UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Enginyeria Electrònica

"RELIABILITY-AWARE CIRCUIT DESIGN TO MITIGATE IMPACT OF DEVICE DEFECTS AND VARIABILITY IN EMERGING MEMRISTOR-BASED APPLICATIONS"

Thesis submitted in partial fulfillment of the requirement for the PhD Degree issued by the Universitat Politècnica de Catalunya, in its Electronic Engineering Program.

Manuel Escudero López

Director: *José Antonio Rubio Solà*

Codirector: *Ioannis Vourkas*

data de lectura de la tesi

Acknowledgement

As a prelude of this dissertation, I would like to express my sincere gratitude to all persons which have contributed to the work reported here, as well as those who have supported me during the journey of my PhD.

First of all, I want to thank my supervisors, Professor Antonio Rubio and Professor Ioannis Vourkas. I am indebted to them, for their support, patience and knowledge. It could be said that my PhD started with Antonio in his office, while hearing for first time the word memristor; I am deeply grateful for all the motivating discussions, all his advices and the freedom he gave me to manage my PhD. Professor Ioannis Vourkas joined later as a co-supervisor; his expertise and initiative has proven truly valuable. Not only that, but I wish to show my gratitude for his kindness whenever I've been in Valparaíso, and all the leisure time I enjoyed there with him, Evelyn and other members of Universidad Técnica Federico Santa Maria.

I don't want to forget any of the members of the HIPICS group for their constructive criticism they delivered when I presented my work in our meetings, and for helping in others matters I had during the PhD.

It is also whole-heartedly appreciated the company of my colleagues during the PhD, and especially Chenna, Santi and Lukasz for many interesting conversations, always accompanied with a coffee, and other nice ones during the lunch time with Vasileios in these lasts months.

Finally, I wish to acknowledge the invaluable love and support of my family: my sister, my brother, and my parents. And not just them, but also the family I chose, my friends. Thank you to all of them.

Abstract

During the last decades, the semiconductor industry has fostered a fast downscale in technology, propelling this way the large scale integration of CMOS-based systems. The benefits of miniaturization are numerous; faster switching frequency, lower voltage supply and higher device density, to name a few. However, such aggressive scaling trend has not been possible without several challenges to overcome, such as the leakage currents, yield reduction or the increase in the overall system power dissipation. New materials, changes in the device structures and new circuit architectures are key to maintain this miniaturization trend. It is foreseen that 2D integration will eventually come to an insurmountable physical and economic limit, in which new strategic directions are required, such as the development of new device structures, 3D architectures, or heterogeneous systems that take advantage of the best properties of different technologies; i.e. both the ones already consolidated as well as the emergent ones that provide performance and efficiency improvements for several applications.

In this context, the resistive switching devices (memristors) emerge as one of several candidate device technologies in this eminent quest for the most "suitable" nanoelectronic devices for future integrated circuits. Memristor, which is a blend of the words memory and resistor, is a passive device first postulated by Leon Chua in 1971. Unlike the rest of fundamental passive elements, memristors have the distinctive feature of modifying their resistance according to the charge that passes through them, which is then kept unchanged when there is no charge flow. Even though there was no physical device evidence when the concept of memristors was first published, the Hewlett Packard (HP) Labs claimed in 2008 the fabrication of the first real memristor. This milestone triggered an unexpectedly high research activity in the memristor field, concerning both the search for new materials and structures as well as the use in a variety of potential applications. Nowadays, memristors are not only appreciated in memory systems for their nonvolatile storage properties, but in many other fields too, such as digital computing, signal processing circuits, non-conventional (neuromorphic) computing, or in analog applications such as in chaotic circuits. In spite of their very promising features, memristors still show some discouraging aspects: they demonstrate significant device variation and limited endurance due to degradation, compared to other device alternatives.

This Thesis explores the challenges that memristor variation and malfunction imposes to potential applications. The main goal is to incorporate efficiently the variability in device modeling and explore circuits design strategies to either avoid reliability problems or take advantage of variability for the benefit of certain applications. Through a collection of different scenarios in which reliability issues are assumed present, their impact is thoroughly studied by means of simulations and solutions are proposed for several emerging applications of memristors. More specifically, this Thesis is first contextualized in **Chapter 1,** where also the principal objectives are listed. In **Chapter 2** the memristor device is introduced, both at conceptual and experimental level, and all the necessary conceptual considerations are expressed, for the comprehension of the content in the following chapters. **Chapter 3** explores in depth the phenomena that cause the lack of reliability in memristors, and introduces several device models that include such defects

in simulations. The rest of the Thesis is devoted to cover different emergent applications of memristors. More specifically, **Chapter 4** exhibits nonvolatile memory systems, and specifically an online test method for faults in memory cells based on memristors. Digital computing is the focus of **Chapter 5**, where a solution is provided for the yield reduction in in-memory logic operations with memristors owing to variability. Finally, **Chapter 6** reviews applications of memristors in the analog domain, and focuses on the exploitation of results observed in faulty networks of memristors used as interconnect mediums for chaotic systems, aiming to achieve synchronization. This Thesis concludes with **Chapter 7** along with some perspectives for future work.

Resumen

En las últimas décadas, la industria de los semiconductores ha propiciado un rápido escalado en la tecnología, propulsando la integración a gran escala de sistemas basados en CMOS. Los beneficios de la miniaturización son amplios, destacando el aumento de frecuencia y densidad de dispositivos, o la disminución del voltaje de alimentación. No obstante, este ritmo de escalado agresivo no ha estado exento de retos, como por ejemplo fugas de corriente, disminución de la fiabilidad o el incremento potencia en los sistemas. Nuevos materiales, cambios en las estructuras de los dispositivos y nuevas arquitecturas han sido claves en aras de mantener el ritmo de la miniaturización. Se prevé que la integración en 2D llegará a un límite físico y económico insalvable, en el que se deberán tomar otras direcciones estratégicas, como el desarrollo de nuevas estructuras en los dispositivos, arquitecturas 3D o bien sistemas heterogéneos que saquen partido a diferentes tecnologías, tanto aquellas consolidadas como otras de emergentes que mejoren el desempeño y eficiencia de las aplicaciones.

En este contexto, el memristor se erige como un posible candidato en la carrera para encontrar dispositivos emergentes adecuados. Acrónimo de 'memory' (memoria) y 'resistor' (resistencia), el memristor es un elemento pasivo postulado por Leon O. Chua en 1971. A diferencia de los otros elementos pasivos fundamentales, los memristores tienen la particularidad de modificar la resistencia según la carga que ha circulado a través de estos dispositivos, viéndose inalterada cuando esta circulación se detiene. Aunque en el momento de su concepción el memristor no tenía ninguna implementación física, en 2008 los laboratorios de HP afirmaron haber obtenido el primer memristor real. Este hito desencadenó una actividad en investigación sin precedentes sobre los memristores, tanto en la búsqueda de materiales y estructuras como en aplicaciones potenciales. Los memristores no solo son apreciados en sistemas de memoria por sus capacidades de almacenamiento no volátil, sino en otros campos como por ejemplo la computación digital, circuitos de procesamiento de señal o aplicaciones no convencionales como computación neuromórfica. A pesar de sus capacidades prometedoras, los memristores padecen de un principal inconveniente: muestran una variación de dispositivos y un tiempo de vida limitado debido a la degradación comparado con otras alternativas.

