et al. [23], targets memristor-CMOS integration. MRL combines memristor and CMOS logic gates to implement the OR and AND functions with memristive voltage dividers. The logic set is completed with the CMOS inverter, which is also used to regenerate the signals.

This logic family uses voltage as the input and output variables, instead of the state of the memristors, saving on additional circuitry to read the memristors' state. The output is the resulting voltage in the common node of the devices.

Fig.3.8 depicts the OR (a) and AND (b) gates using MRL.

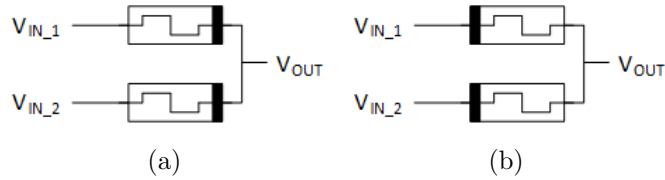


Figure 3.8: (a) MRL OR gate. (b) MRL AND gate.

In both cases, it can be observed that, whenever the inputs are the same, the voltage drop across the series composition is 0V, so no current flows through the devices and the voltage in the common node is equal to the voltage at the inputs. However, when the inputs are different, a voltage divider is formed, the FPM's resistance decreases and the RPM's resistance increases. For the OR gate, this means that the output voltage  $V_{out} = \frac{R_{off}}{R_{on} + R_{off}} V_1 \approx$

$V_1, R_{off} \gg R_{on}$ , as expected for an OR gate. For an AND gate, the output voltage is  $V_{out} = \frac{R_{on}}{R_{off} + R_{on}} V_1 \approx 0V, R_{off} \gg R_{on}$ , as expected of an AND gate.

The number of inputs of an MRL gate can be increased by connecting one more memristor to the common node for each input.

Since the output of any MRL gate is the output of a voltage divider, it can be observed that the level of the signal is degraded. It is also observable that this degradation is small if  $R_{off}$  is large when compared to  $R_{on}$ , which is usually true, and so this degradation is of small impact. However, this degradation can accumulate when cascading MRL gates, or with a large number of inputs, making the necessity of level regeneration evident. Additionally, more complex circuits can be created by other combinations of memristors and CMOS transistors [24].

### 3.3.2 CMOS/Memristor Threshold Logic

A LTG (Linear Threshold Gate) is an universal logic gate capable of implementing functions beyond Boolean logic [25]. The transfer function of a LTG with N inputs is defined as shown in Eq.3.1, where  $x_i$  is the i-th Boolean input variable,  $w_i$  is the weight of the input i and T is the threshold.

$$f(x_1, x_2, \dots, x_N) = \begin{cases} 1, & \sum_{i=1}^N w_i x_i \geq T \\ 0, & otherwise \end{cases} \quad (3.1)$$

For the special case of a symmetric LTG, where all weights have the same value, a maximum of N functions  $f^{(k)}(x_1, x_2, \dots, x_N)$  can be implemented, its weights are defined as  $w_1^{(k)} = w_2^{(k)} = \dots = w_N^{(k)} = w^{(k)} = \frac{1}{k}$  and the threshold  $T = 1$ , where  $k = 1, 2, \dots, n$ . By introducing these values into the previous equation and multiplying both sides of the condition of the first branch by  $k$ , Eq.3.1 becomes Eq.3.2.

$$f^{(k)}(x_1, x_2, \dots, x_N) = \begin{cases} 1, & \sum_{i=1}^N x_i \geq k \\ 0, & otherwise \end{cases} \quad (3.2)$$

In most implementations of LTGs, weights are fixed and unchangeable, where, for in-field reconfiguration purposes, programmable weights is a very appealing feature. The resistive switching properties of memristors make them a very strong candidate for the implementation of such a feature. Also, memristors allow for denser implementations of LTGs, compared to other methods.

To implement these programmable LTGs, a circuit such as the one shown in Fig.3.9(a) can be used. Here, it is assumed that the memristors are threshold-type with a linear conductance above a certain threshold. This conductance is also assumed to be programmable to any value in the interval  $[R_{on}^L, R_{on}^H]$ . The threshold is implemented with a pair of parallel diodes with opposite polarities, in series with the memristor, as shown in Fig.3.9(b).

The result of the LTG should be fed to a CMOS component, such as a D-Flip Flop or a CMOS inverter, to regenerate the voltage level and possibly invert it, thus allowing the output to drive other circuits.

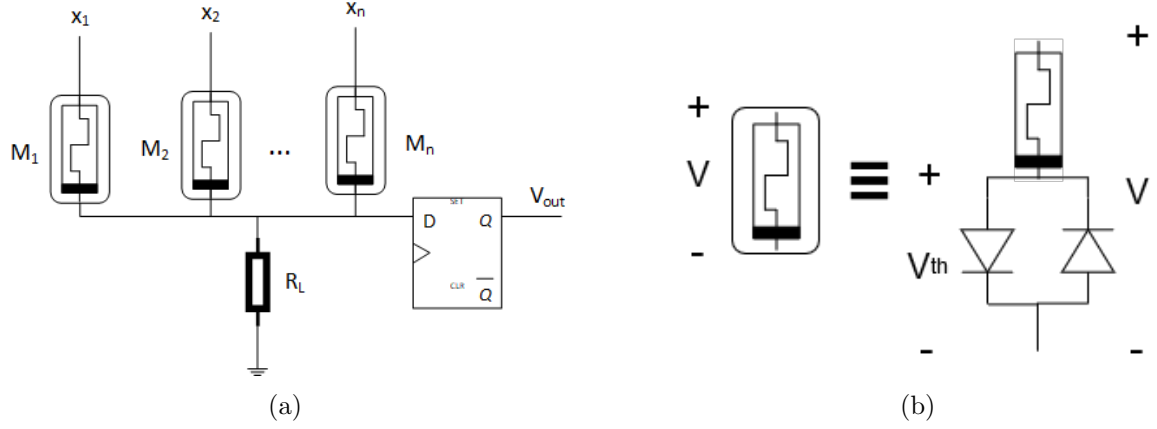


Figure 3.9: (a) Circuit implementation of an LTG. (b) Implementation of the memristors' switching threshold.

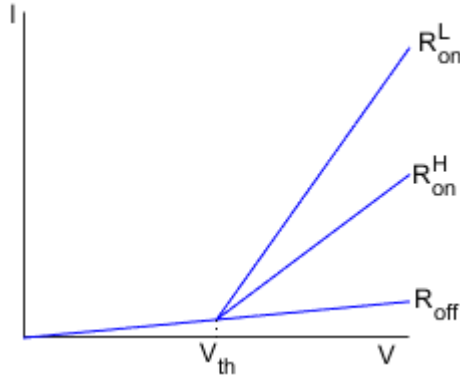


Figure 3.10: IV characteristics of a programmable memristor with two anti-parallel diodes in series

Analyzing the circuit in Fig.3.9(a), it can be seen that the current across  $R_L$  is the sum of the current across each memristor  $M_i$ , as shown in Eq.3.3, where  $V_{DD}$  is the circuit's operating voltage and  $R_i$  the  $i$ -th memristor's resistance. Here, it is assumed that  $V_{th} < V_{DD} < 2V_{th}$ . In this way, when an input is a '1', the diodes will short the corresponding to the rest of the circuit, whereas a '0' input causes the diodes to present an open circuit, so no current will flow to these memristors [25].

$$\frac{1}{R_L} V_0 = (V_{DD} - V_0) \sum_{i=1}^N \frac{1}{R_i} x_i, \quad x_i \in \{0, 1\} \quad (3.3)$$

Solving the above equation with respect to  $V_0$  yields:

$$V_0 = \frac{\sum_{i=1}^N \frac{1}{R_i} x_i}{\frac{1}{R_L} + \sum_{i=1}^N \frac{1}{R_i} x_i} V_{DD}, \quad x_i \in \{0, 1\} \quad (3.4)$$

Letting  $\sum_{i=1}^N \frac{1}{R_i} x_i = \frac{1}{R_L}$ , yields  $V_0 = \frac{V_{DD}}{2}$ , which means that, assuming the threshold voltage of the CMOS element is  $\frac{V_{DD}}{2}$ , the circuit shown in Fig.3.9(a) implements a LTG, such that  $w_i = \frac{1}{R_i}$  and  $T = \frac{1}{R_L}$ . If the LTG is symmetric, the weights  $w^{(k)}$  and threshold  $T$  must be chosen such that  $(k-1)R_L < R^{(k)} < k \times R_L$ . The implementation of all  $n$  possible Boolean functions requires that the interval  $[R_L, N \times R_L]$  lies within the interval  $[R_{on}^L, R_{on}^H]$ . In the limit,  $R_{on}^L = R_L$  and  $R_{on}^H = N \times R_L$ , so  $\frac{R_{on}^H}{R_{on}^L} = N$ . As for programming the memristors, several methods can be employed for this effect [26–30]. One such method is depicted in Fig.3.11.

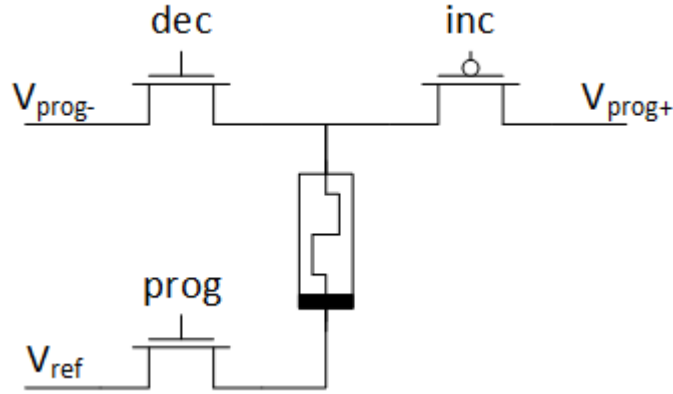


Figure 3.11: Memristor programming circuit

The memristor is firstly initialized to either  $R_{on}$  or  $R_{off}$ , and then, either by using pulses or a continuous signal, the resistance is tuned to the desired value. This is done by fixing the pulse width in the gate of the top transistors and amplitudes  $V_{prog}$  and, by doing so, the weight of the memristor can be tuned by acting on how many pulses are applied, or by how long the signal is applied in the continuous case. Fig.3.12 shows how the resistance of the memristor changes under this regime, when a square pulse with a period of  $5 \mu s$  and an amplitude of  $\pm 500$  mV for an initial state of '0' and '1', respectively.

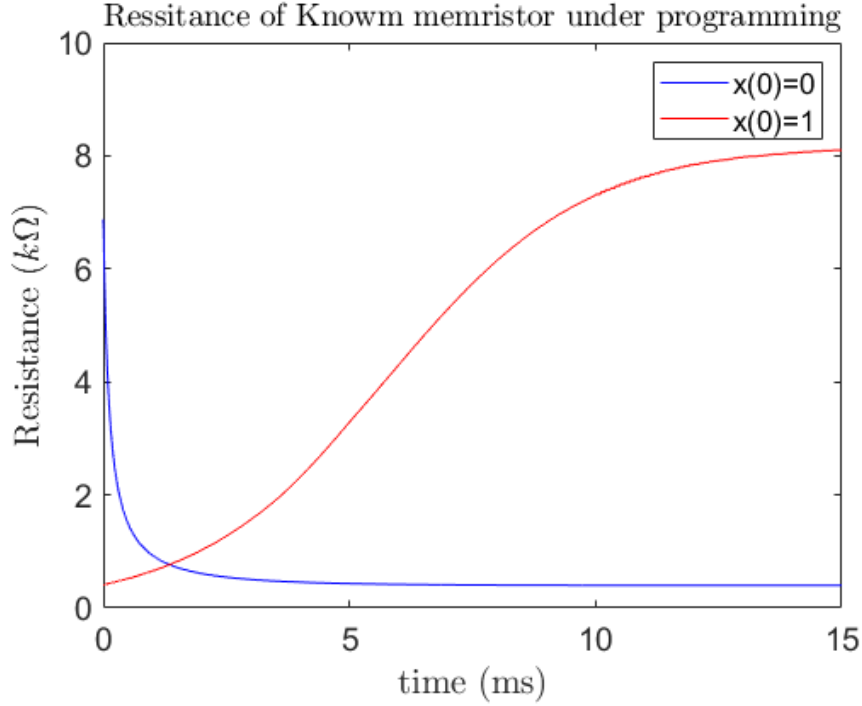


Figure 3.12: Resistance programming of a memristor

As it can be seen in Fig.3.12, when the initial state is '1' and is increased, the resistance changes with less sensitivity than it does when the initial state is '0', meaning the memristor is more easily programmed by initializing its state to '1' first and then decreasing it.

As for the computing stage, restrictions must be set to the values of the input voltages and the weights themselves. Assuming that the input voltage is  $V_{DD}$  for a logic '1' and 0V for a logic '0' and that the threshold is  $T = \frac{V_{DD}}{2}$ , these restrictions apply only to the value of  $V_{DD}$  and, since the inputs in an AND gate should be equally weighted, the resistance of the memristors  $R_w$ .

Since the input voltages should not interfere with the weights of the memristors:

$$V_{DD} - V_0 < V_{set} \quad (3.5)$$

Applying Eq.(3.4) to the above equation and solving with respect to  $V_{DD}$  yields:

$$V_{DD} < (1 + R_L \sum_{i=1}^N \frac{1}{R_i} x_i) \quad (3.6)$$

As for the weights, for an N-input AND gate, when all inputs are '1' the result should be '1',



otherwise '0'. This means that:

$$\begin{cases} \frac{R_L}{R_L + R_w/N} V_{DD} > \frac{V_{DD}}{2} \\ \frac{R_L}{R_L + R_w/(N-1)} V_{DD} < \frac{V_{DD}}{2} \end{cases} \quad (3.7)$$

Which means that:

$$\begin{cases} R_w < NR_L \\ R_w > (N-1)R_L \end{cases} \quad (3.8)$$

In [25], this circuit is presented with two anti-parallel diodes in series with each memristor, so that when a '0' is applied, this presents an open-circuit, preventing current leakage. But, in an integrated circuit, these diodes are created by producing a p-n junction with some area and perimeter as parameters. These parameters make the diode's behavior unpredictable.

To circumvent this problem, instead of diodes, a single MOSFET before the memristor is used and the input signal is connected to the gate. Since the input signal would be  $V_{DD}$  for logic '1', the FET is a PMOS, which makes the input active-low, so now the previously discussed AND gate has both inputs inverted, making it a NOR gate.

The circuit used to evaluate the logic operation can be seen in Fig.3.13.

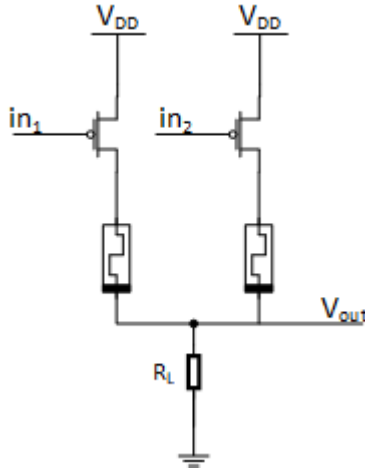


Figure 3.13: Circuit used to evaluate a Threshold Logic gate

### 3.4 Near Memory computing logic families

The following section presents some Near Memory computing logic families. Here, only the circuits that process data are presented. However, it should be noted that what makes these logic families Near Memory computing families is that these processing circuits would be implemented in a crossbar [31–33], which can receive logic values from the memory.

### 3.4.1 CMOS-like Memristor Complementary Logic

The next design paradigm, proposed by Vourkas et al. [31,34], aims at decreasing the circuit complexity imposed by IMPLY Logic, while providing easy implementation of logic functions.

Here, an approach similar to CMOS circuits is adopted, where the NMOS and PMOS transistors are, respectively, replaced by FPMs and RPMs, while maintaining the same design methodology. The logic inputs are represented as a positive voltage for '1' and a negative voltage for '0'. As previously shown, a high enough positive voltage makes the FPMs behave as a closed switch while a negative voltage makes it behave like an open switch, and vice-versa for RPMs, just like in regular CMOS circuits.

The operation of such a logic gate is defined by the topology of the circuit, and its result is determined by the voltage divider formed by the memristors, where a voltage close to  $V_{DD}$  represents a '1', while a voltage close to 0 represents a '0'.