Esta tesis explora los retos que las variaciones y el fallo de memristores imponen en las aplicaciones potenciales. El objetivo principal es la propuesta de circuitos y estrategias para evitar problemas de fiabilidad, o bien sacar ventaja de ellos. A través de una colección de escenarios en los que los problemas de fiabilidad son presentes, el impacto de éstos es estudiado mediante simulaciones. La contextualización de la tesis junto a los objetivos se expone en el **Capítulo 1**. En el **Capítulo 2** introduce el memristor a nivel conceptual y experimental, y presenta diferentes modelos compactos usados posteriormente en simulaciones. El **Capítulo 3** profundiza en los fenómenos que provocan la falta de fiabilidad en los memristores y provee de modelos para incorporar estos defectos en las simulaciones. El resto de la tesis recorre diferentes aplicaciones mediante los tres capítulos restantes. Así entonces, el **Capítulo 4** expone los sistemas de memorias no volátiles, y concretamente un método de test de celdas defectuosas. La computación digital se presenta el **Capítulo 5**, en el cual se propone una solución a la reducción de

fiabilidad en operaciones lógicas debido a la variación en los memristores. Por último, el **Capítulo 6** revisa las aplicaciones en el dominio analógico, y se centra en la explotación los resultados derivados de defectos en medios de interconexión basados en memristores para la sincronización de sistemas caóticos. Finalmente, las conclusiones de la tesis se exponen en el **Capítulo 7** junto a perspectivas de trabajo futuro.

Table of Contents

Chapter 1 Introduction

This chapter highlights the increasing interest of the scientific community in the emergent resistive switching devices (memristors), while also identifying the issues that motivate and justify the research conducted in this Thesis. The initial emphasis is the urge of a new nanotechnology able to compete or overcome the challenges the semiconductor industry is currently facing. The main objectives and the organization of this Thesis are also provided at the end of this chapter.

1.1 Thesis Motivation

The so called "Moore's Law" from 1965 stated that the number of transistors in integrated circuits would double every two years [1]. What initiated as a descriptive analysis of the industry advancements, it finally proved to be the standard for the development of the semiconductor industry for several decades. This trend eventually experienced some reviews due to the multiple difficulties appearing in the manufacturing processes when scaling down the technology.

The technology roadmap for semiconductors (ITRS) [2], while recognizing that the Moore's Law would not stand indefinitely in the future, set multiple goals and designed specific roadmaps covering diverse areas of research. So the focus has been divided in three domains aiming to overcome this scaling slow-down: "More Moore", "More than Moore" and "Beyond CMOS". **Fig. 1-1** presents all these roadmaps and the potential solutions considered.

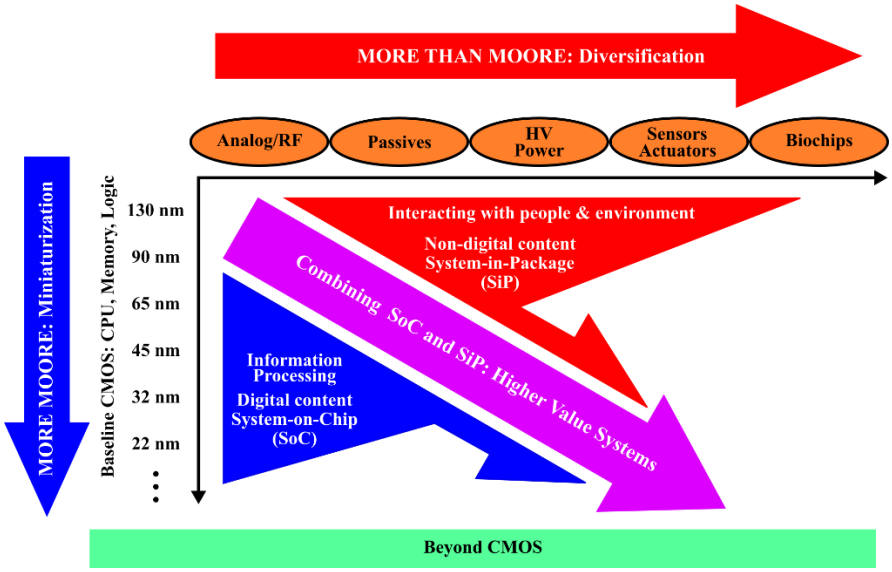


Fig. 1-1. There are three directions fueling the continuous improvement in the nanoelectronics industry: direct miniaturization of digital systems, assistance with non-digital subsystems and thinking beyond the eventual limits in CMOS. Adapted from [2].

More specifically, the "More Moore" domain refers to the efforts for further miniaturization of CMOS circuits by the improvement of processes, research in new materials and device structures, as well as the subsequent benefits in power and cost. In "More than Moore" the direction is to rely on non-digital subsystems to help in the miniaturization of the technology, even if these parts do not scale down at the same pace as the digital counterpart. Finally, "Beyond CMOS" is taken into account, which concerns a long

term vision that holds disruptive technologies to compete against the future-sighted limits of CMOS technology scaling.

All these lines of action converge to a common point: research and industry must cooperate and find suitable technologies and ideas to cope with the future challenges. In this direction, memristors could hold the key to a new era in electronics, being a very promising technology for next-generation chips that need to be highly reconfigurable, scalable and energy-efficient.

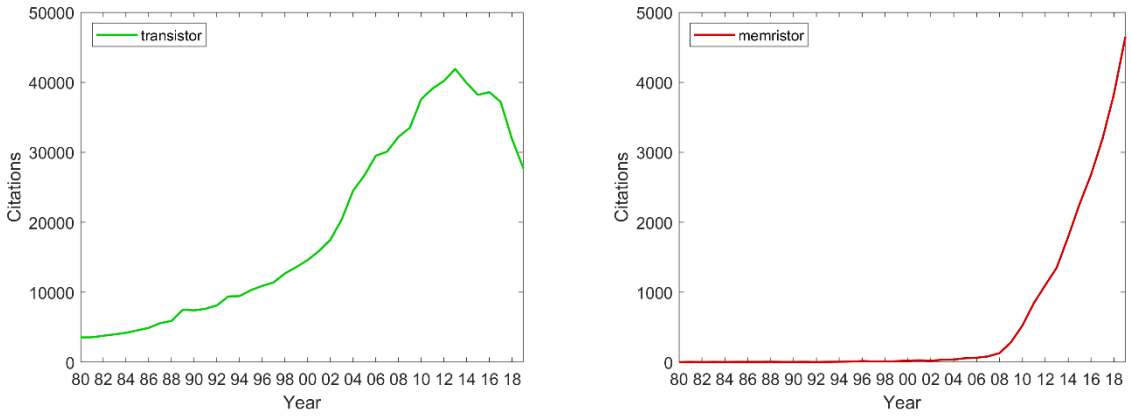
The term memristor was established by Leon Chua in 1971 for a conceptual device, being the fourth passive circuit element along with resistor, capacitor and inductor, which related charge and flux variables [3]. The term is a contraction of "memory" and "resistor", because the history of the control/state variable (charge of flux) determined the resistance value. At the time memristor was first defined, no real device was found to match the expected behavior; for this reason, the topic remained virtually unexplored for some decades. The fact is that resistive switching phenomena in different materials had been already studied from the ´60s, but this was not recognized to be a memristive behavior until 2008 [4]. Failing to establish a practical connection was probably attributed to the fact that charge and flux were not necessarily involved in physical device implementations. Eventually, many years passed until Hewlett Packard Laboratories (HP Labs) matched for the first time in 2008 the theoretical memristor concept with a physical nanodevice, which was a 5-nm TiO2 layer sandwiched between two metal electrodes [5]. This milestone unexpectedly triggered the research activity around the memristors, pushing the development of several such devices and exploration of applications.

Memristors demonstrate many promising features, such as plasticity, analog nature, non-volatility, along with a low power consumption, high integration density, and excellent scalability. In summary, their main characteristics of a memristor are as follows:

- It is a passive element, i.e. no active elements are required to implement it. Furthermore, it is not possible to build it using the other fundamental passive elements, resistors, capacitors or inductors.
- It is normally a device with a non-volatile resistive state; without external stimulus the internal state is kept intact. A non-volatile internal state is a key characteristic for the potential use as a storage element.
- It is expected to achieve better device density than conventional transistors based on simple, regular, and even multi-layer array structures such as the crossbar geometry.
- It's compatible with the back-end-of-line (BEOL) CMOS fabrication processes.

The memristor-related scientific activity has increased enormously during the last decade. At this stage, memristor is not any more just an idea from circuit theory, but technology and theory are reaching a level of maturity so that the foundations of real circuit-level memristive applications can be established. In order to illustrate the increasing interest for memristors in academia, the results per year in the Google

Scholar search engine of the keyword transistor and memristor (excluding patents and citations) are shown in **Fig. 1-2**. Concretely, is clear that after 2008 the increase in research activity was triggered after the achievement described in [5]. Although there is no doubt that transistor is still dominating the scene, it is true that the nearly exponential increase in the memristor-related citations and also the inflection point in the results concerning the transistors, both indicate the emergence of a new trend in electronics with an ever growing variety of memristive applications.



(a) (b)

Fig. 1-2. Evolution of citations per year according to Google Scholar for the keywords (a) transistor and (b) memristor, from the 80's up to the present.

Owing to its unique characteristics, memristor is considered to be a suitable candidate to boost several applications and provide them with low energy/area advantages combined with CMOS integration compatibility. Memristor has been proposed for applications in several fields (**Fig. 1-3**) where, among others, we consider memory, digital computing, analog signal processing, analog computing and security.

Definitely, the most straightforward application is as core elements in memory systems [6]. The memristor itself can act as a memory cell, whose content can be read or programmed in the form of a resistance. Its nonvolatile nature and potential high device density makes them suitable as storage elements in dense, low-power resistive memories. The storage of information in a memristor can be binary or multi-level, depending on the switching behavior of the device and other characteristics that we will have the opportunity to comment extensively in the following Chapters.

Moreover, memristors can be used in digital computing applications using different approaches [7]. One such approach considers the memristor switching in a manner analogous to the well-known operation of a transistor; i.e. behaving as a binary switch. In this sense, a memristor could be used as a pass gate. On the other hand, memristor non-volatile nature allows the development of new computing paradigms. For instance, in-memory computing allows computation and memory tasks taking place in the same devices, where memristor acts both as memory cell and as part of a logic gate implementation.

On the other hand, given that some memristors can tune their state/memristance in an analog manner, they could also be considered in analog applications [8]. Applicability of memristors is as simple

as it sounds: a memristor could essentially replace any resistor that has a particular role in a given circuit (e.g. a feedback resistor in an amplifier or a reference resistor in a comparator, to name a few), thus improving the overall functionality of signal processing circuits and/or upgrading the circuit from static to programmable, leading for instance to programmable gain amplifiers, filters or analog-to-digital converters.

Signal processing is not the only application in the analog realm. Further (and more ambitious) applications include the field of analog computing, which also includes neuromorphic computing [9]. Neuromorphic computing systems are designed to work in a similar way to the human brain, seeking high savings in power while performing heavy computational tasks in massively parallel architectures. Memristor has been recognized as the electrical equivalent of a biological synapse. On top of that, there are considerable device features, such as the small footprint and nonvolatile storage, which makes memristor a suitable candidate for the implementation of large scale artificial neural networks, taking advantage of the crossbar geometry.

Finally, in the field of secure communications, there are applications of physical unclonable functions (PUF) based on memristors, owing to the intrinsic variability they demonstrate in the switching behavior [10]. Memristor could also play a significant role in secure communications via chaotic system synchronization, where memristors act as coupling elements between chaotic circuits in providing adaptive synchronization properties [11].

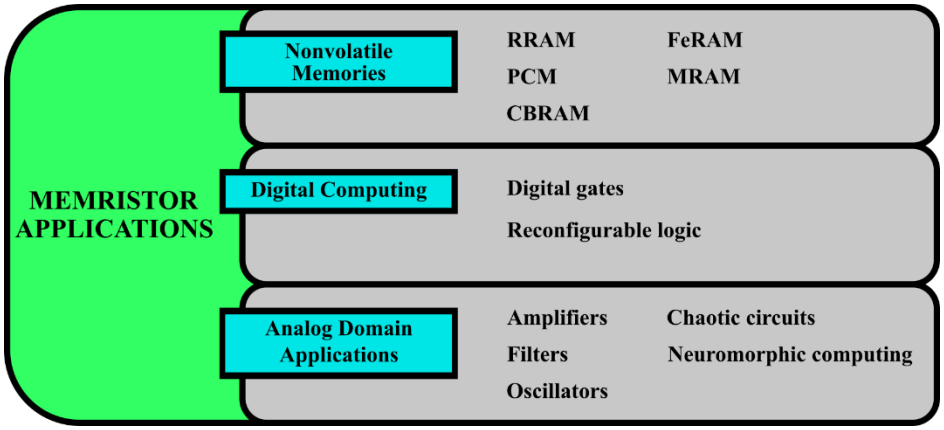


Fig. 1-3. Diagram of different applications benefited by memristor characteristics, classified in three main fields: nonvolatile memories, digital computing and analog domain applications.

Nevertheless, besides the promising advantages offered by memristors in their potential applications, there are some important "brick walls" to overcome on their way towards commercial establishment, such as variability, aging and switching faults, which constitute well-known issues while working in nanoscale devices. In the case of life aging, the memristor is limited in the number of switching cycles it can perform (SET and RESET resistive transitions from a high resistive state to a low resistive state, and vice versa). The degeneration of the resistive states with time (see **Fig. 1-4a**) may put significant limitations for some applications [12]–[14]. On the other hand, the variability is not caused only by the manufacturing processes, but it is known to be intrinsic of every device and demonstrated in various forms

[15]. As a result, the observed memristor characteristics can be related to stochastic behavior (see **Fig. 1-4b**). While some of these issues may put at risk the reliability of several applications, their impact has not been widely considered so far in works published in the literature.