Fig.3.14 shows an example of a complementary resistive switch (CRS). If the FPM's resistance is  $R_{off}$  and the RPM's is  $R_{on}$ , the output voltage is  $V_{out} = \frac{R_{on}}{R_{on}+R_{off}}V_{DD} \approx 0V, R_{off} \gg R_{on}$ . If, on the other hand, the FPM's resistance is  $R_{on}$  and the RPM's resistance is  $R_{off}$ , the output voltage is  $V_{out} = \frac{R_{off}}{R_{on}+R_{off}} \approx V_{DD}, R_{off} \gg R_{on}$ .

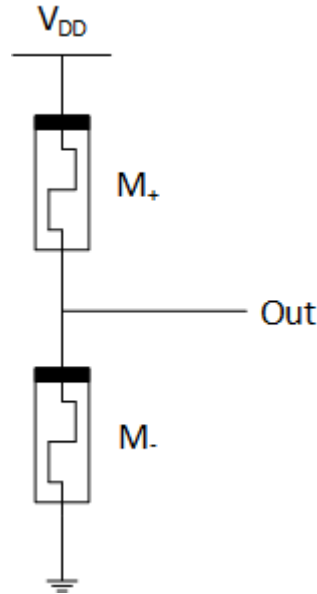


Figure 3.14: CMOS-like Memristor Complementary Logic NOT Gate

All that remains is some way to program the circuit so that the memristors are programmed with complementary states. This is solved by using a circuit architecture such as the one shown in Fig.3.15, which implements a NOR Gate. The circuit's driving mechanism for each operation phase is described in Tab.3.1.

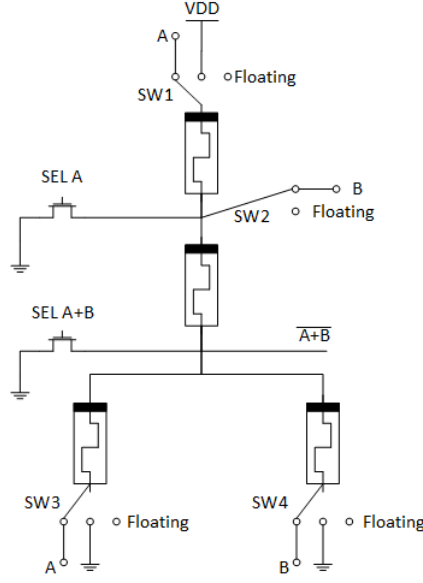


Figure 3.15: Circuit architecture of a CMOS-like NOR gate

Table 3.1: Switch positions for all operation phases

	Read	Apply A	Apply B
SEL A	0	1	0
SEL A+B	0	1	1
SW 1	VDD	A	Floating
SW 2	Floating	Floating	B
SW 3	GND	A	Floating
SW 4	GND	Floating	B

As for the voltage values, there are also some restrictions: the input signals' voltage should be chosen such that it exceeds the memristors' thresholds and the pulse length should be greater than the switching time of the devices. The operating voltage  $V_{DD}$  should not exceed the memristors' threshold voltages, in order for these to keep the internal states during reading phase.

The programming voltages are applied directly to the terminals of the memristors, except for the voltage drop across the selecting transistors. Hence this voltage should be greater than the maximum of the voltages  $V_{set}$  and  $-V_{reset}$ , when the input is a '1' and it should be smaller than the minimum between  $-V_{set}$  and  $V_{reset}$  for a '0'.

As for the reading voltage, the analysis is more particular to the gate in question. In general, this voltage should not change the state of any memristor.

The topology proposed for a NOT gate is as shown in Fig.3.16, which is based on [34]. The transmission gates are needed so the input signal that reaches the memristors, whether it is a '0' or a '1', has only a small voltage drop.

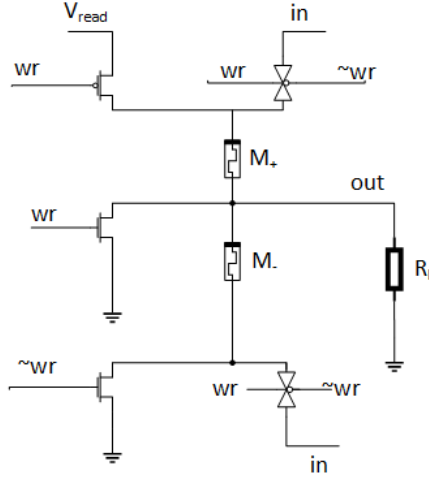


Figure 3.16: Circuit topology of a cmos-like NOT gate

The design constraints for this gate are:

$$\begin{cases} V_{out} - V_{read} < V_{set} & in = '1' \\ -V_{out} > V_{reset} & in = '1' \\ V_{out} - V_{read} > V_{reset} & in = '0' \\ -V_{out} < V_{set} & in = '0' \end{cases} \quad (3.9)$$

Assuming that  $V_{read}$  is a positive voltage, the first and last equations are guaranteed to be verified, and so they can be discarded. When  $in='1'$ ,  $M_1 = R_{off}$  and  $M_2 = R_{on}$  and when  $in='0'$ ,  $M_1 = R_{on}$  and  $M_2 = R_{off}$ . Then, by substituting in Eq.(3.9) and solving for  $V_{read}$  yields:

$$\begin{cases} V_{read} < -(1 + \frac{R_{off}}{R_{on} \parallel R_L})V_{reset} \\ V_{read} < -(1 + \frac{R_{off} \parallel R_L}{R_{on}})V_{reset} \end{cases} \quad (3.10)$$

Close analysis on the two members of Eq.(3.10) concludes that, since  $R_{on} \parallel R_L < R_{on}$  and  $R_{off} > R_{off} \parallel R_L$ , the bottom equation is less than the top one, so the latter can be discarded:

$$V_{read} < -(1 + \frac{R_{off} \parallel R_L}{R_{on}})V_{reset} \quad (3.11)$$

If the load is assumed to be an open circuit, and  $R_{on} < R_{off}$  the above expression can be further simplified:

$$V_{read} < -\frac{R_{off}}{R_{on}}V_{reset} \quad (3.12)$$

Intuitively, from Eq.(3.12), it would seem that the upper bound for  $V_{read}$  is too high and high voltages should affect the state of the memristors. However, that is not the case, because in

the case where the output is '0', meaning that  $M_+ = R_{off}$  and  $M_- = R_{on}$ , the voltage across  $M_-$  is 0V and it remains unaffected, and  $V_{out} - V_{read} < 0V$  which actually reinforces the '0' state of  $M_+$ . The same effect is observed when the output is '1'.

One interesting observation is that flipping the polarity of a memristor is equivalent to variable complement. However, the effect described above is no longer observed, so the voltage conditions are more restrictive. For example, flipping the polarity of both memristors of the circuit in Fig.3.14, when the output is '0', the voltages and resistances of the memristors are still as described above. However,  $M_+ = R_{off}$  is now forward polarized, so the voltage across its terminals will cause it to switch to '1'. This aspect is somewhat analogous to a complementary CMOS circuit, where the pull-up network is composed only of PMOS and the pull-down network is composed only of NMOS because a PMOS passes a weak '0' and the NMOS passes a weak '1' and building a non inverting logic gate would diminish the voltage excursion of the circuit, the same way as if a non inverting gate were designed following the CMOS-like Memristor Complementary logic strategy.

### 3.4.2 Parallel Input-Processing Memristor Logic

Most of the memristor logic design methodologies (IMPLY, MAGIC, CMOS-like) consider a serial processing of inputs, whereas parallel processing would be more advantageous. The MRL methodology attains this end, but the set of Boolean operations is very restricted. The next logic family, which was proposed by Papandroulidakis et al. [35], aims at parallel processing of inputs by exploiting the switching behavior of compositions of memristors. In Fig.3.17 the IV characteristics of the compositions used for logic gates are shown. These characteristics can be derived from the ones presented in section 3.1

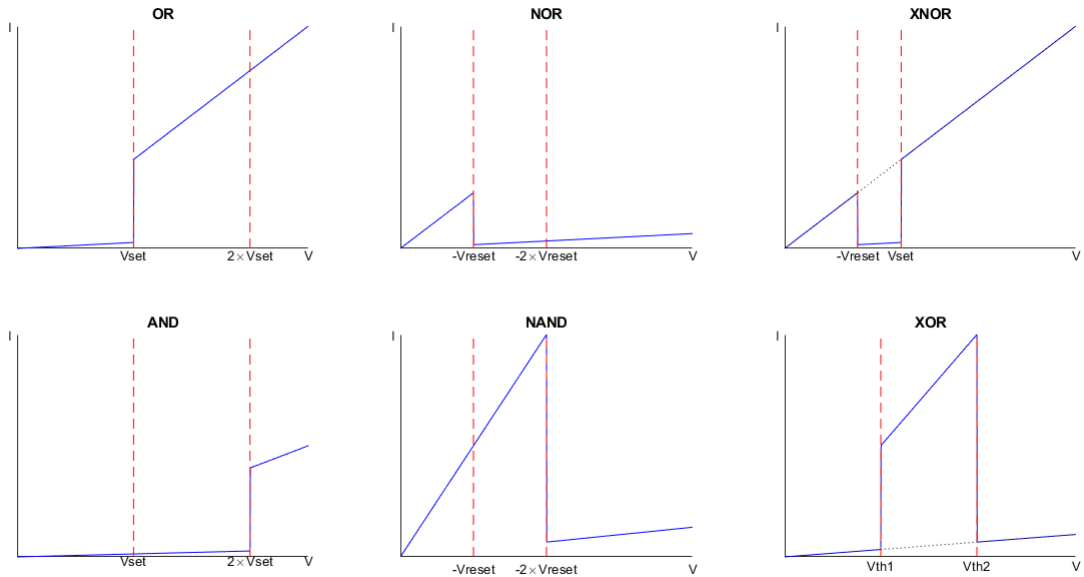


Figure 3.17: IV characteristics of composite memristive circuits

As it can be seen from Fig.3.17, different conductances are achieved by applying different voltages to the composite device. By assuming the logic '0' to be 0 V and the logic '1' to be some voltage between the first and second thresholds, then, applying the sum of all the input voltages to the memristive composition, a logic gate is obtained, whose function depends on the topology of the circuit and the result is stored in the composite device. Fig.3.18 shows a simplified version of the implementation of such a gate.

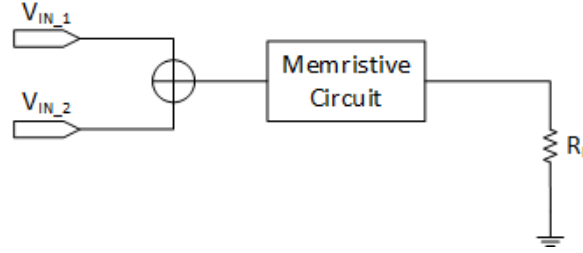


Figure 3.18: Simplification of the circuit Implementation of a Parallel Input-Processing Gate

Tab.3.2 summarizes the compositions of memristors that can be substituted into the memristive circuit block to implement a two input logic gate.

Table 3.2: Two variable logic gates implementations

AND	OR	NAND	NOR/NOT	XOR	XNOR

The number of inputs can be increased for such gates. For OR and NOR gates, the simple addition of extra input ports achieves this purpose. The maximum number of inputs is only restricted by the maximum power a memristor can dissipate before being damaged. For AND and NAND, however, an extra series memristor of the same polarity must also be included per input.

Since the memristors used should be non volatile, the inputs may change the conductance of the devices, which may lead to subsequent wrong results. Generally speaking, any combination of inputs whose voltage exceeds a threshold of the memristive block will change its state irreversibly, unless a reset step is applied. This reset step can be achieved simply by applying a reset pulse to the compound block, without accessing each memristor separately.

The resistor  $R_L$  should have a small resistance when compared to  $R_{on}$ , in order to not hinder the switching of the memristors, unless it can be applied only during the reading phase. Also, even though cascading is possible, memristors are passive devices and, therefore, the output

voltage levels degrade with each gate, which means that, for long cascades of logic gates, CMOS circuits may be necessary to restore the signal.

The generic circuit proposed is shown in Fig.3.19, where the signals with a tilde( $\sim$ ) before the name represent digital signals which are active low.

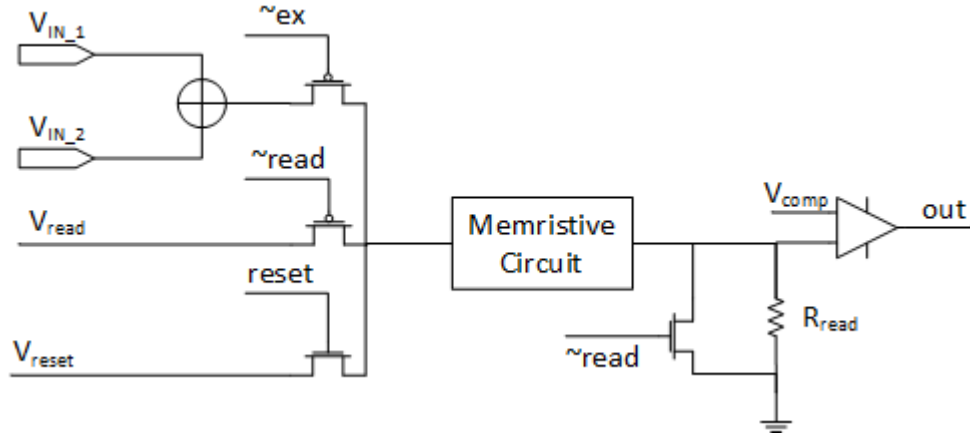


Figure 3.19: Implementation of a Parallel Input Processing circuit

The voltage  $V_{reset}$  is used to reset the state of the memristive circuit. Its value, as do all voltages, depends on the actual circuit inside the box, but, in general, it will be a negative voltage high enough (in absolute value) to force a known state upon the memristive circuit. The voltage  $V_{read}$ , however, should be such that it doesn't affect the state of the memristors when applied.

The three signals " $\sim ex$ ", " $reset$ " and " $\sim read$ " should be mutually exclusive, meaning that only one of those should be active at the same time.

When " $\sim ex$ " is active, the circuit becomes the one in Fig.3.20.

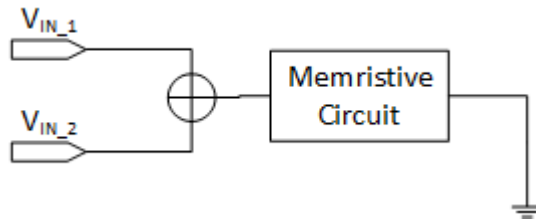


Figure 3.20: Equivalent circuit when " $\sim ex$ " is active

The transistor shown in parallel with  $R_{read}$  in Fig.3.19 allows for the shorting out of that resistor. This not only simplifies the problem of choosing the values of the inputs voltages, but also makes the switching faster, since the voltage drop across the memristive circuit is greater. This was also proposed in [20].

When " $reset$ " is active, the reset voltage is simply applied to the circuit, resetting it. Finally, when " $\sim read$ " is active the equivalent circuit is as shown in Fig.3.21.

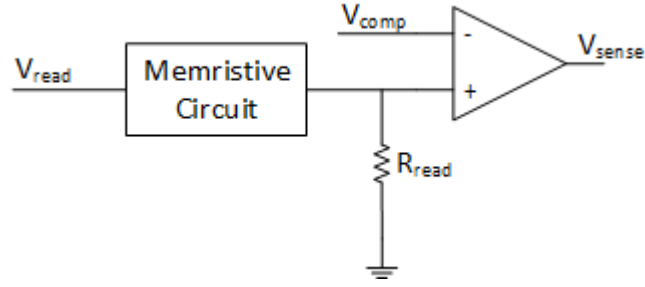


Figure 3.21: Equivalent circuit when " $\sim$ read" is active

Now the  $V_{read}$  voltage is applied to a voltage divider between the memristive circuit and  $R_{read}$ . This voltage divider produces several different voltages depending on the resistance of the memristive circuit. The result of this division is then compared to another voltage to determine the result of the operation.

### 3.5 In Memory computing logic families

The next section discusses some In Memory computing logic families. These logic families are different in that all processing is done inside the memory element, using only memristors' states for memory and computing, without any reading or writing in between.

#### 3.5.1 IMPLY Logic

Fig.3.22 depicts the architecture of an ImPLY Logic gate [36,37], where  $R_{on} \ll R_G \ll R_{off}$  and  $V_{cond}$  is a voltage such that it does not affect the state of the memristor to which it is applied. The memristors are driven by tri-state buffers, which yield a high-impedance output when not driven. In this architecture, the gate's inputs are the states of the memristors, and the output is the state of one of the memristors involved after the gate's output computation. A memristor is considered to hold a logic '1' if its resistance is  $R_{on}$  and a '0' if it has resistance  $R_{off}$ . As previously shown, a memristors resistance changes from a low value to a high value when a voltage higher than  $V_{set}$  is applied to it. Similarly, it changes from a high value to a low value if a voltage lower than  $V_{reset}$  is applied.