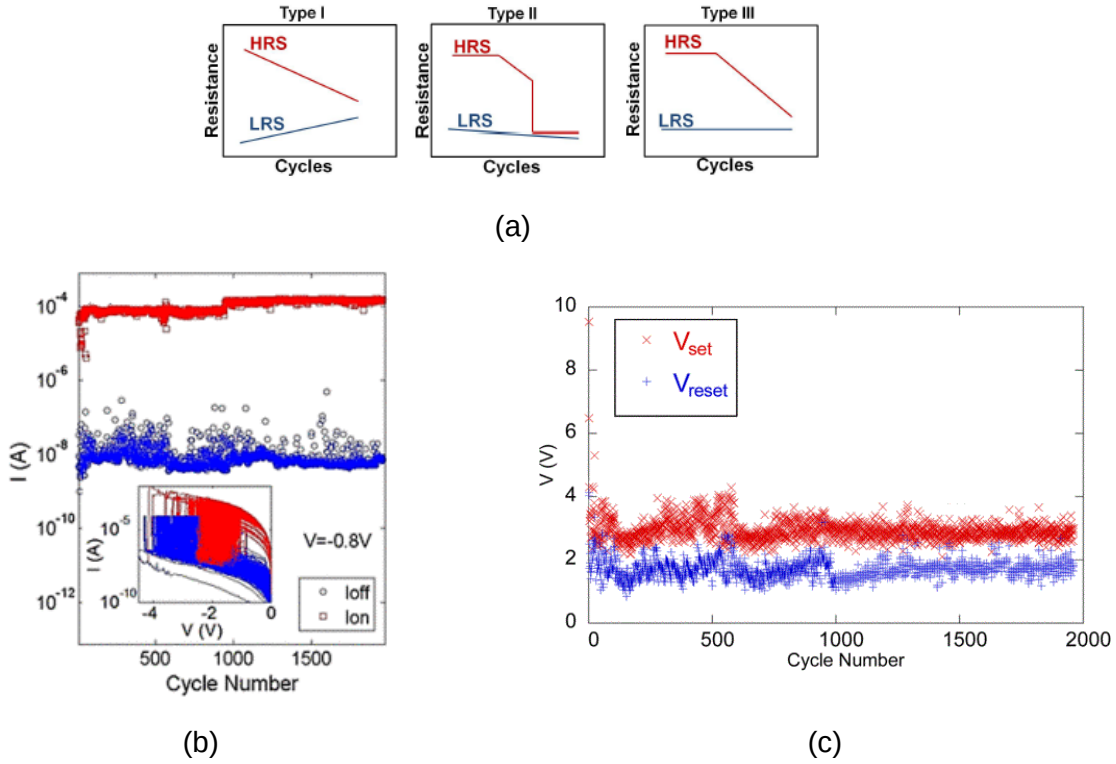


Fig. 1-4. Transition metal oxide switching devices suffer from various limitations related with reliability. Regarding endurance, (a) three types of failures were observed in a device due to electrode oxidation (type I) or lack of adequate oxygen vacancies (type II and III). The drawings show the effects of the degradation in high resistance state (HRS) and/or low resistance state (LRS) over several switching cycles, until both states are not distinguished (extracted from [13]). Variation in measured parameters have been also observed for Ni/HfO_2-based resistive switching devices, in resistive state by means of (b) current measurements of HRS and LRS for several switching cycles (inset shows *i-v* curves of the measured cycles) and (c) variation of the switching voltages V_{SET} and V_{RESET} (extracted from [15]).

1.2 Objectives

The present Thesis investigates the effect of different issues at device-level such as process and cycle to cycle variability or failing memristors, and explores techniques to overcome these issues in applications or alternatively to improve the performance by taking advantage of these issues rather than trying to resolve them. Whenever necessary to clarify some points, experimental data are presented taken from measurements on real device samples. This notwithstanding, the contributions rely mainly on electrical simulations owing to the convenience they offer to incorporate the required type and amount of faults, as well as to control variability in the devices. In this direction, the use of reliable memristor models was considered critical in order to achieve an acceptable degree of certainty in the results of the simulation-based study. The principal goals of this Thesis are:

- to develop testing strategies for memristors in the presence of variability and/or malfunctioning devices.
- to analyze the impact of variability on digital computing circuits and to propose variability-tolerant circuit implementation techniques.

- to study applications where variability or malfunction of devices could improve the performance and/or are favorable in some way to the operation of circuits.

1.3 Thesis Organization

This Thesis is organized as follows: **Chapter 2** provides a thorough overview of the recent history of the memristor, while reviewing the state of the art of real devices, the evolution in modelling techniques and the wide application fields where memristors could be used. **Chapter 3** introduces the concerns that are key in this dissertation: variability and malfunction of memristors, along with some practical experiences and their modelling for simulation purposes. **Chapter 4** discusses memory applications and presents a strategy to detect faulty memristor devices in memory arrays. **Chapter 5** covers digital computation applications, the outcome of taking into account variability issues and novel forms of computation that are proved to be variability-tolerant. **Chapter 6** presents applications where variability and faults are not an issue but instead could be beneficial for circuit/system operation, while studying alternative topologies to improve the performance of the synchronization of a number of chaotic circuits targeting security applications. Finally, **Chapter 7** concludes this thesis.

Chapter 2 Memristors: State of the Art and Preliminary Considerations

An unprecedented attention and extensive research activity on memristor device technology has been observed only since 2008 after the first demonstration of the TiO2-based memristor by Hewlett-Packard Laboratories (HP Labs), who managed to connect the nature of such devices with Chua's 1971 theory (although the hysteretic resistive switching property of oxides sandwiched between metal electrodes was known from the `60s [16], [17]).

Nevertheless, even if we focus on these last few years, memristor has not been a purely monolithic concept, with the relevant theory continuously changing with the new physical evidence reported by the research community. The HP Labs invention in 2008 was soon followed by the identification of additional material compounds which can be the basis of memristive devices and a wide variety of such materials are being studied [18]. While this technology is continuously evolving/maturing, some commercially available memristors were recently released by the Knowm Inc. company [19] bringing this technology closer to researchers that might not have access to fabrication facilities. As typically happens with all new electronic device technologies, modeling and simulation [20] are the first steps to exploring general attributes, verifying theoretical aspects and studying innovative application prospects.

The main objective of this chapter is to give the “big picture” of memristors, starting with the memristor as a concept, following with the state of the art of real devices and device models that were developed to facilitate incorporation of memristors in current circuit designs for existing or new applications. Next, the memristor crossbar topology is introduced, a geometry widely considered to be used in memory and computing applications, permitting to take full advantage of certain characteristics of memristor. Finally, the terminology and considerations that are applied/used in the rest of this Thesis are presented.

2.1 Memristor: Definition and Evolution of the Concept

Today the term memristor usually refers to any resistive switching (RS) device that complies with a set of certain properties known as ``fingerprints" [21]. However, the existence of the memristor was originally proposed by Leon Chua in 1971 as a conceptual device (it was much later when it eventually was related to a physical implementation), being the fourth fundamental circuit element [3]. At that time, it was the result of the analysis of the three fundamental and well-known passive circuit elements; the resistor, the capacitor and the inductor, each one providing a relationship between voltage-current, voltage-charge, and current-flux, respectively. Chua inferred that a fourth fundamental element should exist, the memristor, to relate the charge and flux variables, as shown in **Fig. 2-1a.** Back then, no physical device was found to match the required properties, but it´s operation was successfully emulated by means of active circuit elements.

The ideal memristor is defined by a relation of type $g(q, \varphi) = 0$. The memristor is charge (flux)-controlled if this relation can be expressed as a single valued function of the charge (flux). Equations (1) –

(4) show the explicit relation of charge (flux)-controlled memristors and their characteristic memristance *M* (equivalent to resistance) or memductance *W* (equivalent to conductance).

$$v(t) = M\big(q(t)\big)i(t) \quad (1)$$

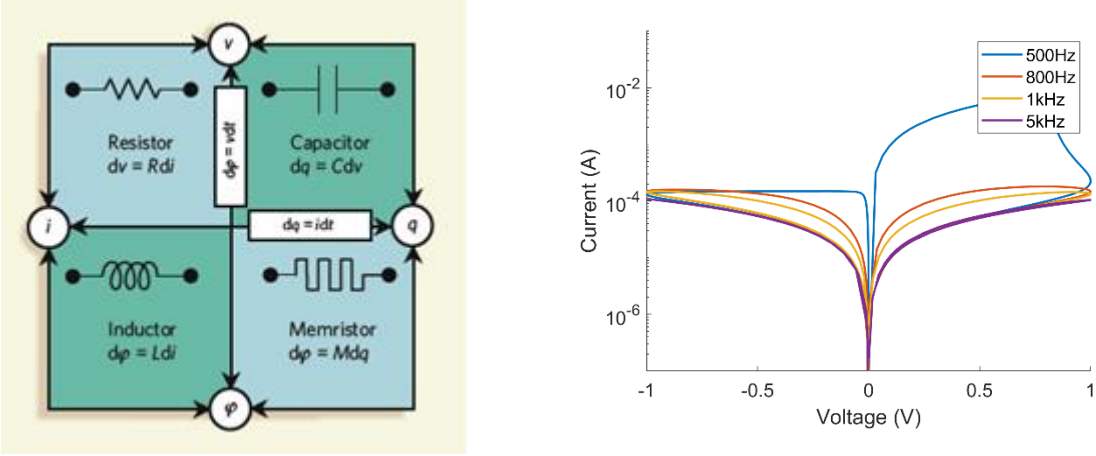
$$M(q) = d\varphi/dq \quad (2)$$

$$i(t) = W\big(\varphi(t)\big)v(t) \quad (3)$$

$$W(q) = dq(\varphi)/d\varphi \quad (4)$$

The main properties that identify a memristor are the following:

- It is a passive element.
- Memristance is not constant but instead it depends on the past history of charge and flux; otherwise it would behave as a typical resistor
- Under certain conditions, the *i*-*v* plot shows a pinched hysteresis loop; high frequencies force this loop to degenerate to a straight line (behavior of a linear resistor). As an example, **Fig. 2-1b** illustrates a simulation of a linear memristor model [5] excited by a 1 V-amplitude sinusoidal voltage using four different frequencies: 500 Hz, 800 Hz, 1 kHz and 5 kHz. Clearly, the loop is narrowing as frequency goes up, losing the hysteretic properties.



(a) (b)

Fig. 2-1. Memristor, the fourth fundamental circuit element. (a) Among the four known magnitudes of current *i*, voltage *v*, charge *q* and fluz *φ*, the memristor was proposed to provide a flux-charge relationship through the memristance M (extracted from [5]). (b) Current-voltage (*i*-*v*) curves of after excitation using sinusoidal voltage of 1 V amplitude and four different frequencies. Results are obtained by means of simulations using the linear memristor model [5].

In 1976, a more general definition of memristors was presented in [22] aiming to incorporate empirically observed phenomena that resembles a memristor, such as in thermistors at constant temperature or the Hodgkin-Huxley axon model. A memristive system is defined in (5), where *x* is an internal state variable, *u* is the independent control variable and *y* is the dependent (output) variable. If restricted to the electrical variables *v* as voltage across the system and *i* as the current passing through the system, a voltage-

controlled memristor is considered when $u = v$ and $y = i$. Likewise, a current-controlled memristor is defined by setting $u = i$ and $y = v$.

$$\begin{aligned} y &= g(x,u,t)u \\ dx/dt &= f(x,u,t) \end{aligned} \tag{5}$$

After the extensive scientific research conducted during all these years and the results presented for real implementations of memristors, some doubts arose about whether the physical resistive switching devices and their respective device models matched the definition of a memristor or memristive device, as they were defined by Chua. In this direction, there have been some initiatives to generalize even more the previous definitions [23]. For instance, in [24] we find an extended definition which includes the effect of non-zero crossing, due to the electromotive force that appear in many resistive switching devices. The definition of the extended system is shown in (6); the mathematical model starts with a voltage-controlled version of equation (5), contained in the term i_{el} and the state equation. The added term i_{ion} models a nanobattery, which essentially consists in an electromotive force v_{emf} and a nonlinear resistor in series that depends on the voltage. About the internal state x, it is bounded between the geometric limits of the active layer thickness d. The state equation is governed by the i_{ion} current (which in turn is controlled by the input v), while k is a constant.

$$\begin{aligned} i &= i_{ion}\left(v_{emf}, v\right) + i_{el}(x,v) \\ dx/dt &= k i_{ion}, \qquad 0 \le x \le d \end{aligned} \tag{6}$$

For the time being, the recent history of the memristor as a device is anything but a clear picture, with constant redefinitions and adaptations of previously established criteria to embrace the appearance of new physical evidences and new devices that are being developed and demonstrate a memristive behavior. Memristors definitely constitute an emerging trend in modern electronics, but a deeper understanding of memristive behavior is the only safe way towards maximum exploitation of the favorable properties and the analog nature of this very promising device technology in innovative applications.

2.2 Memristor Device Implementations

Even though any conversation about memristor devices can nowadays be thought of as being quite "controversial", owing to the contemporary discussion in the scientific community about the existence of the ideal memristor concept and the extent to which real devices comply with the expected behavior, hereinafter we use the term memristor in this Thesis to refer to several types of resistive switching devices.

After the TiO_2-based bipolar memristor by HP in 2008, more material compounds were proposed as the basis of memristor devices [25]. A wide variety of such materials have been studied so far aiming to predict the switching profiles and manipulate them appropriately to achieve better on–off ratio, endurance, and state retention, properties that we will discuss later in more detail. RS devices are normally fabricated in metal-insulator-metal (MIM) structures in which the RS property is observed in the insulating material.

There is a wide range of materials used as electrodes and insulators. The latter can be a metal oxide [26] such as HfOx [27], TiOx [28], AlOx [29], or Ta_2O_5 [30], a chalcogenide such as GeSx [31], a nitride (e.g. SiN [32]), a perovskite [33] or a graphene oxide [34]. It is worth noting though that using such materials as the functional RS layer might not always be compatible with current integrated circuit (IC) industry infrastructure, and this has also led to a focused interest in silicon and silicon oxide as well [35]. Moreover, every material can behave in a different way when exposed to the electrical field induced by the applied voltage.

First of all, in order to better comprehend memristive behavior, there are some terms that must be defined. Memristor devices experience two switching events where resistance is modified: there SET and RESET processes. SET process allow resistance to decrease from a high resistance state (HRS) to a low resistance state (LRS). On the contrary, a RESET process rises the device resistance from LRS to HRS. Usually, for threshold-type switching devices the V_{SET} and V_{RESET} threshold voltages are defined to localize the minimum voltage at which the SET or RESET process is triggered, respectively. However, such voltages are not absolute values, and their values fluctuate due to many factors, such as temperature, external stimulus or device variability. Typically, in experimental results we observe an abrupt current increase during the SET operation, whereas the RESET operation may be more gradual [36]. Moreover, the *i-v* characteristics may be asymmetric w.r.t. the origin. Last, higher absolute voltage values not only provoke higher resistance switching rates but also drive the device to saturation at different resistive levels, as observed in a wide range of analog memristors [37].