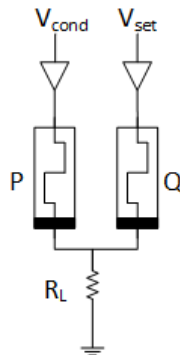


Figure 3.22: IMPLY Logic circuit architecture



The idea behind this technique is the material implication logic function which is represented as " $p \implies q$ " and is read as "p implies q" or "if p then q". This function, combined with the logic constant '0' yields a computationally complete set. The truth table of the imply operation is shown in table 3.3.

Table 3.3: Material implication truth table

$p$	$q$	$p \implies q$
0	0	1
0	1	1
1	0	0
1	1	1

To operate such a circuit, two pulses must be applied simultaneously. For example, to implement  $q \leftarrow p \implies q$ , where  $\leftarrow$  means that q's state becomes the result of  $p \implies q$ , a pulse of amplitude  $V_{set}$  must be applied to  $Q$  and a pulse of amplitude  $V_{cond}$  to  $P$ . Applying solely  $V_{cond}$  to  $P$  does not change its state, and applying just  $V_{set}$  to  $Q$ , makes  $q \leftarrow '1'$ . When applied together, these pulses interact through the voltage divider formed by  $P$ ,  $Q$  and  $R_G$  and cause changes to the memristors' states, dependent on the current states of the devices. If  $P = R_{off}$ , the voltage divider is simply  $Q$  and  $R_G$ . If  $Q = R_{off}$  then the voltage in the central node is  $V_{center} = \frac{R_G}{R_G + R_{off}} V_{set} \approx 0$  and  $Q$  changes state from off to on. On the other hand, if  $Q = R_{on}$ , then  $V_{center} = \frac{R_G}{R_G + R_{on}} V_{set} \approx V_{set}$  and  $Q$  does not change its state. So, it can be concluded that, if  $P = '0'$  then, applying the two pulses makes  $q \leftarrow '1'$ . If  $P = R_{on}$ , the voltage across  $Q$  is  $V_{set} - V_{cond}$  and  $Q$  remains in the same state. Both of the above situations, where  $P = '0'$  and  $P = '1'$  describe the material implication operation.

As said above, the material implication and the '0' constant, constitute a complete set of Boolean operations. This is done by cascading implies in such a way as to obtain the desired logic operation. Table 3.4 lists the basic logic operations, and their equivalent using only the material implication operation and the constant '0'.

Table 3.4: Basic logic operations and equivalents using implies and '0'

Operation	Equivalent using implies
$\bar{q}$	$q \implies 0$
$p \cdot q$	$(p \implies (q \implies 0)) \implies 0$
$p + q$	$(p \implies 0) \implies q$
$\overline{p \cdot q}$	$p \implies (q \implies 0)$
$\overline{p + q}$	$((p \implies 0) \implies q) \implies 0$
$p \oplus q$	$(p \implies q) \implies ((q \implies p) \implies 0)$

The table above reveals a big disadvantage in this design technique: in order to implement a basic logic operation, such as an OR, it is necessary to perform several sequential imply operations. In fact, it can be shown that, for a logic gate of  $n$  inputs, up to  $2^{n-1} + 1$  steps are required to compute said gate's output [36]. Also, since all operations require the use of the constant '0', gates with multiple inputs require, at least, a third memristor to hold that constant, without destroying the value of the inputs before the operation, as shown in

Fig.3.23. An example of the operation of such a circuit can be seen in table 3.5, where the implementation of an OR gate through the use of an IMPLY logic circuit can be seen.

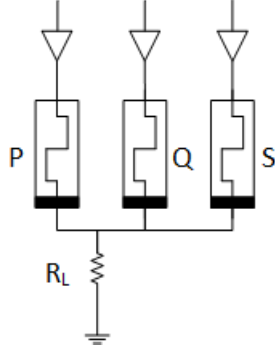


Table 3.5: Steps to perform an OR gate through IMPLY Logic

Step	$V_p$	$V_q$	$V_s$
$s \leftarrow 0$			$V_{reset}$
$s \leftarrow p \implies s$	$V_{cond}$		$V_{set}$
$q \leftarrow s \implies q$		$V_{set}$	$V_{cond}$

Figure 3.23: IMPLY Logic circuit using 3 memristors.

It is noteworthy that any gate that uses this methodology is composed of several sequential IMPLY operations. This means that the analysis to determine the voltage values to be used is reduced to the simple case of the IMPLY operation. Recalling the circuit architecture of an IMPLY logic gate from Fig.3.22, it can be observed that the current flowing through  $R_G$  is the sum of the currents flowing through the memristors:

$$\frac{V_L}{R_L} = \frac{V_{cond} - V_L}{M_P} + \frac{V_{set} - V_L}{M_Q} \quad (3.13)$$

where  $V_L$  is the voltage across  $R_L$  and  $M_P$  and  $M_Q$  are the resistances of the memristors P and Q, respectively.

Solving with respect to  $V_L$ :

$$V_L = \frac{V_{cond}/M_P + V_{set}/M_Q}{R_L^{-1} + M_P^{-1} + M_Q^{-1}} \quad (3.14)$$

Noticing that the voltage across memristor P,  $V_P$ , is the difference between  $V_{cond}$  and  $V_L$ , it follows that:

$$V_P = V_{cond} - V_L = \frac{(M_Q + R_L)V_{cond} - R_L V_{set}}{M_Q + R_L(1 + M_Q/M_P)} \quad (3.15)$$

Similarly, for the voltage across memristor Q:

$$V_Q = V_{set} - V_L = \frac{(M_P + R_L)V_{set} - R_L V_{cond}}{M_P + R_L(1 + M_P/M_Q)} \quad (3.16)$$

The operation of this logic gate branches into two cases: the case where  $Q=0$  and  $Q=1$ . The latter case is rather uninteresting: the proper functioning of the gate in this case is guaranteed

because Q's state is reinforced by the positive voltage. On the other hand, the case where Q='0' requires more study. The correct functioning of the gate in this case is enforced by the following conditions, where  $V_{th}$  is the positive threshold voltage of the memristors, to distinguish from the  $V_{set}$  voltage:

$$\begin{cases} V_Q \geq V_{th}, & P = Q = '0' \\ V_Q \leq V_{th}, & P = '1', Q = '0' \end{cases} \quad (3.17)$$

The different states of P make  $M_P$  have different values between conditions, namely, when P='1',  $M_P = R_{on}$  and when P='0',  $M_P = R_{off}$ . Since Q='0',  $M_Q = R_{off}$ . Substituting Eq.(3.16) into Eq.(3.17) and rearranging, yields Eq.(3.18). As a simplification, the term  $R_{on}/R_{off}$  is considered to be very small and hence ignored.

$$\begin{cases} R_{off}V_{set} + R_L(V_{set} - V_{cond}) \geq (R_{off} + 2R_L)V_{th} \\ R_{on}V_{set} + R_L(V_{set} - V_{cond}) \leq (R_{on} + R_L)V_{th} \end{cases} \quad (3.18)$$

Subtracting both shows that:

$$V_{set} \geq (1 + \frac{R_L}{R_{off} - R_{on}})V_{th} \quad (3.19)$$

Solving both inequations with respect to  $V_{cond}$  gives:

$$\begin{cases} V_{cond} \leq (1 + \frac{R_{off}}{R_L})(V_{set} - V_{th}) - V_{th} \\ V_{cond} \geq (1 + \frac{R_{on}}{R_L})(V_{set} - V_{th}) \end{cases} \quad (3.20)$$

In conclusion,  $V_{set}$  and  $V_{cond}$  should be chosen so that  $V_{cond}$  is below the positive threshold of the memristor, and ensuring the conditions imposed by Eqs.(3.19) and (3.20) are met. As a final observation, assuming  $R_{on} \ll R_L \ll R_{off}$  allows the following simplifications to these conditions:

$$\begin{cases} V_{set} \geq V_{th} \\ V_{cond} \leq \frac{R_{off}}{R_L}(V_{set} - V_{th}) \\ V_{cond} \geq V_{set} - V_{th} \end{cases} \quad (3.21)$$

Now all that remains is some circuit that allows the reading of the state of the output memristor. Since a voltage divider is already formed between the memristors and a fixed resistance, that divider can be used to compare the resistance of the memristor with said reference resistor. The final circuit is presented in Fig.3.24, where the logic gate was implemented following a crossbar circuit design paradigm [37].

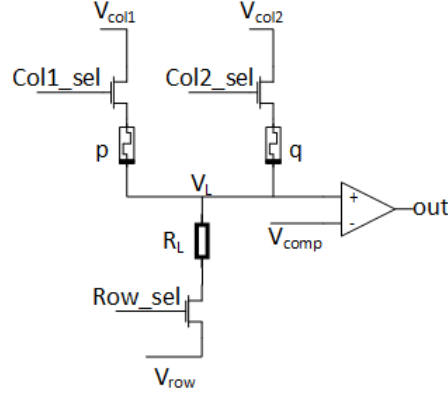


Figure 3.24: ImPLY circuit final design

The states of the memristors are written one at a time. To write into a memristor, its column and row are selected. Then, to write a '1', the column voltage is set to  $V_{set}$  and the row voltage to ground. To write a '0', the opposite occurs: the column is connected to ground and the row voltage to  $V_{set}$ , thus effectively applying a negative voltage to the memristor.

To read the state of a memristor, the column and row of said memristor are selected. The row is connected to ground and the column to a reading voltage. This creates the voltage divider between the memristor and  $R_L$ , and the output of that voltage is compared to another voltage.

### 3.5.2 MAGIC-Memristor Aided loGIC

The following logic family was also proposed by Kvatinsky et al. [38,39] as an attempt to simplify the operation and complexity of other sequential logic techniques. This logic family is called MAGIC (Memristor Aided loGIC). Fig.3.25 shows the circuits for the gate set provided by this logic family. In this logic family, the input and output variables are the states of the memristors, specifically, a high resistance corresponds to the logic '0' and a low resistance to a logic '1'.

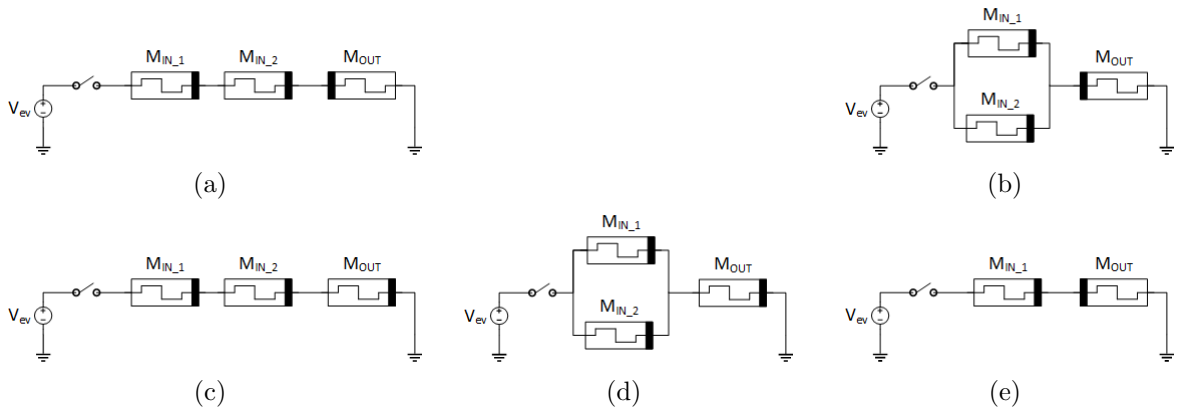


Figure 3.25: Circuit for different MAGIC gates: (a) NAND, (b) NOR, (c) AND, (d) OR, (e) NOT

The operation of a MAGIC gate consists of two sequential steps. First, the output memristor is set to a value dependent on the logic function and the input memristors are programmed to the proper state. Then a voltage pulse  $V_o$  is applied. When the function is inverting, i.e. a NOR, NAND or NOT, the output memristor should be set to '1', while if it is non-inverting, i.e. an OR or and AND, it should be set to '0'. The setting of the memristors is accomplished by applying an adequate voltage pulse to each of the devices, which requires the use of MOSFETs that are omitted in the above figure for simplicity.

The principle behind the behavior of a MAGIC gate is the voltage divider formed by the output memristor and the input memristors' combination. When the voltage pulse  $V_o$  is applied, a voltage, which depends on the states of the input devices, appears at the terminals of the output memristor, and may cause it to change state.

Taking, for example, the NAND gate shown in Fig.3.25(a), first,  $M_{out}$  must be set to '1' and the input memristors to their respective states. When  $V_o$  is applied,  $V_{out} = \frac{R_{out}}{R_{in1}+R_{in2}+R_{out}}V_o$ . Whenever one of the inputs is '0' and, therefore, its corresponding memristor's resistance is  $R_{off}$ ,  $V_{out} \approx 0V$ ,  $R_{off} \gg R_{on}$ , and  $M_{out}$ 's state remains unchanged. However, when both inputs are '1', the resistance of both memristors is  $R_{on}$ , so  $V_{out} = \frac{1}{3}V_o$ , and  $M_{out}$  switches its state to '0', assuming  $|V_{reset}| < \frac{1}{3}V_o$ .

The exact value of  $V_o$  should be chosen, taking into account that high voltages decrease the switching times of the memristors and, therefore, increases delay of the logic gates. However, the threshold voltages of the output memristor impose an upper bound on the value of  $V_o$ . This bound is dependent on the threshold voltages of the memristor, the number of inputs of the gate and the topology of the circuit.

From all the realizable logic gates under the MAGIC design paradigm, only the NOR and NOT can be easily implemented in a crossbar [38]. A possible circuit for a N input NOR gate, following a simplified version of the architecture presented in [38] and [39], the circuit to be analysed is the one shown in Fig.3.26.

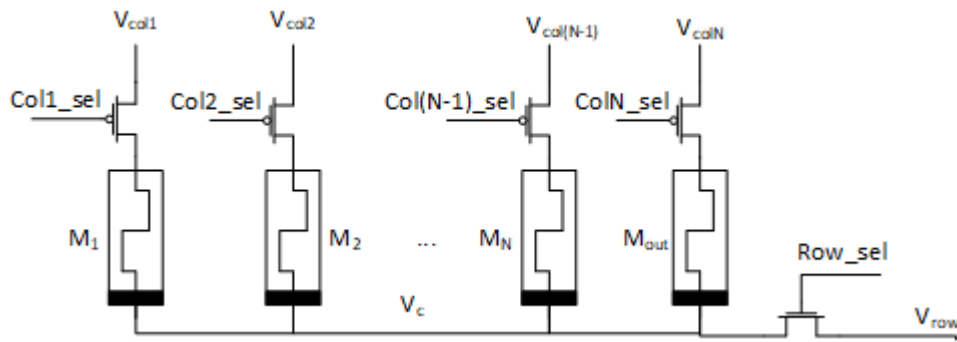


Figure 3.26: Circuit implementation of a N-input MAGIC NOR gate

From Fig.3.25, it should be noted that the MAGIC NOT gate is a special case of a N-input NOR gate, where  $N=1$ . Therefore, the following analysis will focus on the general case of a N-input MAGIC NOR gate. Conclusions for the NOT gate can be derived from the calculations for the NOR gate, by setting  $N=1$ .