Switching mechanism can be classified as unipolar or bipolar. Examples for the characteristic i-v curve for both categories are shown in **Fig. 2-2**. For a bipolar device the SET and RESET processes occur in opposite polarities. On the contrary, a unipolar device operates with one polarity only, and has SET and RESET processes dependent on the applied voltage amplitude, rather than polarity. Devices with electrodes made of noble metals like Pt, Cu or Ru, are unipolar, while a bipolar device may be built by replacing one of the top or bottom electrode materials with an oxidizable material [6].

Fig. 2-2. Memristor switching mechanisms: (a) unipolar switching and (b) bipolar switching. Compliance current is applied to limit the maximum current during the SET process, to protect the device from damage. Extracted from [6].

There are identified three types of memristors regarding the location and geometry of the atomic reconfiguration in switching events: filament, interface and bulk type memristors [38]. In the first case, RS has been identified as originating from the local formation/disruption of conductive filaments (CFs) between the metallic electrodes [39] (see **Fig. 2-3a**). The filamentary type switching includes thermochemical (TCM), electrochemical (ECM), and valance change (VCM) mechanisms. Such memristors may be unipolar or bipolar and their switching is fast and abrupt, thus they are recommended for digital computing applications. On the other hand, RS can be caused by an interfacial mechanism [40] (see **Fig. 2-3b**) with distributed migration of traps at the metal-insulator interface (e.g. oxygen vacancies distributed along an interface that controls the conductivity of the MIM structure). The interface-type switching includes VCM, purely electronic and electrostatic effects, and ferroelectric polarization mechanism. Unlike filamentary devices which generally show sharp conductance transitions, interfacial (or homogeneous) switching devices change their resistance by a uniform interface effect, thus tend to respond more slowly to the applied input signals, modifying their state in a more incremental manner, desired for analog applications. For a proper analysis of the main differences resulting in filamentary versus interfacial RS types, from a device performance viewpoint, see [41]. Finally, the bulk memristors typically consist in a phase change effect inside the switching layer (see **Fig. 2-3c**).

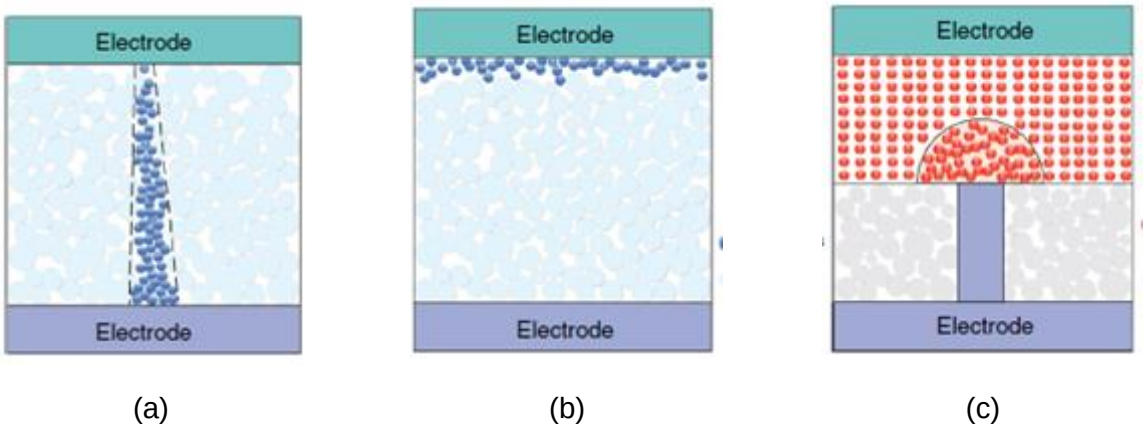


Fig. 2-3. Types of switching mechanisms in memristors depending on the atomic reconfiguration. (a) Filamentary type, (b) interface and (c) bulk type. Extracted from [38].

There are certain parameters which are normally used to evaluate the behavioral properties of memristors. IRDS 2018 in [42] summarizes the potential different emerging memory devices. **Table 2-1** gathers some of these conclusions for unipolar and bipolar filamentary memristors, bipolar interface memristors, and phase change memories (PCM). Bipolar filamentary memristors show promising features in many of the presented categories, especially in fabrication cost, retention and endurance, but the variability and power are their weak points. On the other hand, bipolar interface memristors do better in these two categories but with the trade-off of performing worse in other important factors such as retention or endurance. A recent review on resistive switching device [43] shows how some of these technology requirements have been surpassed for certain devices, achieving milestones such as endurance $>10^{12}$ cycles

or switching times as low as 300 ps, and providing other interesting data as operating voltages as low as 0.5 V or power consumption of 0.1 pJ per switching.

Table 2-1. Potential of the emerging memory technology candidates **(From** [42]**)**

	Unipolar Filament	Bipolar Filament	Bipolar Interface	PCM
Scalability	Fmin < 10 nm			
MLC	Possible			Feasible
3D integration	Feasible			
Fabrication Cost	Medium	Low	Medium	
Retention	Modest (> 1 yr)	Long (> 10 yrs)	Modest (> 1 yr)	Modest (> 1 yr)
Latency	Medium (0.3 – 10 μs)			
Power	Medium	High	Medium	High
Endurance	$< 10^{10}$ cycles		$< 10^{5}$ cycles	$< 10^{10}$ cycles
Variability	Problematic		Low	Reasonable

Finally, it's worth mentioning that, even with the good performance observed, there is still much work to do in order to achieve the ultimately goal of transferring memristors into a real, competitive solution. The good news is that nowadays there are companies working in this direction. Such examples are Knowm Inc. [19], that commercializes memristors encapsulated in arrays or crossbars, Memryx [44], a start-up that designs brain-inspired computer chips for AI applications, and MemComputing Inc. [45], attempting to solve complex optimization problems using an innovative analog computing paradigm.

2.3 Memristor Compact Device Models

Ever since the first nanoscale memristors appeared, the development of accurate models for memristors emerged as a research topic of utmost interest. The scientific community was interested in using memristor models to enable circuit design in simulation environments for the exploration of different application prospects. Hence, there is a considerable demand for reliable compact models that capture the rich dynamics and characteristics of these devices. Indeed, the development of compact behavioral models has contributed significantly to the progress of research in memristor technology, being adequate to capture the experimentally observed RS behavior. In this context, we next present a brief overview covering a wide variety of models ranging from simple and preliminary ones to more complex physics-based models that go deeper into device dynamics and also include certain additional characteristics such as variability, temperature dependency, and stochasticity, to name a few.

A model should be as simple as possible, but also as complex as needed to capture essential properties of the device. The HP Labs in 2008 presented a simplistic device model, which has been the basis for several other models published later. Such model, based on a real device implementation, is the

linear drift model [5] which was later developed in SPICE by Biolek *et al*. [46] and is described by equations (7) and (8). It is a current-controlled model, where the memristance is a function of the state variable *w* and the switching rate, given by (8), is proportional to the current flowing through the memristor and also depends on some device-specific material/geometrical properties such as the dopant mobility μ_v and the width of the metal-oxide structure *D*. The model considers that the semiconductor film of thickness *D* has two regions: one populated with a high concentration of dopants and thus it has a low resistance R_{ON}, and another one with low concentration of dopants and thus with a high resistance R_{OFF}. The boundary between the two aforementioned regions can be displaced based on the current passing through the device, according to (8). The state variable *w* value is bounded between 0 and *D*. **Fig. 2-4a** depicts the device model along with an equivalent model circuit. The equivalent circuit is a series connection of two variable resistors, which their value is a fraction of R_{ON} and R_{OFF}. These fractions depend on the size of the doped and undoped regions. If the doped region is dominant, R_{ON} fraction prevails while R_{OFF} fraction is lower; the opposite happens when the undoped region is the dominant one.

$$v(t) = \left(R_{ON} \frac{w(t)}{D} + R_{OFF} \left(1 - \frac{w(t)}{D} \right) \right) i(t) \tag{7}$$

$$dw(t)/dt = \mu_v \frac{R_{ON}}{D} i(t) \tag{8}$$

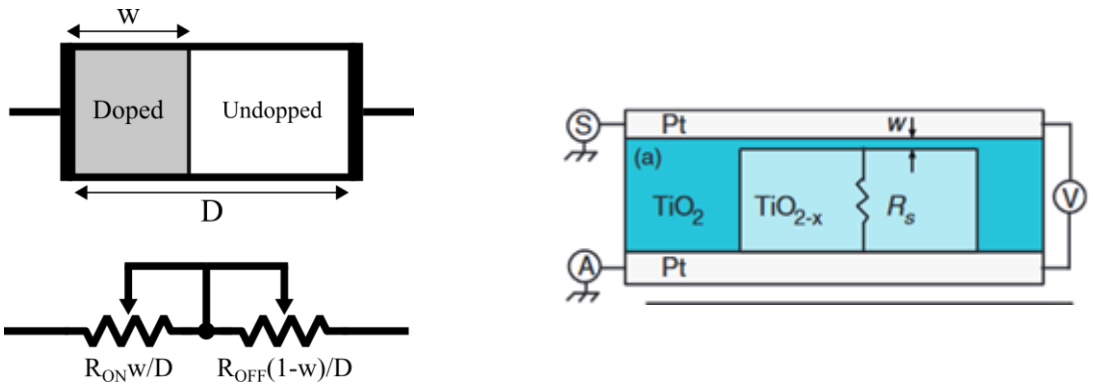


Fig. 2-4. Circuit schematics describing different model schemes. (a) HP linear drift model (adapted from [5]). (b) Simmons tunnel barrier model (extracted from [47]).

The linear relationship with charge in (8) is an assumption that actually does not consider some important effects occurring inside the devices. For instance, it was assumed that the ions are displaced freely along the device. Likewise, in the majority of modeling works, the resistance switching mechanism is not always known in all details, thus several models show limited accuracy [48], [49]. In this context, empirical window functions (WFs) have been employed to enhance such models by encapsulating a nonuniform rate of change of the resistance. WFs are dimensionless functions employed as a multiplicative factor (normally $0 \leq wf(\cdot) \leq 1$) to slow down the change of a state variable as it approaches a boundary

value. The HP Labs' model used a WF to account for certain behavioral constraints; i.e. to incorporate nonlinearity caused by very high electric fields and electric transport like in tunneling effects observed at interfaces or high-field electron hopping [5]. WFs must be 0 in the boundaries of the internal state variable whereas the value is maximum in the middle of the value range. Moreover, very similar memristor models, differing only in their WF, may exhibit qualitatively different dynamics [50]. So, the selection of a WF is of significant importance as well as it is the development of WFs for modelling of memristors.

Nevertheless, even with window functions incorporated, there are still nonlinear phenomena not covered in the linear memristor model. The exponential model proposed in [51] takes into account the electric field nonlinearity due to the electrode/oxide interface. The current in the model is expressed by (9).

$$\begin{aligned} i &= x(t)^n \beta sinh\big(\alpha v(t)\big) + \chi\big(exp\big(\gamma v(t)\big) - 1\big) \\ \frac{dx}{dt} &= a \cdot v(t)^q \end{aligned} \tag{9}$$

The hyperbolic sinusoidal function can be used to approximate quite well the *i*-*v* relationship of metal-insulator-metal (MIM) memristive structures. Therefore, it was found reasonable to explore this kind of expressions in several modeling works as detailed in [52], given that memristor devices are commonly fabricated in a MIM structure. Moreover, the Simmons tunnel barrier model [47] is an alternative to the HP linear drift-based model, consisting of a resistor R_S representing a low resistance channel and a tunneling barrier of width *w*, as shown in **Fig. 2-4b**.

$$\frac{dw(t)}{dt} = \begin{cases} f_{off} sinh\left(\frac{i}{i_{off}}\right) exp\left[-exp\left(\frac{w - a_{off}}{w_c} - \frac{|i|}{b}\right) - \frac{w}{w_c}\right], & i > 0 \\ f_{on} sinh\left(\frac{i}{i_{on}}\right) exp\left[-exp\left(\frac{w - a_{on}}{w_c} - \frac{|i|}{b}\right) - \frac{w}{w_c}\right], & i < 0 \end{cases} \tag{10}$$

While incorporating more features in new models is desirable for reliable simulations, the computer efficiency is also an important concern for computationally intensive models. Memristor models like TEAM [53] (or its voltage-controlled version VTEAM [54]) make strong assumptions in order to reduce the complexity during circuit simulations. In fact, the assumptions under consideration are the following: (a) state is not changed for stimulus below thresholds, (b) the relationship between current (voltage) and the derivative of the state variable(s) is polynomial instead of exponential.

$$\frac{dw(t)}{dt} = \begin{cases} k_{off}\left(\frac{i(t)}{i_{off}} - 1\right)^{\alpha_{off}} f_{off}(x), & 0 < i_{off} < i \\ 0, & i_{on} < i < i_{off} \\ k_{on}\left(\frac{i(t)}{i_{on}} - 1\right)^{\alpha_{on}} f_{on}(x), & i < i_{on} < 0 \end{cases} \tag{11}$$

Advanced models that rely on physical description of the devices and capture most of the device behavioral characteristics, are also available in the literature, and may be better option to perform more realistic circuit simulations. One of such model is the Stanford/PKU model [55], which is a physics-based model describing the switching mechanisms of a bipolar memristor. The model assumes a CF growth process described by a change of the CF geometry during the SET and RESET processes under various bias conditions. The core of the model is a two-dimensional description of a unique CF (see **Fig. 2-5a**), which includes both the CF gap region *g* and the CF width *w* as control variables. Therefore, the current is composed by an ohmic contribution of the region occupied by the CF, whereas the rest corresponds to the hopping current in the tunneling gap between the tip of the CF and the opposite electrode and the current in the area surrounding the CF. Most importantly, the model includes parasitic effects (shown in **Fig. 2-5b**) such as the parasitic resistance of the switching layer and the electrodes R_H and R_L, respectively, as well as the parasitic MIM capacitance C_P. Furthermore, the model supports intrinsic variation effects, such as statistical distribution of switching thresholds and resistance states, temperature dependency and dynamic current fluctuations for the RESET process, thus supporting literally all the ReRAM device variation effects known to date.