As mentioned before, the operation of a MAGIC gate consists of two stages: the programming of the memristors and the evaluation of the boolean operation. During the programming stage, the row is selected and connected to ground. The columns to be used are also selected and its memristors are programmed accordingly. The output memristor is always programmed to '1'. In the evaluation stage, the row is deselected, the input memristors' columns' voltages are all set to  $V_o$  and the output memristor's column is connected to ground. The resulting circuit is the one shown in Fig.3.25(b), with the switch closed and the voltage  $V_c$  is now the output of a voltage divider formed by the parallel of the  $M_{ks}$ ,  $M_{out}$  and  $V_o$ :

$$V_c = \frac{M_{out}}{M_{out} + M_1 \parallel M_2 \parallel \dots \parallel M_N} V_{ev} \quad (3.22)$$

If all the inputs are '0',  $M_k = R_{off}, \forall k$  and the resulting state in  $M_{out}$  should be '1'. Since that memristor is initialized to '1', the voltage  $-V_c$ , should be greater than  $V_{reset}$ :

$$-V_c = -\frac{R_{on}}{R_{on} + R_{off}/N} V_o > V_{reset} \quad (3.23)$$

Solving Eq.(3.23) with respect to  $V_o$ :

$$V_{ev} < -(1 + \frac{R_{off}}{NR_{on}}) V_{reset} \quad (3.24)$$

As a simplification on Eq.(3.24), when the number of inputs is small, the following lower bound can be applied:

$$V_{ev} < -\frac{R_{off}}{NR_{on}} V_{reset} \quad (3.25)$$

For all other input combinations  $-V_c$  should be lower than  $V_{reset}$ . The most extreme case is when one input is '1' and all others are '0'. Therefore the following condition must be satisfied:

$$-V_c = -\frac{R_{on}}{R_{on} + R_{on} \parallel \frac{R_{off}}{N-1}} V_{ev} < V_{reset} \quad (3.26)$$

Solving Eq.3.26 with respect to  $V_o$ :

$$V_{ev} > -(1 + \frac{R_{on} \parallel \frac{R_{off}}{N-1}}{R_{on}}) V_{reset} \quad (3.27)$$

Eq.(3.27) can be simplified by taking the maximum value as shown in Eq.(3.28), by ignoring the  $\frac{R_{off}}{N-1}$  resistance in parallel with  $R_{on}$ , for small enough N:

$$V_{ev} > -2V_{reset} \quad (3.28)$$

However, it can also be desirable to have a non-destructive computation, i.e, the computation does not change the state of the input memristors. Assuming positive  $V_o$ , this means that:

$$V_{ev} - V_c < V_{set} \quad (3.29)$$

Worst case comes when all inputs are 0 and the voltage across the memristors is at its maximum value. It then follows that:

$$\frac{R_{off}/N}{R_{on} + R_{off}/N} V_{ev} < V_{set} \quad (3.30)$$

And solving with respect to  $V_o$  results in:

$$V_{ev} < (1 + N \frac{R_{on}}{R_{off}}) V_{set} \quad (3.31)$$

In short, to ensure the correct functioning of a N-input MAGIC NOR gate, when the number of inputs is small, the condition shown in Eq.(3.32) must be met:

$$-2V_{reset} < V_{ev} < -\frac{R_{off}}{NR_{on}} V_{reset} \quad (3.32)$$

For a large value of N, the approximations taken to reach this condition are less accurate, so Eqs.(3.24) and (3.27) should be verified:

$$-(1 + \frac{R_{on} \parallel \frac{R_{off}}{N-1}}{R_{on}}) V_{reset} < V_{ev} < -(1 + \frac{R_{off}}{NR_{on}}) V_{reset} \quad (3.33)$$

For a non-destructive computation, Eq.(3.31) should also be met.

Additionally, in [38], the use of a sensing amplifier is proposed for the reading of the memristors' states. The circuit used in this work is shown in Fig.3.27. This circuit is used assuming an overall computing system such as [40], where a control unit selects, through decoders, what is connected to each column and row. In particular, the row decoder is able to connect a sensing resistor to the corresponding row. Following this perspective, the transistors connected to  $V_c$  are considered to be parts of the same decoder.

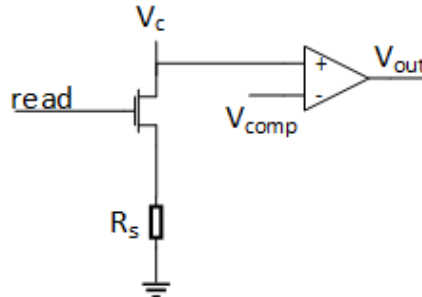


Figure 3.27: Sensing amplifier for MAGIC circuits

To read a memristors state, its column is selected and a reading voltage is applied to it. Also, the sensing resistor is connected to the row, thus forming a voltage divider. The output of this voltage divider is compared to a value between the voltage obtained from a '1' and a '0' state memristor.

The complete circuit for a single row of the memristive crossbar is depicted in Fig.3.28.

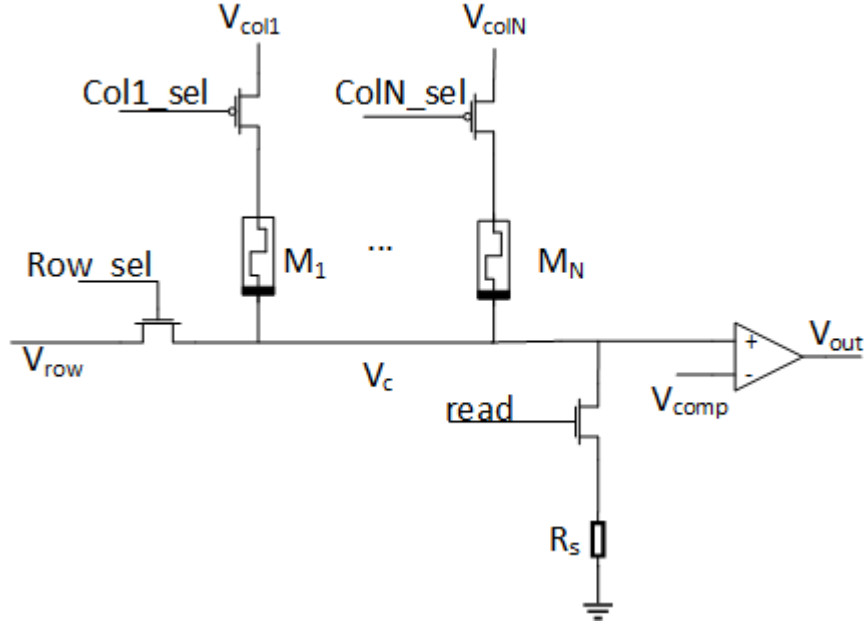


Figure 3.28: Complete MAGIC crossbar row circuit

### 3.6 Final Remarks

This chapter presented some strategies to design and implement a logic gate using memristors. It provided some insight as to how such gates function, not only by intuitive reasoning but also resorting to simple math and circuit theory. It was shown that most of these strategies are quite different from what is done in the traditional CMOS paradigm.



## Chapter 4

# Circuit Simulations

This chapter presents simulations done on memristive circuits from some of the logic families presented in the previous chapter. The intent behind these simulations is to extract some metrics to evaluate the performance of the logic gates of each of those families.

### 4.1 Metrics

Before running simulations on these circuits, some metrics must be defined in order to evaluate the performance of these logic families. Those metrics are hereby defined as:

- Read Time ( $t_{read}(V_{read})$ ) := maximum time available for the reading of the output,  $V_{out}$ , for a given reading voltage,  $V_{read}$ , of the circuit before its value is lost, assuming that input writing and evaluation are done correctly;
- Evaluation Time ( $t_{ev}(V_{ev})$ ) := minimum time necessary for the correct evaluation of the logic operation, for a given evaluation voltage  $V_{ev}$ , assuming that input writing is done correctly;
- Write Time ( $t_{wr}(V_{wr})$ ) := minimum time required for the writing of one input, for a given  $V_{wr}$ ;
- Reading Stage Dissipated Power ( $P_{read}(V_{read}, t_{read})$ ) := average power dissipated during the evaluation stage of the operation, for a given  $V_{read}$  and its corresponding  $t_{read}$ .
- Writing Stage Dissipated Power ( $P_{wr}(V_{wr}, t_{wr})$ ) := average power dissipated during the writing stage of the operation, for a given  $V_{wr}$  and its corresponding  $t_{wr}$ ;
- Evaluation Stage Dissipated Power ( $P_{ev}(V_{ev}, t_{ev})$ ) := average power dissipated during the evaluation stage of the operation, for a given  $V_{ev}$  and its corresponding  $t_{ev}$ ;
- Noise Margin ( $NM(V)$ ) := difference between the output voltages before the comparison circuit, if any, when the output is supposed to be interpreted as '1' and '0' [41, 42]. This depends on the resistance of the memristors involved in the output reading which, in turn, depends on the voltage used to write their states;

## 4.2 Simulation Circuits

The following section presents the circuits that were simulated in order to measure the metrics presented above. The logic families that were simulated were: MRL, CMOS-like, Parallel Input Processing and MAGIC. The circuits used are the ones presented in the Figs.4.1 through 4.4.

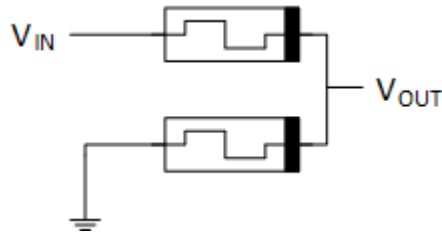


Figure 4.1: Circuit used to simulate an MRL OR gate

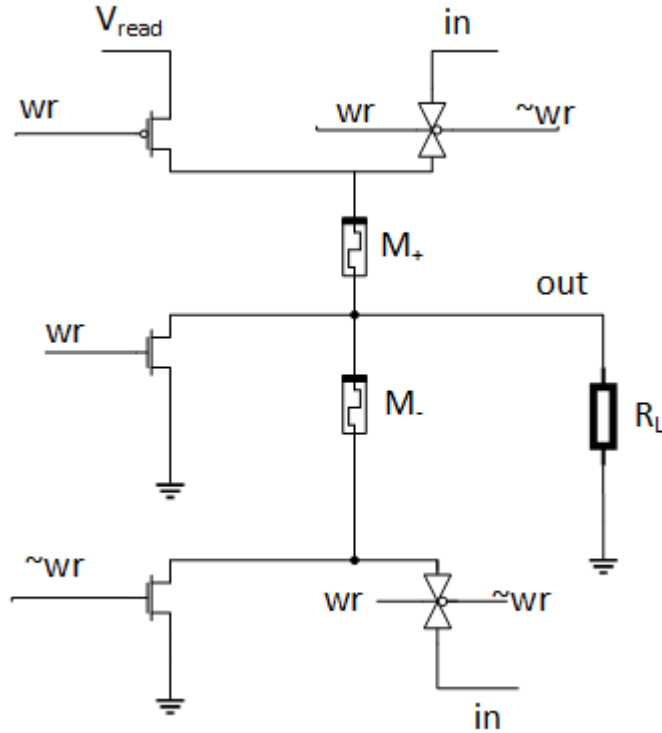


Figure 4.2: Circuit used to simulate a CMOS-like memristive NOT gate

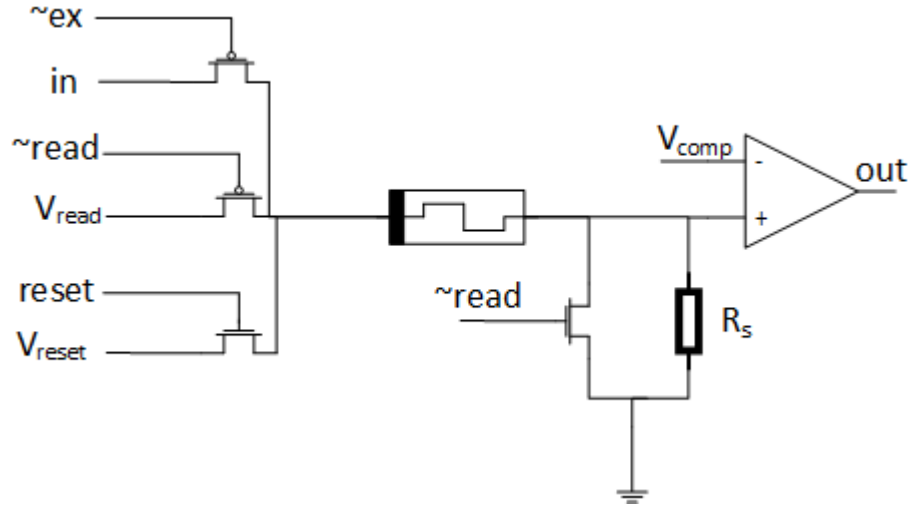


Figure 4.3: Circuit used to simulate a Parallel Input Processing NOT gate

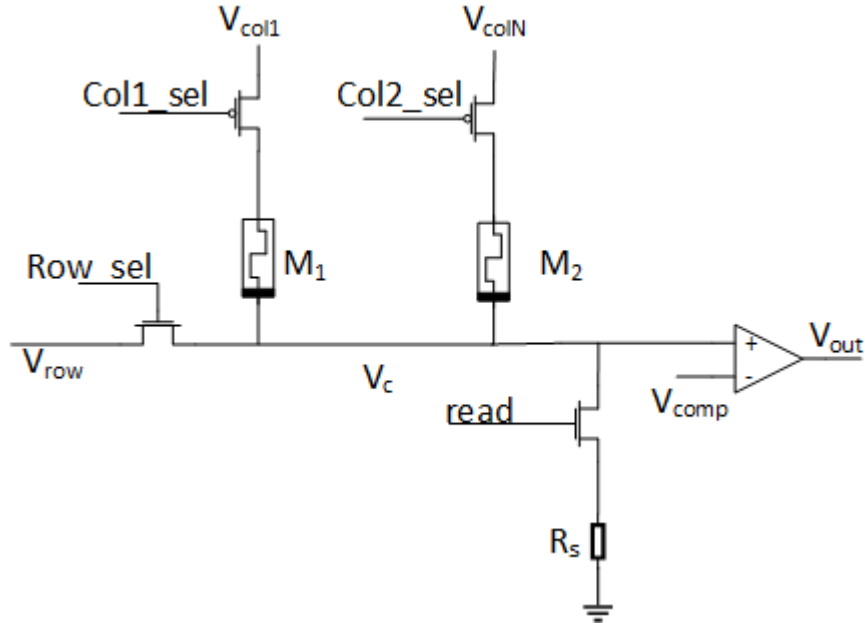


Figure 4.4: Circuit used to simulate a MAGIC NOT gate

In all of the circuits shown above, the bulk of all NMOS is assumed to be connected to  $V_{DD} = 1.65V$  and the bulk of all PMOS is connected to  $V_{SS} = -1.65V$ , and all control signals assume logic '0' as  $V_{SS}$  and logic '1' as  $V_{DD}$ , except for the MRL case where logic '0' is represented as 0V. These transistors' dimensions were set following the condition  $\frac{W}{L} > \frac{1}{0.1k/R_{on}}(V_{gs} - V_t)$ . This ensures a low voltage drop across the transistors.

When a sensing resistor is required, the value of this resistor is the geometric mean of the '0' state resistance and the '1' state resistance  $R_s = \sqrt{R_{off}R_{on}} \approx 2k\Omega$ . This value was chosen

because it maximizes the noise margin of the circuit [41].

In the case of the MRL gate, since all gates from this family have at least two inputs, one of the inputs is connected to ground.

## 4.3 Simulation results

The next section presents and discusses the results obtained in the extraction of the metrics previously presented.

### 4.3.1 Read Time

As previously defined, the read time of a circuit is the time slot available for reading the gate's output. This was measured as the minimum time obtained for different inputs since the start of the reading stage until the output changes to a wrong value. In each of the simulations, a parametric analysis was made, varying the reading voltage of the circuit from 350 mV to 1.65V with a step of 50 mV. Since the MRL has no reading stage, and the output does not change after it stabilizes for a given input, no reading time was measured for this logic family. The results of these measurements are depicted in Fig.4.5.

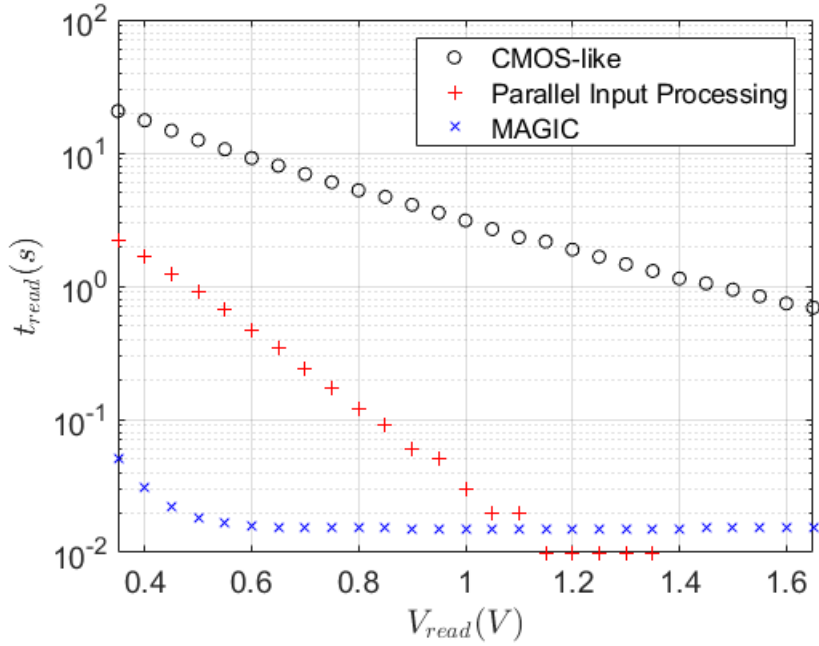


Figure 4.5: Results obtained for the measurements of  $t_{read}$

As can be seen in the figure above, the CMOS-like has the best reading time, followed by the Parallel Input Processing logic gate and the MAGIC logic gate. This is due to the fact that, in a CMOS-like logic gate, there is always one memristor in the OFF state, making the current that passes through the memristors small, which reduces the effect of the reading operation on the overall state of the circuit. This does not happen in the other logic families.

For voltages less than 1V, the Parallel Input Processing logic gate has better reading time than the MAGIC. An explanation for this is that the evaluation of the MAGIC gate always brings the state of the output memristor closer to '0'. More specifically, when the input is '0', the output memristor's state falls slightly to a state between '1' and '0', and when the input is '1', the output memristor's state falls to '0'. This means that when the output is read, the state of the memristor, when the input was '0', is closer to the '0' state, requiring less time to switch completely to '0'. Also, it can be observed that the evaluation time tends to some value after some amount of voltage is being applied. This happens because the voltage applied to the memristors is greater than its  $V_{set}$  or  $V_{reset}$ . This also happens when measuring the other time metrics, for the same reason.

### 4.3.2 Evaluation Time

The evaluation time of a logic gate is defined as the minimum time necessary for a logic gate to evaluate the intended logic operation. To measure this, the auxiliary XSV terminal mentioned in subsection 2.3.1 was used to measure the rise and fall time of the state variable of the memristor that stores the result of the operation during the evaluation stage, for the cases of the Parallel Input Processing logic family and the MAGIC family. For the MRL case, the rise and fall time of the output signal was measured instead. The evaluation time is taken to be the maximum of these two times, because when this maximum time is reached the inputs has been processed independently of its values, and was measured in a parametric analysis where the voltage applied during the evaluation phase,  $V_{ex}$ , varied from 350 mV to 1.65V with a step of 50 mV. Since the CMOS-like family's logic gates have no evaluation stage, this metric does not apply. The results obtained can be consulted in Fig.4.6.

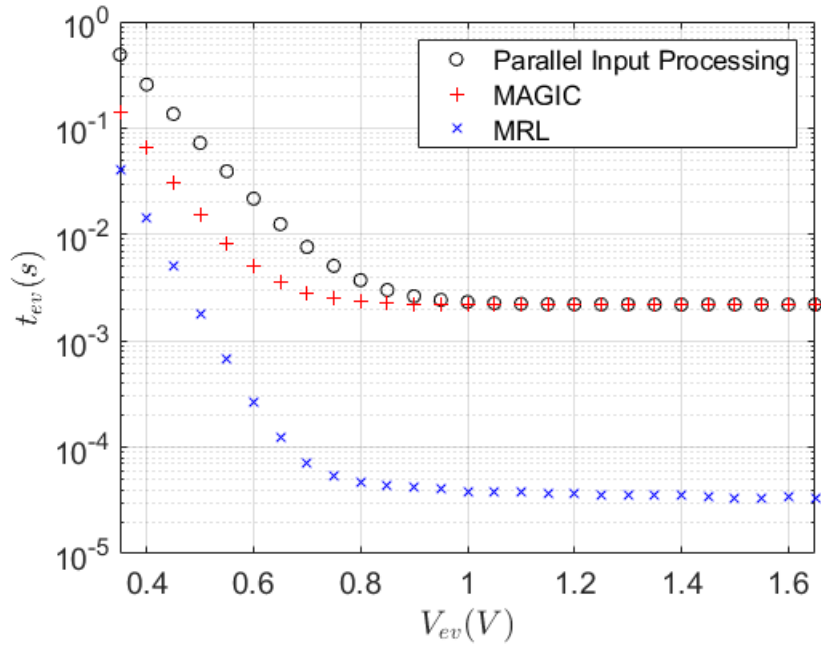


Figure 4.6: Results obtained for the measurements of  $t_{ev}$

From the previous figure it can be observed that the MRL gates possess the best evaluation time. This is because the input is applied directly to the memristors, that is, there are no pass transistors to cause a voltage drop in the input signal.

### 4.3.3 Writing Time

As stated before, the writing time of a logic gate is defined as the minimum duration of the writing stage required to properly write the inputs, or initialize them, to ensure a proper computation. Since the MRL logic gates require no such writing or initialization, this metric does not apply. This was measured again as the maximum of the rise and fall times of the state variable of the memristors to be written into, since for this time the inputs have been written regardless of its value, assuming the memristors are initialized to the opposite state, by varying the input or reset voltage from 350 mV to 1.65V with a step of 50 mV. The resulting data can be observed in Fig.4.7.

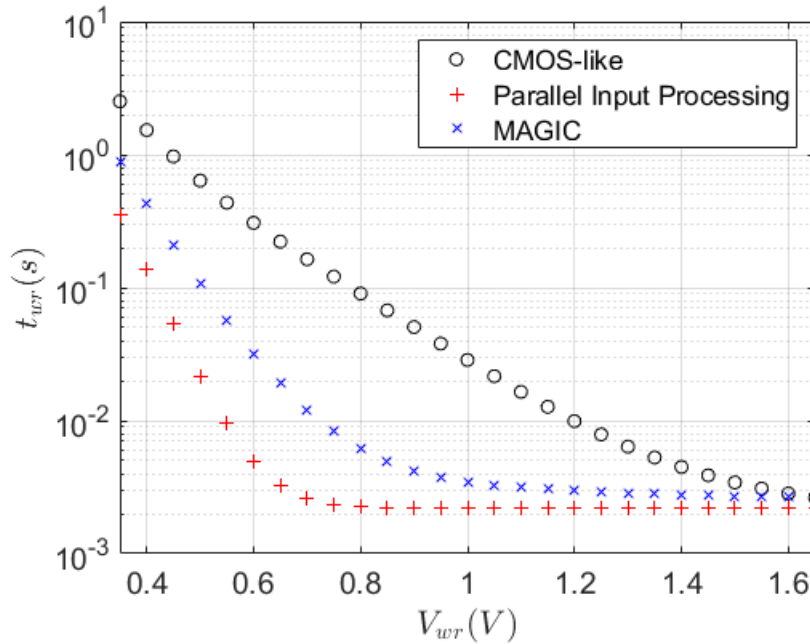


Figure 4.7: Results obtained for the measurements of  $t_{wr}$

From Fig.4.7, it can be concluded that the Parallel Input Processing logic family has the best writing time. This can be justified by the fact that this logic family requires fewer memristors than the other two. This is always true regardless of the number of inputs of the logic gate, at least for the case of a NOR logic gate, since it only requires one memristor for any number of inputs  $N$ , while the MAGIC NOR gates requires  $N+1$  memristors and the CMOS-like NOR requires  $2N$  memristors. Additionally, for the CMOS-like case, the inputs must be written one at a time, thus increasing the writing time even further.

#### 4.3.4 Reading Stage Dissipated Power

The Reading Stage Dissipated Power is, as described above, the average power dissipated during the reading stage for a given reading voltage. As for the duration of the stage, it is made to be the reading time previously measured. This was also done when measuring the power dissipated in the other phases of the circuits' operations, and was done like this because it ensures that the operation is done correctly. Additionally, it allows one to take conclusions about the advantage of using a smaller voltage during a longer time or of using a larger voltage in a small period of time.

Since  $P = \frac{V^2}{R}$ , simply varying the voltage for the same amount of time would not allow for an interesting conclusion, because, unless the memristor's state goes to '0', in which case the information would be lost, the dissipated power will always increase with the voltage.

Here, the parametric analysis on the reading voltage ran from 350 mV to 1.35 V with a step of 50 mV.

The graphs resulting from this simulation are presented in Fig.4.8.

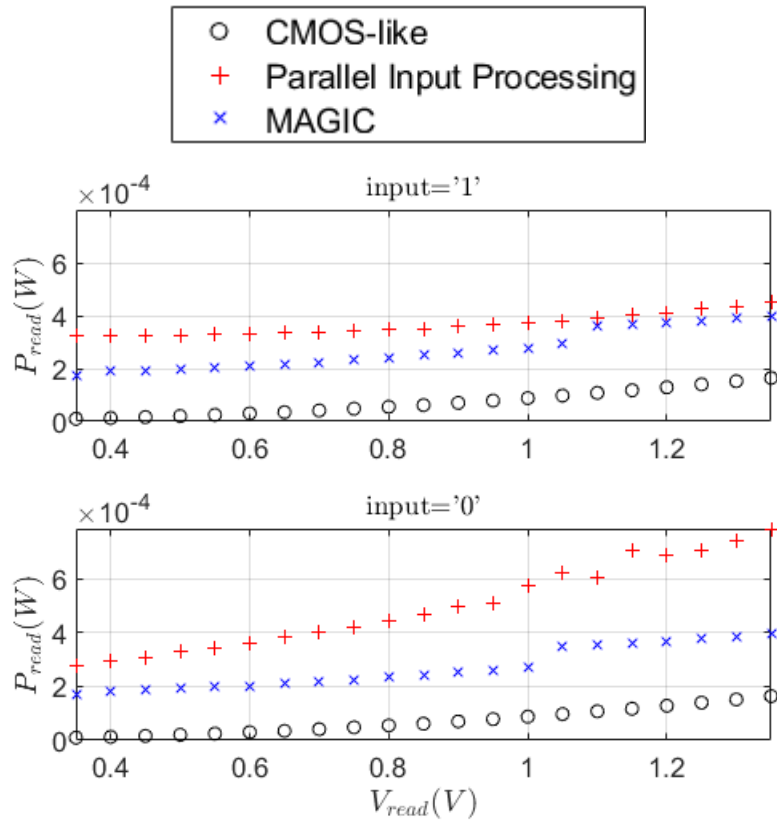


Figure 4.8: Results obtained for the measurements of  $P_{read}$ .

As shown in the figures above, the CMOS-like family dissipates less power during the reading stage. This is due to the fact that, in the reading phase, the circuit always has one memristor with high resistance, reducing the current supplied to the circuit thus reducing the power dissipated by the circuit.

The MAGIC logic NOT gate dissipates roughly the same power in the reading stage for both inputs. This is due to the fact that, in the previous stage, the memristor's state approaches the '0' state, as explained in subsection 4.3.1.

For the Parallel Input Processing case, when the input is '0' the resulting memristor's state is '1' and vice-versa. Because of this, the circuit dissipates more power when the input is '0', since the resistance is less than when the input is '1'.

#### 4.3.5 Evaluation Stage Dissipated Power

The Evaluation Stage Dissipated Power was defined as the power dissipated during the evaluation stage. This was measured by varying the evaluation voltage from 350 mV to 1.65V with a step of 50 mV and setting the evaluation time to be the one presented in subsection 4.3.2 corresponding to each voltage used. In the case of the Parallel Input Processing, the input actually means that the memristor holds the complementary bit, i.e, if the input is '1' the memristor is holding a '0' and vice-versa, and the memristor is being reseted to the '1' state. The resulting graphs are presented in Figs.4.9.

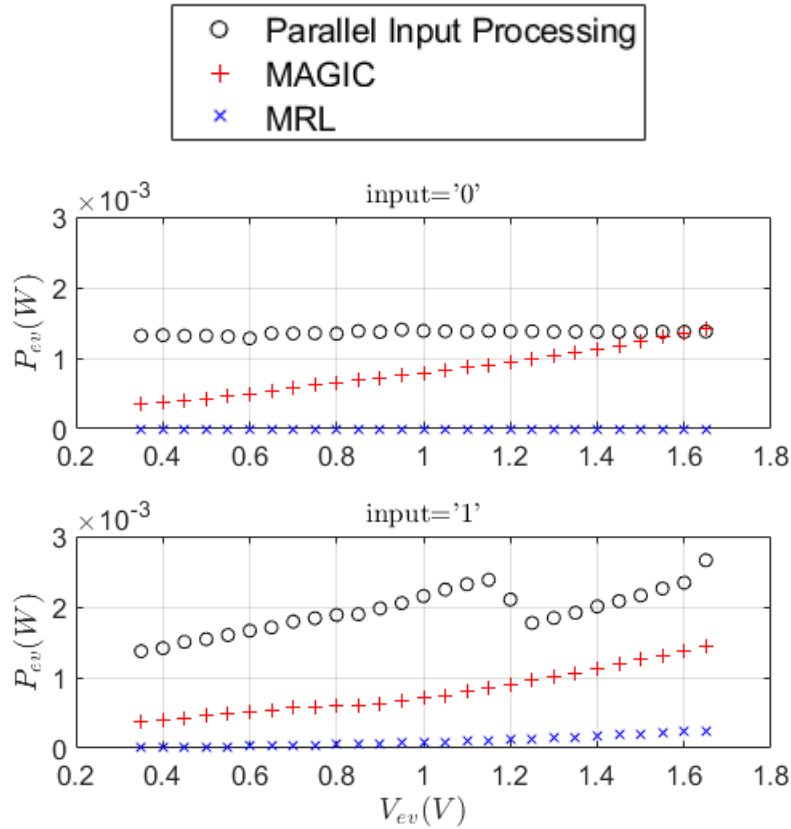


Figure 4.9: Results obtained for the measurements of  $P_{ev}$

From Fig.4.9, it may be inferred that the MRL logic family dissipates the least amount of power during the evaluation stage. The reason for this is that, when the inputs of the gate are different, the memristors' states change in such a way that there is always one memristor



that presents a high resistance path to the current flow, thus reducing the amount of power dissipated. On the other hand, the Parallel Input Processing logic family dissipates the most power, because current passes through only one memristor, whereas in the case of a MAGIC logic gate, it passes through two, reducing the amount of current supplied by the voltage source, hence reducing the amount of power dissipated by the overall circuit.

#### 4.3.6 Writing Stage Dissipated Power

As described, the Writing Stage Dissipated Power is the power dissipated during the writing stage. This was measured by varying the writing voltage from 350 mV to 1.65V with a step of 50 mV and setting the writing time to be the writing time shown in subsection 4.3.3 corresponding to each voltage used. In the case of the Parallel Input Processing, the input actually means that the memristor holds the complementary bit, i.e, if the input is '1' the memristor is holding a '0' and vice-versa, and the memristor is being reseted to the '1' state. The resulting graphs are presented in Figs.4.10.

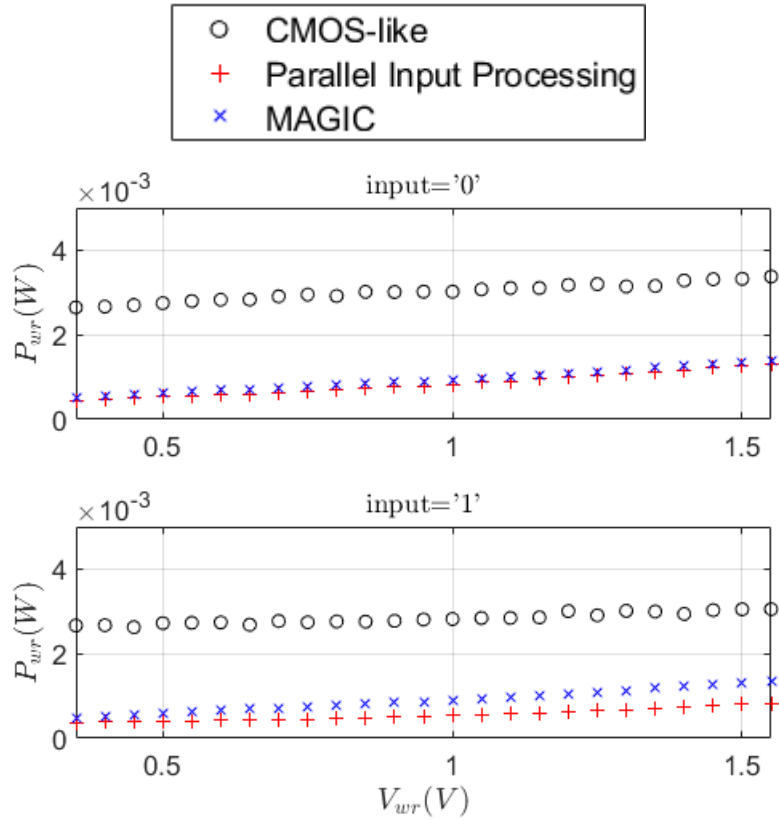


Figure 4.10: Results obtained for the measurements of  $P_{wr}$

By observing the figures above, it can be concluded that the CMOS-like family dissipates more power than the other families. This may be justified by the fact that two complementary memristors are being written into, meaning that the writing voltage is always being applied to one low resistance memristor, which dissipates more power.

### 4.3.7 Noise Margin

Noise Margin of a memristive logic circuit has been defined as the difference between the output voltage of the voltage divider when the output should be 1 and 0. This value is normalized by dividing by the reading voltage used. This normalization expresses the noise margin as a fraction of the reading voltage which may be expressed as a percentage, and it allows for a better comparison between logic families. A pictorial interpretation of the noise margin is depicted in Fig.4.11.

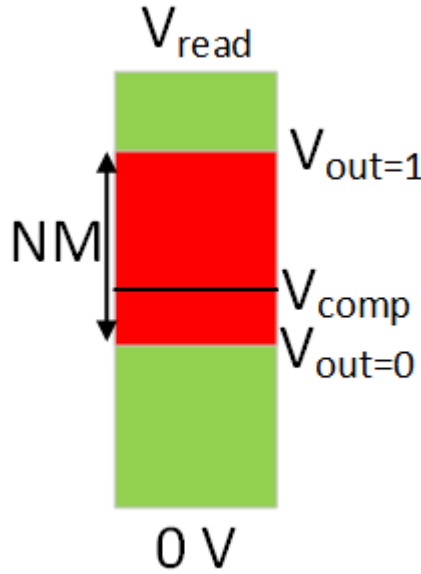


Figure 4.11: Representation of the Noise Margin

As shown in the figure above, the noise margin represents the length of the region of voltages bounded by the output voltages obtained from the voltage divider when the output is '1' and '0'. To convert the output of the voltage divider into a digital quantity, a comparator is used to compare that voltage with  $V_{comp}$ , which should be inside the region previously mentioned. In this perspective, the noise margin gives information as to how much noise is allowed so that an output of '1' is not misinterpreted as '0', and vice-versa. Intuitively, it would make the most sense to place  $V_{comp}$  at the mean of the region. However, it could prove advantageous to place it somewhere else, for example, if the dynamic of the state variable is different when decreasing and increasing. For this reason, this definition of the noise margin does not include a dependence on  $V_{comp}$ .

The noise margin depends on the state of the memristors involved in the reading operation, which, in turn, depends on the voltage and time used in the evaluation phase, or the writing phase if the logic family does not assume an evaluation stage. So, when comparing the noise margins of different logic families, the time the evaluation (writing) voltage is applied should be the same, and a parametric analysis is done on the voltage. Hence, the time used to evaluate the logic operation (write the inputs) in this case was 5 ms, and the voltage was varied from 350 mV to 1.65 V with a step of 50 mV. The results of this simulation can be

consulted in Fig.4.12.

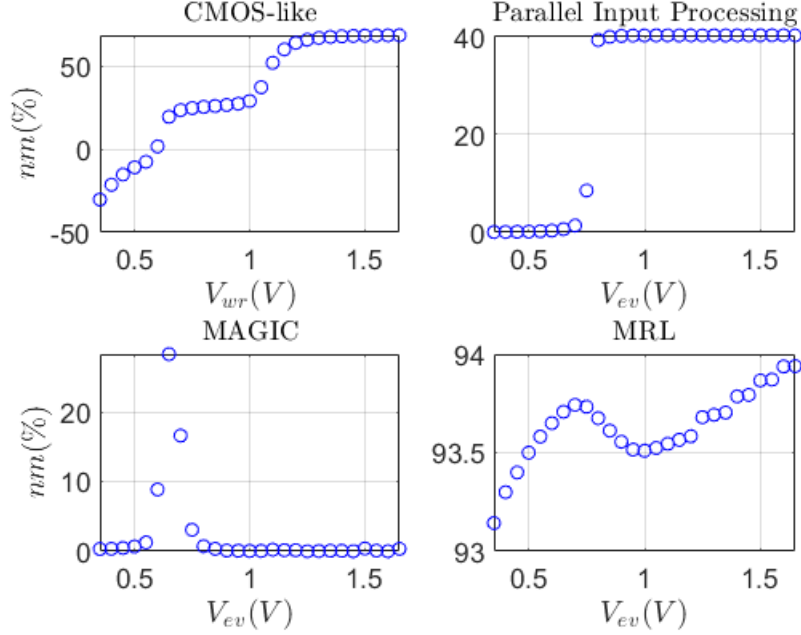


Figure 4.12: Results obtained for the measurements of  $NM$

As can be seen in Fig.4.12, the MRL logic family has the greatest noise margin out of the four. This is because the inputs force the states of the memristors in such a way that the memristor with the lowest resistance is the one where the highest voltage is injected, placing this voltage at the output with very little voltage drop. Additionally, when both the inputs are '0', there is no current flow, meaning that the '0' input appears at the output with no voltage drop at all. Similarly, the CMOS-like logic family has the next best noise margin, at least for high enough voltages, because both memristors involved hold complementary states.

It is also noticeable that, for low writing voltages, the CMOS-like logic family actually presents negative noise margin (by virtue of the definition). This means that the writing voltage did not switch the memristors enough so that the output becomes what was intended, and the operation was unsuccessful. Similarly, for the Parallel Input Processing case, the noise margin is near zero for low voltages, until a certain voltage is achieved. This is because only when that voltage value is applied at the input of the logic gate the voltage drop at the memristor's terminals is greater than  $V_{set}$ , after taking the voltage drops across the transistors into account.

Finally, for the MAGIC logic family case, there is a noticeable noise margin only when for a narrow region of evaluation voltage values. Since the output memristor is initialized to '1' before evaluation, there is a somewhat narrow region of voltages that actually allows for distinction between voltage levels. Voltages lower than this region result in no switching of the memristor, and its state stays at '1' independently of the input. Voltages higher than this region, always switch the memristor to '0' no matter the input.

It should be kept in mind that, this analysis was made for a specific value of  $t_{ev}$ . The voltage values mentioned in all cases above depend on  $t_{ev}$ , and changing this evaluation time would

change the value of the voltages mentioned. However, even though the values change, the overall behavior of the curves depicted is similar, for different values of  $t_{ev}$ .

## 4.4 Final Remarks

This chapter defined some metrics that may be used to ascertain the performance of memristive logic circuits, and presented the procedures on how to measure them. Some of these metrics depend on observing the evolution of the state variable. While this is possible on a simulation environment, in a practical environment one does not have access to this state variable. One workaround for this problem would be to periodically check the value of the resistance of the memristor, which is equivalent to checking the state, since the resistance of the memristor is a bijective function of the state variable. This could possibly involve the use of a voltage divider with a fixed resistance, a low reading voltage as to not interfere with the state of the memristor and a low input offset voltage amplifier, like a chopper amplifier.

Also in this chapter, these metrics were evaluated for some logic families, and the results were discussed in order to better comprehend how these logic gates operate and the advantages and disadvantages of the usage of each of them:

- MRL: This logic family proved to be, by far, the fastest one of all the four considered, since it only assumes an evaluation stage, which is the fastest one of all. It also presented the least overall dissipated power and the best noise margin.

However, using only memristors, one can only realize two logic gates. Other digital circuits require the use of transistors and other components. While these logic gates are the fastest and more efficient, its non statefull, Out-of Memory computing nature make this logic family not suited for crossbar implementation. It may be used, however, to create hardware accelerators. For example, for circuits to realize the minimum and maximum operations [43] and other auxiliary circuits [24].

- CMOS-like: This logic family presented the best reading time of all the tested families. This may be preferred if the following circuit is slow and requires some time to read the output. It also presented the least reading stage dissipated power. These two characteristics make this logic family a good one if the output is to be read repeatedly.

Even though it does not have an evaluation stage, which is an advantageous feature for a small number of inputs, multiple inputs must be introduced one at a time, which may cause a significant delay.

It presented the worst performance in the writing stage. This shows that this family is not ideal if this logic gate is to be computed several times.

Finally, it presented considerable noise margin, which is a desirable feature, as it makes this family less prone to reading errors.

- Parallel Input Processing: This logic family presented many features similar to the MAGIC one, especially for higher voltages, namely, evaluation time, writing time, writing stage dissipated power but it presented a better reading time.

It also presented the worst evaluation stage dissipated power and a reading stage dissipated power worse than the CMOS-like family, however its writing stage dissipated

power is the lowest one. This can be further improved by evaluating the memristor's state before initializing it, since there is no need for initialization if the memristor is already in the desired state

The noise margin presented by this logic family is reasonable, at least if the evaluation voltage is high enough.

- MAGIC: This logic family presented the worst read time of all, meaning that the circuit that receives the voltage obtained from the voltage divider must be fast enough to read the bit before it is lost. It did, however, present a lower evaluation stage dissipated power than the Parallel Input Processing one and a writing time lower than the CMOS-like one.

Its noise margin is very low except for a somewhat narrow range of voltages. This means that the circuit has low tolerance for variations in the evaluation voltage.

Although there are many downsides to the usage of this logic family, the fact that it is a statefull logic family with In Memory computing characteristics makes this logic family the most easily realizable within a crossbar out of all the four.



## Chapter 5

# Conclusions

### 5.1 Work Done

The aim of this dissertation was to study how boolean logic functions may be implemented using memristors as the main circuit component and to compare different memristive logic circuits. To achieve this objective, in chapter 2, the first step taken was to understand the theory behind how memristors work, by studying the general equations that describe its behavior, the different classes a memristor may belong to, some properties and tools that help understand this device and how to operate it.

Afterwards, also in chapter 2, it was necessary to obtain a more specific mathematical model to represent the behavior of a memristor. For that, a review on existing mathematical models and their implementation in a SPICE-like simulator, such as Cadence, was conducted. Since the only commercially available memristor was provided by Knowm, the mathematical model that was chosen to be implemented in the simulator was the one proposed by Knowm which was shown to correlate to physical data obtained from measurements on their device. To obtain the parameters to be introduced into the model, the IV-characteristic of a memristor obtained from Knowm was measured.

In chapter 3, the implementation of logic gates using memristors was addressed. First, the switching dynamics of the memristor and compositions of memristors was reviewed, by assuming a simpler behavior than the one described by the model used in the simulations. This allowed for a more intuitive perspective on the functioning of the memristor and combinations thereof.

After learning about the switching dynamics of memristive compositions, some families of logic gates using memristors were presented. It was shown that there exist some properties that differ from one logic family to another, making these fall into three categories. These properties were referred to as proximity of computation and statefulness. For each one of these categories, two logic families were introduced and studied, by both introducing some intuition of how these logic gates operate and some circuit analysis, by using the simplified behavior previously referenced. This analysis brought to light some limitations concerning the values that the different voltages involved in the operation may take. It was also shown that some of the logic families require several stages to operate and that the stage is defined by

some control signals and, sometimes, by the values of the voltages applied to the memristors.

In chapter 4, some metrics were defined to evaluate the performance of different memristive logic families and how to measure them, at least in a simulation environment. Then, these metrics were extracted from some circuits of some logic families. The circuits were all single input gates, to facilitate the parametric analysis conducted, however some conclusions were drawn for multiple input logic gates whenever possible. These measurements led to the realization that some families are more suited for certain applications than others:

- CMOS-like: This logic family proved to be the most advantageous in the reading stage, with the best reading time and dissipated power and with decent noise margin. These characteristics may be good if the load this circuit is driving is slow. However, in the writing stage, it performed the worst of all logic families, with the highest writing time and dissipated power;
- Parallel Input Processing: This logic family performed similarly to the MAGIC one, however it presented a better reading time, better noise margin and the best writing stage dissipated power;
- MAGIC: This logic family presented a very short read time, so the reading circuit must be fast enough to read the output before it is lost. It was also observed that the noise margin is very low except for a range of voltages. This indicates the need for a very precise voltage supply for the evaluation voltage. Also, because this is a statefull logic family and the NOR gate provided has a repeatable topology, an is functionally complete, it is the logic family most easily implemented in a crossbar array. This makes this logic family a good candidate for logic-in-memory applications;
- MRL: This logic family showed the best overall performance, since it only assumes one operation phase, and in that phase presented the best performance. It also presented the best noise margin of all the tested logic families. However, unlike the MAGIC logic family, it does not provide a complete set of boolean functions, unless aided by CMOS-components, and it is not statefull. This makes this logic family unsuited for crossbar implementation. Still, given its good performance, it could be usefull when implementing hardware acceleration circuits and specific digital circuits.

## 5.2 Future Work

In an attempt to carry on the work done thus far, some suggestions for future work are presented bellow:

1. Study of other memristive logic families: the simulations presented in this work left out many logic families that may be worth studying, not only some presented in this document but also some that were not mentioned [22, 44];
2. Simulations on memristive logic circuits using different models: the simulations that were done on this dissertation were made with a model and parameters that represent the behavior of a specific device. It could prove insightful to run these simulations using a different model representing the behavior of different devices;
3. Better extraction of the model parameters: the way the model parameters were ex-



tracted from the measured data was not ideal. This could be improved upon by using methods like nonlinear least squares fitting. This might prove challenging since it involves a state equation;

4. Study on different methods to read the output of a memristive logic circuit: the approach used in this work to read the output of a logic gate, in particular when the output is represented as the state of a single memristor may not be ideal, because simply applying a DC voltage to a voltage divider changes the value of the memristor's state variable. A solution to this problem may be desirable. One way to solve this problem might be to use a bipolar signal to read the state and rectify the output of the voltage divider [41]. This way of reading the state of the memristor should guarantee that the state variable is within a certain range, however, this could introduce some delay;
5. Experimental verification: the circuits presented in this work were only subject to simulation. A more practical approach to this study could be interesting. As of now, Knowm sells, not only discrete memristor chips, but also memristor crossbar arrays and PCI-E breakout boards to interact with these arrays. This, together with some microcontroller like Arduino or PIC, might facilitate this practical study of memristive logic families. Some metrics proposed rely on the fact that, in a simulation environment, it is possible to read the value of the state variable of the memristor in question, which is not true in a practical situation. Using a microcontroller makes it possible to read the resistance of the memristor periodically, bypassing this problem;
6. Study and comparison of different ways to program a memristor: many ways to program a memristor have been proposed [26–28, 30], however no work has been done to try to compare these different methods. It could prove useful to ascertain what methods to program a memristor are best for different applications. For example, since the memristor can be used as an analog memory element, if, instead of using a memristor as a two state device it were to be used as a four state device, its state would then represent two bits instead of one, allowing for a denser memory element. This would require a more accurate programming of the memristor.
7. Study between memristive sequential circuits: some sequential circuits using memristors have been proposed [45–47]. Since these are considered to be very important digital circuit elements, it could be interesting to compare the performance between the approaches suggested and, perhaps, improve them.



# Appendices

## Appendix A

# LTSpice Netlist Codes and Simulation Schematics

```
* HP Memristor SPICE Model Using Joglekar Window

* Connections:
* TE: Top electrode
* BE: Bottom electrode
* XSV: External connection to plot state variable
*      that is not used otherwise

.SUBCKT MEM_JOGLEKAR TE BE XSV

* Ron: Minimum device resistance
* Roff: Maximum device resistance
* D: Width of the thin film
* uv: Dopant mobility
* p: Parameter for window function
* x0: State variable initial value

.PARAMS Ron=100 Roff=10K x0=.56 D=12N uv=50F p=7

* Joglekar Window Function
.func f(V1) = 1-pow((2*V1-1),(2*p))

* Memristor I-V Relationship
.func IVRel(V1,V2) = V1/(Ron*V2 + Roff*(1-V2))

* Circuit to determine state variable
Gx 0 XSV value={ I(Gmem)*Ron*uv*f(V(XSV,0))/pow(D,2) }
Cx XSV 0 {1}
.ic V(XSV) = x0

* Current source representing memristor
Gmem TE BE value={IVRel(V(TE,BE),V(XSV,0))}

.ENDS MEM_JOGLEKAR
```

Figure A.1: HP Labs model with Joglekar Window Netlist

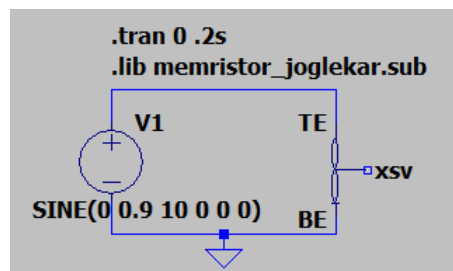


Figure A.2: Simulation schematic of HP Labs model with Joglekar Window

```

* HP Memristor SPICE Model Using Biolek Window

* Connections:
* TE: Top electrode
* BE: Bottom electrode
* XSV: External connection to plot state variable
*      that is not used otherwise

.SUBCKT MEM_BIOLEK TE BE XSV

* Ron: Minimum device resistance
* Roff: Maximum device resistance
* D: Width of the thin film
* uv: Dopant mobility
* p: Parameter for window function
* x0: State variable initial value

.PARAMS Ron=100 Roff=1K x0=.076 D=16N uv=40F p=7

* Biolek Window Function
.func f(V1,I1)={1-pow((V1-stp(-I1)),(2*p))}

* Memristor I-V Relationship
.func IVRel(V1,V2) = V1/(Ron*V2 + Roff*(1-V2))

* Circuit to determine state variable
Gx 0 XSV value={I(Gmem)*Ron*uv*f(V(XSV,0),I(Gmem))/pow(D,2)}
Cx XSV 0 {1}
.ic V(XSV) = x0

* Current source representing memristor
Gmem TE BE value={IVRel(V(TE,BE),V(XSV,0))}

.ENDS MEM_BIOLEK

```

Figure A.3: HP Labs model with Biolek Window Netlist

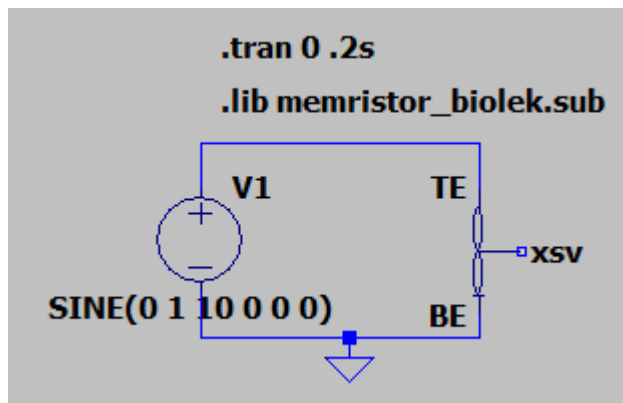


Figure A.4: Simulation schematic of HP Labs model with Biolek Window

```

.subckt MEM_LAIHO TE BE XSV
* Subcircuit Body
.param a1=4e-8 b1=1.2 a2=1.25e-7 b2=1.2 c1=6e-4 d1=2
+c2=6.6e-4 d2=3.8 x0=0.001 p=1

*IV relationship
.func IVRel(V1,V2) = IF(V1 >= 0, a1*V2*sinh(b1*V1),
+a2*V2*sinh(b2*V1))

*state equation
.func SV(V1) = IF(V1 >= 0, c1*sinh(d1*V1), c2*sinh(d2*V1))

*Biolek window function
.func f(V1,I1)={1-pow((V1-stp(-I1)),(2*p))}

*current source
Gmem TE BE value = {IVRel(V(TE,BE),V(XSV,0))}

*calculate state variable
Gxsv 0 XSV value = {SV(V(TE,BE))*f(V(XSV,0),I(Gmem))}
Cx XSV 0 {1}
.ic V(XSV) = x0

.ends MEM_LAIHO

```

Figure A.5: General Hyperbolic Sine model netlist

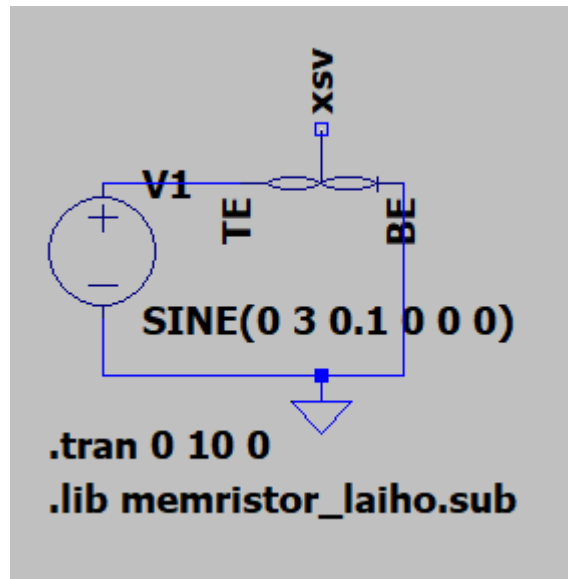


Figure A.6: Simulation Schematic of the General Hyperbolic Sine Model

```

* Memristor subcircuit developed by Chang et al.

.SUBCKT MEM_UMICH TE BE XSV

* Parameters:
* alpha: Prefactor for Schottky barrier
* beta: Exponent for Schottky barrier
* gamma: Prefactor for tunneling
* delta: Exponent for tunneling
* xmax: Maximum value of state variable
* xmin: Minimum value of state variable
* drift_bit: Binary value to switch the ionic drift in (1)
* or out (0) of the equation
* lambda: State variable multiplier
* etal, eta2: State variable exponential rates
* tau: Diffusion coefficient

.param alpha=0.5e-6 beta=0.5 gamma=4e-6 delta=2 xmax=1 xmin=0
+drift_bit = 0 lambda=4.5 etal=0.004 eta2=4 tau=10

.param cp={1}
Cpvar XSV 0 {cp}

* Rate equation for state variable
Gx 0 XSV value={ trunc(V(TE,BE),cp*V(XSV))*lambda*(etal*sinh(eta2*V(TE,BE))-
+drift_bit*cp*V(XSV)/tau) }

.ic V(XSV) = 0.0

* Auxiliary functions to limit the range of x
.func sign2(var) {(sgn(var)+1)/2}
.func trunc(var1,var2) {sign2(var1)*sign2(xmax-var2)+sign2(-
+var1)*sign2(var2-xmin)}

* Memristor IV Relationship
Gm TE BE value={{(1-cp*V(XSV))*alpha*(1-exp(-
+beta*V(TE,BE)))+(cp*V(XSV))*gamma*sinh(delta*V(TE,BE))}}

.ENDS MEM_UMICH

```

Figure A.7: U.Michigan model netlist

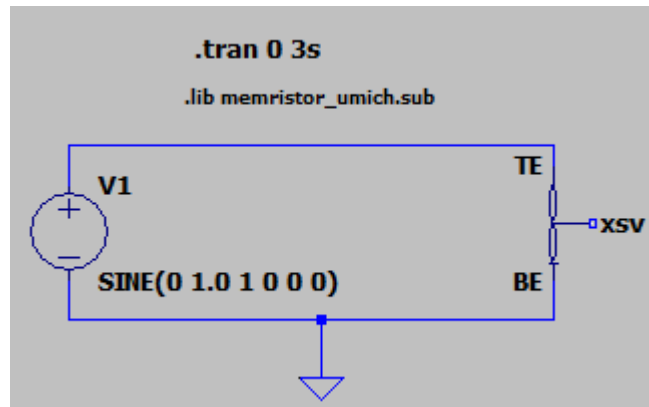


Figure A.8: Simulation schematic of the U.Michigan model

```

* SPICE model for memristive devices Created by Chris Yakopcic
|
.subckt MEH_YAKOPCIC TE BE XSV

* Fitting parameters to model different devices
* a1, a2, b: Parameters for IV relationship
* Vp, Vn: Pos. and neg. voltage thresholds
* Ap, An: Multiplier for SV motion intensity
* xp, xn: Points where SV motion is reduced
* alphap, alphan: Rate at which SV motion decays
* xo: Initial value of SV
* eta: SV direction relative to voltage

.params a1=-1.7 a2=.17 b=.05 Vp=.65 Vn=-0.56 Ap=4000
+An=4000 xp=0.3 xn=0.5 alphap=1 alphan=5 xo=0.11 eta=1

* Multiplicative functions to ensure zero state
* variable motion at memristor boundaries
.func wp(V) = xp/(1-xp) - V/(1-xp) + 1
.func wn(V) = V/(1-xn)

* Function G(V(t)) - Describes the device threshold
.func G(V) = IF(V <= Vp, IF(V >= -Vn, 0, -An*(exp(-
+V)-exp(Vn))), Ap*(exp(V)-exp(Vp)))

* Function F(V1,V2) - Describes the SV motion
.func F(V1,V2) = IF(eta*V1 >= 0, IF(V2 >= xp, exp(-
+alphap*(V2-xp))*wp(V2), 1), IF(V2 <= (1-xn),
+exp(alphan*(V2+xn-1))*wn(V2), 1))

* IV Response - Hyperbolic sine due to MIM structure
.func IVRel(V1,V2) = IF(V1 >= 0, a1*V2*sinh(b*V1),
+a2*V2*sinh(b*V1))

* Circuit to determine state variable
* dx/dt = F(V(t),x(t))*G(V(t))
C1 XSV 0 {1}
.ic V(XSV) = xo
G1 0 XSV value={eta*F(V(TE,BE),V(XSV,0))*G(V(TE,BE))}

* Current source for memristor IV response
Gm TE BE value = {IVRel(V(TE,BE),V(XSV,0))}

.ends MEH_YAKOPCIC

```

Figure A.9: Yakopcic model netlist

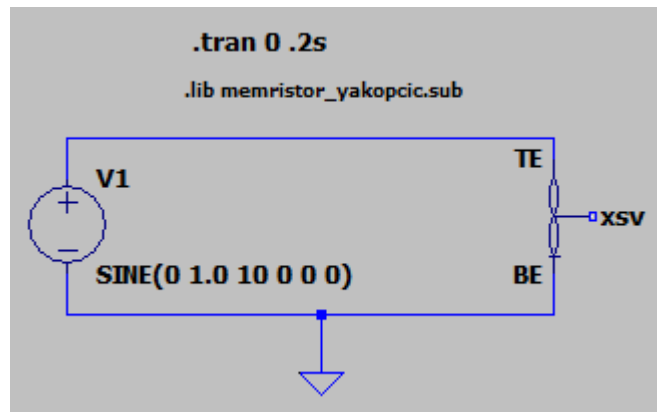


Figure A.10: Simulation schematic of Yakopcic's Model



```

* Known Mean Metastable Switch Memristor SPICE Model

* Copyright Tim Molter Known Inc. 2017

* Connections:
* TE: Top electrode
* BE: Bottom electrode
* XSV: External connection to plot state variable
*      that is not used otherwise

.SUBCKT MEM_KNOWNM TE BE XSV

* Ron: Minimum device resistance
* Roff: Maximum device resistance
* Von: Threshold voltage to turn device on
* Voff: Threshold voltage to turn device off
* TAU: Time constant
* T: Temperature

.PARAMS Ron=500 Roff=1500 Voff=0.27 Von=0.27 TAU=0.0001 T=298.5 x0=0

* Function G(V(t)) - Describes the device threshold
.func G(V) = V/Ron+(1-V)/Roff

* Function F(V(t),x(t)) - Describes the SV motion
.func F(V1,V2) = (1/TAU)*(( 1/(1+exp(-1/(T*boltz/echage)*(V1-Von))))*(1-V2)-
| | | | | 1/(1+exp(-1/(T*boltz/echage)*(V1-Voff)))))*V2

* Memristor I-V Relationship
.func IVR1(V1,V2) = V1*G(V2)

* Circuit to determine state variable
* dx/dt = F(V(t),x(t))*G(V(t))
Cx XSV 0 {1}
.ic V(XSV) = x0
Gx 0 XSV value={F(V(TE,BE),V(XSV,0))}

* Current source for memristor IV response
Gmem TE BE value={IVR1(V(TE,BE),V(XSV,0))}

.ENDS MEM_KNOWNM

```

Figure A.11: Mean Metastable Switch model netlist

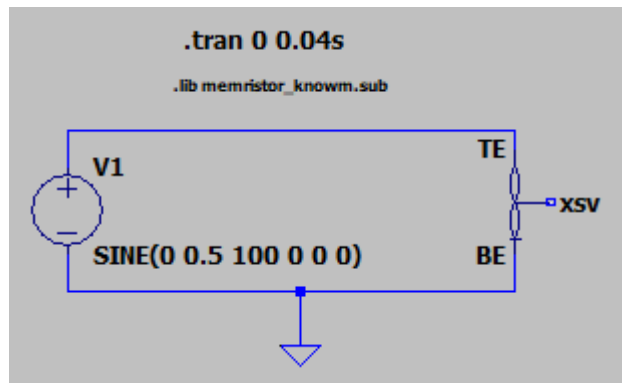


Figure A.12: Simulation schematic of the Mean Metastable Switch model

## Appendix B

### Remaining results obtained from the memristor's IV-curve measurements

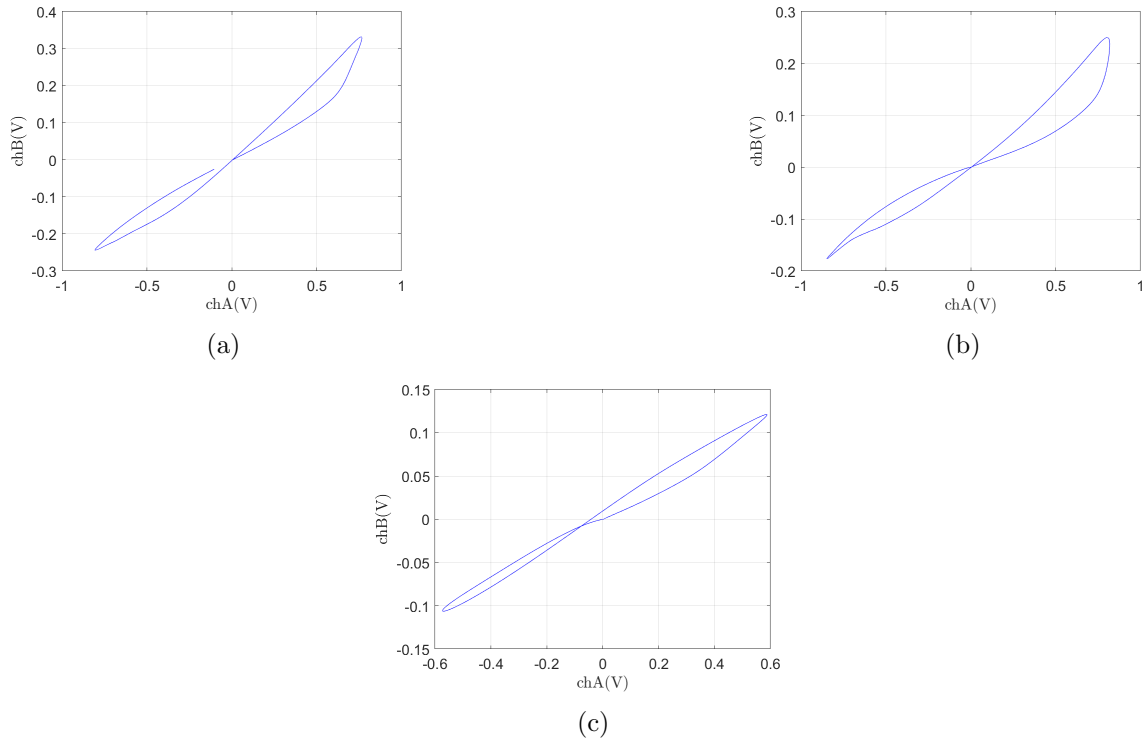
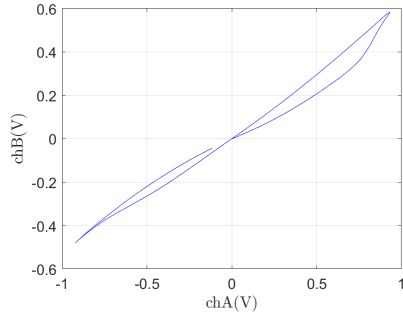
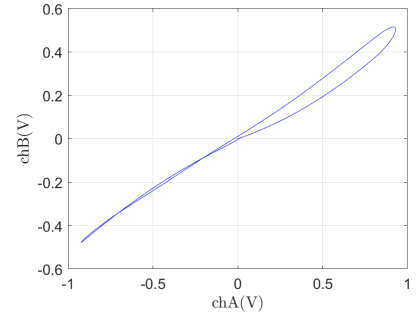


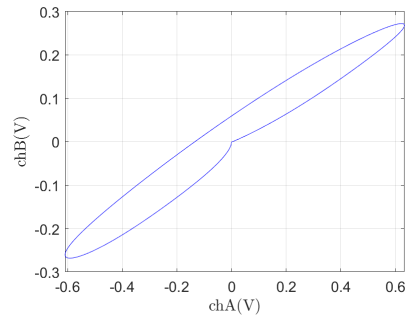
Figure B.1: IV-curves measured with  $R_L = 1k\Omega$ : (a) 10kHz, (b) 100kHz, (c) 1MHz



(a)

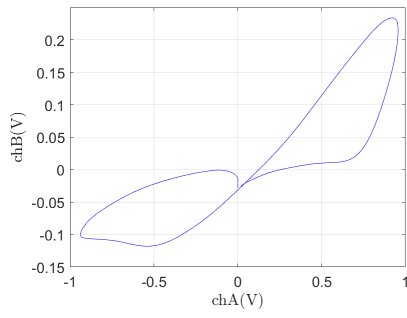


(b)

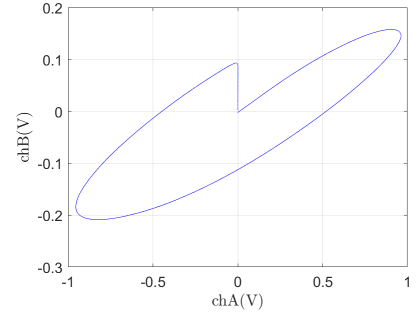


(c)

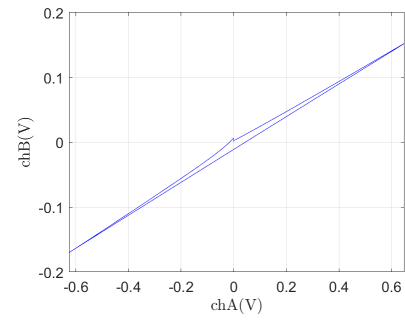
Figure B.2: IV-curves measured with  $R_L = 10k\Omega$ : (a) 10kHz, (b) 100kHz, (c) 1MHz



(a)



(b)



(c)

Figure B.3: IV-curves measured with  $R_L = 100k\Omega$ : (a) 10kHz, (b) 100kHz, (c) 1MHz

## Appendix C

# MATLAB code used to process .csv data from PicoScope

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%code used to process the measured data from PicoScope%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all
set(0,'defaultTextInterpreter','latex');
buf_size=32;      %size of the picoscope buffer.
                  %Same as number of cycles.

meanWindow=200; %size of moving mean window

Rdir=["r1k" "r10k" "r100k"]; %directories for each load

dirs=["f1a1" "f10a1" "f100a1" "f1ka1" "f10ka1" "f100ka1" "f1Ma1"];
%directories inside each load directory for each frequency

headers=["Frequency" "ChannelA" "ChannelB"];
%headers of the csv files

genPath="C:\Users\joao\Documents\MATLAB\tese\medidas pico\";
%path where all csv files are stored

for i=1:length(Rdir)      %for each load resistance
    Rdir(i)      %check progress
    Rpath=strcat(genPath,Rdir(i),"\");
    %path where csv files for 1 load are stored

    for j=1:length(dirs)      %for each measurement made
        dirs(j)      %check progress on command window
        fpath=strcat(Rpath,dirs(j));
        %path for csv files of a specific frequency
        cd(fpath);
        for l=1:buf_size      %for each cycle
```

```

1           %check progress
filename=strcat(dirs(j),'_'); %specific csv file
if (l<10)
    filename=strcat(filename,'0');
end
filename=strcat(filename,num2str(l),'.csv');

[time,chA,chB]=readcsv(filename,headers,2);

tmp1=strings(1,length(chA(:,1)));
tmp2=strings(1,length(chB(:,1)));
%tmp3=strings(1,length(time(:,1)));

%cast output of readcsv to string
for k=1:length(chA(:,1))
    tmp1(k)=chA(k,:);
    tmp2(k)=chB(k,:);
    %tmp3(k)=time(k,:);
end

%cast string to double
chA=str2double(tmp1);
chB=str2double(tmp2);
%time=str2double(tmp3);

figure(1)
plot(chA,chB);
hold on
grid on

%create variable avchA and avchB ...
%if it doesn't exist
if (~exist('avchA','var'))
    avchA=zeros(1,length(chA));
end
if (~exist('avchB','var'))
    avchB=zeros(1,length(chB));
end
%add the results from each cycle for averaging
avchA=avchA+chA;
avchB=avchB+chB;
end

%get units from csv file
[ut,uchA,uchB]=readcsv(filename,headers,1);
ut=(ut(1,:));
uchA=(uchA(1,:));
uchB=(uchB(1,:));

%create directory to save figures ...
%if it doesn't exist and go there
saveDir=['C:\Users\joao\Documents\MATLAB\' ...
        'tese\medidas pico\figs\'];
saveDir=strcat(saveDir,Rdir(i));
if (~exist(saveDir,'dir'))
    mkdir(saveDir);
end
cd(saveDir);

```

```

xlabel(strcat("chA",uchA));
ylabel(strcat("chB",uchB));
title(strcat(Rdir(i),"/",dirs(j),"(all)"));
savefig(strcat(dirs(j),"(all).fig"));
print(strcat(dirs(j),"(all)","-dpng");
close

%calculate average of all cycles
avchA=avchA/buf_size;
avchB=avchB/buf_size;
figure(2)
plot(avchA,avchB,"b")
xlabel(strcat("chA",uchA));
ylabel(strcat("chB",uchB));
title(strcat(Rdir(i),"/",dirs(j),"(avg)"));
savefig(strcat(dirs(j),"(avg).fig"));
print(strcat(dirs(j),"(avg)","-dpng");
close

%smoothing of signal with moving mean
avchA=movmean(avchA,meanWindow);
avchB=movmean(avchB,meanWindow);
figure(3)
plot(avchA,avchB,"b");
grid on
title(strcat("R=",num2str(10^(i+2),"%10.1e"),...
" $\Omega$, f=",num2str(10^(j-1),"%10.1e")," Hz"));
xlabel(strcat("chA",uchA));
ylabel(strcat("chB",uchB));
savefig(strcat(dirs(j),"(mean).fig"));
print(strcat(dirs(j),"(mean)","-dpng");
close

figure(4)
plot(avchA,avchB,'DisplayName',dirs(j));
legend('-DynamicLegend');
grid on
hold on

clear avchA avchB
%delete avchA and avchB for next iteration
%done this way to account for csv ...
%files of different sizes
end
figure(4)
xlabel(strcat("chA",uchA));
ylabel(strcat("chB",uchB));
title(strcat(Rdir(i),"(sup)"));
savefig(strcat(Rdir(i),"(sup).fig"));
print(strcat(Rdir(i),"(sup)","-dpng");
close
end

```

```

function [varargout] = readcsv(filename,headers,startrow)
%reads .csv file specified by the filename argument,
%storing in output the columns with the headers specified,
%starting in row startrow (header doesn't count as row).
%If headers' first element is "*", then all columns are read.
    warning('off','all');

    Tbl = readtable(filename);
    varNames = Tbl.Properties.VariableNames;
    Tbl = readtable(filename);

    getAll=headers(1)=="*";

    if(getAll)
        outsize=width(Tbl);
    else
        outsize=length(headers);
    end

    varargout=cell(1,length(headers));

    for k=1:outsize
        if(getAll)
            varargout{k} = table2array(Tbl(startrow:end,k));
        else
            varargout{k} = table2array(Tbl(startrow:end,strcmp(varNames,headers(k))));
        end

        tmp=varargout{k};
        if iscell(tmp)
            tmp=tmp{1};
            varargout{k}=cast(varargout{k},class(tmp));
        end
    end
    warning('on','all')

end

```





# Bibliography

- [1] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.
- [2] B. Mohammad et al., “State of the art of metal oxide memristor devices,” *Nanotechnol. Rev.*, vol. 5, no. 3, pp. 311–329, 2016.
- [3] L. Chua, “Memristor-The missing circuit element,” *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.
- [4] L. Chua and S. M. Kang, “Memristive devices and systems,” *Proc. IEEE*, vol. 64, no. 2, pp. 209–223, 1976.
- [5] L. Chua, “Everything you wish to know about memristors but are afraid to ask,” *Radio-engineering*, vol. 24, no. 2, pp. 319–368, 2015.
- [6] L. Chua, *Five non-volatile memristor enigmas solved*, vol. 124, no. 8. Springer Berlin Heidelberg, 2018.
- [7] R. Kozma, R. E. Pino, and G. E. Paziienza, *Advances in neuromorphic memristor science and applications*. 2012.
- [8] F. Z. Wang, L. Li, L. Shi, H. Wu, and L. O. Chua, “ $\Phi$  memristor: Real memristor found,” *J. Appl. Phys.*, vol. 125, no. 5, 2019.
- [9] Y. N. Joglekar and S. J. Wolf, “The elusive memristor: Properties of basic electrical circuits,” *Eur. J. Phys.*, vol. 30, no. 4, pp. 661–675, 2009.
- [10] Z. Biolek, D. Biolek, and V. Biolková, “SPICE model of memristor with nonlinear dopant drift,” *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.
- [11] M. D. Pickett et al., “Switching dynamics in titanium dioxide memristive devices,” *J. Appl. Phys.*, vol. 106, no. 7, pp. 1–6, 2009.
- [12] A. S. Oblea, A. Timilsina, D. Moore, and K. A. Campbell, “Silver chalcogenide based memristor devices,” *Proc. Int. Jt. Conf. Neural Networks*, 2010.
- [13] M. Laiho, E. Lehtonen, A. Russell, and P. Dudek, “Memristive synapses are becoming reality,” *Neuromorphic Eng.*, no. June 2015, pp. 10–12, 2010.
- [14] M. Laiho, E. Lehtonen, A. Russell, and P. Dudek, “Memristive synapses are becoming reality,” *Neuromorphic Eng.*, no. June 2015, pp. 10–12, 2010.

- [15] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Lett.*, vol. 32, no. 10, pp. 1436–1438, 2011.
- [16] K. Miller, K. S. Nalwa, A. Bergerud, N. M. Neihart, and S. Chaudhary, "Memristive behavior in thin anodic Titania," *IEEE Electron Device Lett.*, vol. 31, no. 7, pp. 737–739, 2010.
- [17] T. Molter, "The Generalized Metastable Switch Memristor Model – Knowm.org," 2015-04-19. [Online]. Available: <https://knowm.org/the-generalized-metastable-switch-memristor-model/>. [Accessed: 08-Mar-2019].
- [18] T. Molter and A. Nugent, "The Mean Metastable Switch Memristor Model in Xyce – Knowm.org," 2017-01-15. [Online]. Available: <https://knowm.org/the-mean-metastable-switch-memristor-model-in-xyce/>. [Accessed: 08-Mar-2019].
- [19] I. Misbah Ramadan EE dept. Technion, Israel Shahar Kvatinsky EE dept. Technion, "(451) VTEAM - Memristor Model Tutorial (SPICE) - YouTube." [Online]. Available: <https://www.youtube.com/watch?v=wFqL5yYUEw8>. [Accessed: 23-Apr-2019].
- [20] I. Vourkas and G. C. Sirakoulis, "Emerging memristor-based logic circuit design approaches: A review," *IEEE Circuits Syst. Mag.*, vol. 16, no. 3, pp. 15–30, 2016.
- [21] I. Vourkas and G. C. Sirakoulis, "On the generalization of composite memristive network structures for computational analog/digital circuits and systems," *Microelectronics J.*, vol. 45, no. 11, pp. 1380–1391, 2014.
- [22] J. Reuben et al., "Memristive logic: A framework for evaluation and comparison," 2017 27th Int. Symp. Power Timing Model. Optim. Simulation, PATMOS 2017, vol. 2017–January, no. March 2018, pp. 1–8, 2017.
- [23] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL - Memristor Ratioed Logic," *Int. Work. Cell. Nanoscale Networks their Appl.*, pp. 1–6, 2012.
- [24] K. Cho, S. J. Lee, and K. Eshraghian, "Memristor-CMOS logic and digital computational components," *Microelectronics J.*, vol. 46, no. 3, pp. 214–220, 2015.
- [25] L. Gao, F. Alibart, and D. B. Strukov, "Programmable CMOS/memristor threshold logic," *IEEE Trans. Nanotechnol.*, vol. 12, no. 2, pp. 115–119, 2013.
- [26] A. Fabien, G. Ligang, D. H. Brian, and B. S. Dmitri, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 75201, 2012.
- [27] Y. V. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 57, no. 8, pp. 1857–1864, 2010.
- [28] I. Vourkas, J. Gómez, Á. Abusleme, N. Vasileiadis, G. C. Sirakoulis, and A. Rubio, "Exploring the voltage divider approach for accurate memristor state tuning," *LASCAS 2017 - 8th IEEE Lat. Am. Symp. Circuits Syst. R9 IEEE CASS Flagsh. Conf. Proc.*, no. 3160042, pp. 2–5, 2017.

- [29] T. Bunnam, A. Soltan, D. Sokolov, and A. Yakovlev, "An Excitation Time Model for General-purpose Memristance Tuning Circuit," *Proc. - IEEE Int. Symp. Circuits Syst.*, vol. 2018–May, 2018.
- [30] Y. V. Pershin, J. Martinez-Rincon, and M. Di Ventra, "Memory circuit elements: From systems to applications," *J. Comput. Theor. Nanosci.*, vol. 8, no. 3, pp. 441–448, 2011.
- [31] I. Vourkas and G. C. Sirakoulis, "Memristor-based combinational circuits: A design methodology for encoders/decoders," *Microelectronics J.*, vol. 45, no. 1, pp. 59–70, 2014.
- [32] G. Papandroulidakis, I. Vourkas, A. Abusleme, G. C. Sirakoulis, A. Rubio, and S. Member, "Crossbar-Based Memristive Logic-in-Memory Architecture," vol. 16, no. 3, pp. 491–501, 2017.
- [33] I. Vourkas and G. C. Sirakoulis, "A novel design and modeling paradigm for memristor-based crossbar circuits," *IEEE Trans. Nanotechnol.*, vol. 11, no. 6, pp. 1151–1159, 2012.
- [34] I. Vourkas and G. C. Sirakoulis, "A novel design and modeling paradigm for memristor-based crossbar circuits," *IEEE Trans. Nanotechnol.*, vol. 11, no. 6, pp. 1151–1159, 2012.
- [35] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, and G. C. Sirakoulis, "Boolean logic operations and computing circuits based on memristors," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 61, no. 12, pp. 972–976, 2014.
- [36] E. Lehtonen, J. Poikonen, and M. Laiho, "Implication logic synthesis methods for memristors," *ISCAS 2012 - 2012 IEEE Int. Symp. Circuits Syst.*, no. 1, pp. 2441–2444, 2012.
- [37] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 10, pp. 2054–2066, 2014.
- [38] S. Kvatinsky et al., "MAGIC - Memristor-aided logic," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.
- [39] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided loGIC (MAGIC)," *IEEE Trans. Nanotechnol.*, vol. 15, no. 4, pp. 635–650, 2016.
- [40] R. Ben Hur and S. Kvatinsky, "Memristive memory processing unit (MPU) controller for in-memory processing," *2016 IEEE Int. Conf. Sci. Electr. Eng. ICSEE 2016*, pp. 1–5, 2017.
- [41] S. Smaili and Y. Massoud, "Memristor State to logic Mapping for Optimal Noise Margin in Memristor Memories," *14th IEEE Int. Conf. Nanotechnol.*, no. 1, pp. 291–295, 2014.
- [42] C. Yakopcic, T. M. Taha, and R. Hasan, "Hybrid crossbar architecture for a memristor based memory," *Natl. Aersp. Electron. Conf. Proc. IEEE*, vol. 2015–February, pp. 237–242, 2015.
- [43] S. H. Amer, A. H. Madian, and A. S. Emara, "Design and analysis of memristor-based min-max circuit," *Proc. IEEE Int. Conf. Electron. Circuits, Syst.*, vol. 2016–March, pp. 187–190, 2016.

- [44] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, and K. Bertels, “Boolean logic gate exploration for memristor crossbar,” *Proc. - 2016 11th IEEE Int. Conf. Des. Technol. Integr. Syst. Nanoscale Era, DTIS 2016*, no. October 2017, 2016.
- [45] J. Zheng, Z. Zeng, and Y. Zhu, “Memristor-based nonvolatile synchronous flip-flop circuits,” *7th Int. Conf. Inf. Sci. Technol. ICIST 2017 - Proc.*, pp. 504–508, 2017.
- [46] P. W. C. Ho, H. A. F. Almurib, and T. N. Kumar, “Non-volatile D-latch for sequential logic circuits using memristors,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2016–January, no. iii, pp. 1–4, 2016.
- [47] J. Chowdhury, J. K. Das, N. K. Rout, and P. Roy, “Delay and Power Optimization of Memristor based basic Sequential Circuit Element,” vol. 4, no. 28, pp. 4–6, 2016.