As far as operation is concerned, a positive applied voltage produces a SET process, where the oxide layer suffers a soft-breakdown; the CF is formed and the device is found at a low resistive state (LRS). In the model, the CF growth is performed in two steps: first the gap *x* is decreased to 0, and then the width *w* is increased until reach its upper boundary. On the other hand, a negative applied voltage causes a RESET process in which the CF is dissolved through ion diffusion or drift processes and the device is found at a high resistive state (HRS); this rupture of the CF is modelled by increasing *x*. Because of the completeness of this model and its verification using experimental data, it is our default choice in all simulations, unless otherwise stated. For that purpose, the Verilog-A code in [56] is adopted. The default parameters recommended for this model are listed in **Table 2-2**. Unless specified, these parameters will be used in the simulations.

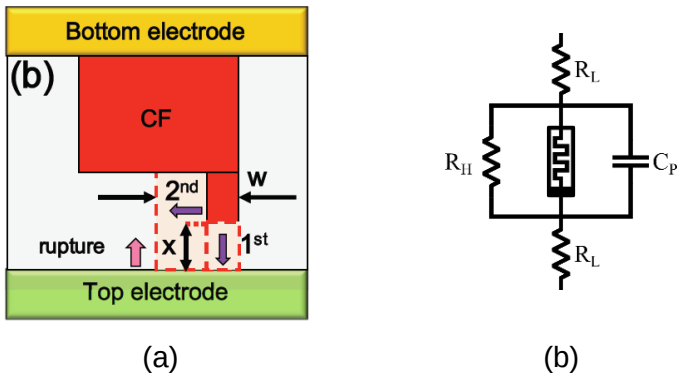


Fig. 2-5. Stanford-PKU memristor model. (a) Geometrical description of the CF in the model and evolution of the CF in SET and RESET processes. The RESET process or rupture of the CF makes the tunneling gap *x* to increase. The SET process reduces *x* until 0 and then increases *w* to its maximum value. (b) The model is completed with a circuit comprising the memristor description along with parasitic elements. Extracted from [55].

Table 2-2. Default parameters of Stanford/PKU memristor model [56].

I_0	10^{13} A/m^2	E_a	0.7 eV	Δr	0.5 nm	WCF	5 nm
g_T	0.4 nm	E_i	0.82 eV	R_{th}	$5{\cdot}10^5$ K/W	γ	1.5
V_T	0.4 V	E_h	1.12 eV	T_0	300 K	R_H	200 MΩ
ρ	19.64 μΩ·m	α_a & α_h	0.75 nm	a	0.25 nm	R_L	20 Ω
L_0	3 nm	Z & e	1 & e	f	10^{13} Hz	C_P	20 fF

2.4 Memristor Crossbar Architecture

A memristor crossbar is an assembly of memristors localized at the cross-points where a grid of vertical and horizontal nanowires cross, as shown in **Fig. 2-6a**. Usually, vertical and horizontal nanowires are also called column and row nanowires. Every memristor in a row is connected with a common nanowire, and each memristor in a column is connected through the other terminal with another common nanowire in another level. Therefore, in order to access a single memristor, access to its column and row wires is required.

The crossbar geometry has been identified as a key architecture for future resistive random access memory (ReRAM) modules based on memristors. They offer several advantages such as simplicity, high connectivity and maximum possible device integration density $4F^2$ (F is the feature size), as well as fault tolerance in the context of memory applications. However, it is also considered a key topology for the implementation of multi-layer neural networks in neuromorphic computing structures. Moreover, crossbar allows pushing toward low-power applications due to the scaling possibilities and the benefits to certain applications, e.g. the possibility to perform computation and storage operations in the same physical module (in-memory computing). Crossbar architectures are able to provide this owing to vertical stacking and the simple geometry of the architecture.

Nevertheless, crossbar structures impose certain limitations in the operation with the memristor devices. For instance, when applying a voltage pulse to a specific cross-point where a memristor resides (either for writing or reading purposes), the state of the rest of the memristors in the array could be affecting the result of the readout process, or their state could be modified unwillingly. Consequently, smart driving strategies are required to address the problem known as the "sneak current path problem". In this context, different solutions have been published so far; they can be divided into two different alternatives: those achieved owing to external circuitry or to cross-point device modifications, and those achieved by using memristors with highly nonlinear behavior.

Solutions brought by circuit-modifications involve adding more circuit elements, usually in every cross-point cell, to mitigate or cut possible current paths. In this direction, 1T1R crossbars add a transistor in series with each memristor (**Fig. 2-6b**), hence managing the circulation of current by controlling their gate

terminals. However, this solution comes with a noticeable penalty in circuit area of about $8F^2$ to $12F^2$ for planar devices, according to [57]. Another solution concerns applying Complementary Resistive Switching (CRS), where the cross-point cell is composed by two memristors connected in series with opposite polarity [58], thus mitigating the sneak path currents by increasing the resistance in these paths (**Fig. 2-6c**); in this case, logic 0 and logic 1 consist in different resistive state combinations of the two memristors. In this scenario, the memory cell shows a more complex behavior. However, both memory and digital computing applications would be benefited by this technique, as shown in [59].

On the other hand, device-level solutions contemplate the use of passive crossbars composed of memristors with high nonlinearity behavior in their current characteristics. In other words, when there is a low voltage drop across the device, the current should be very low, thus mitigating sneak path currents in an inherent manner. Only when a high enough voltage is applied directly to the memristor terminals, the current increases significantly. When using memristors of this kind, the externally applied stimulus to the crossbar array is strategically designed to take advantage of this nonlinearity in the device behavior. For this reason, the V/2 and V/3 schemes have been proposed. In brief, for these schemes the row and column of the target memristor to be written are driven with the proper voltage and ground signals, as usual. However, the rest of rows and columns of the array are not left floating but instead are driven using an intermediate voltage level (1/2 or 1/3 of the voltage applied to the target memristor), this way forcing the rest of the devices to have significantly lower voltages across them than has the target memristor.

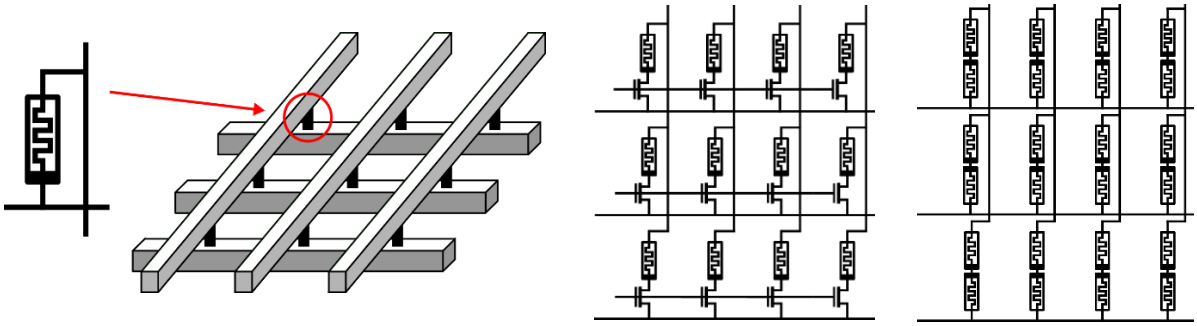


Fig. 2-6. Memristor crossbar structure and different schemes regarding the elements in the cross-points. (a) Structure of a 1R crossbar, or a crossbar with a memristor in the cross-point. (b) 1T1R crossbar (a memristors and a transistor in series) and (c) complementary resistive switching crossbar (two memristors connected in anti-series).

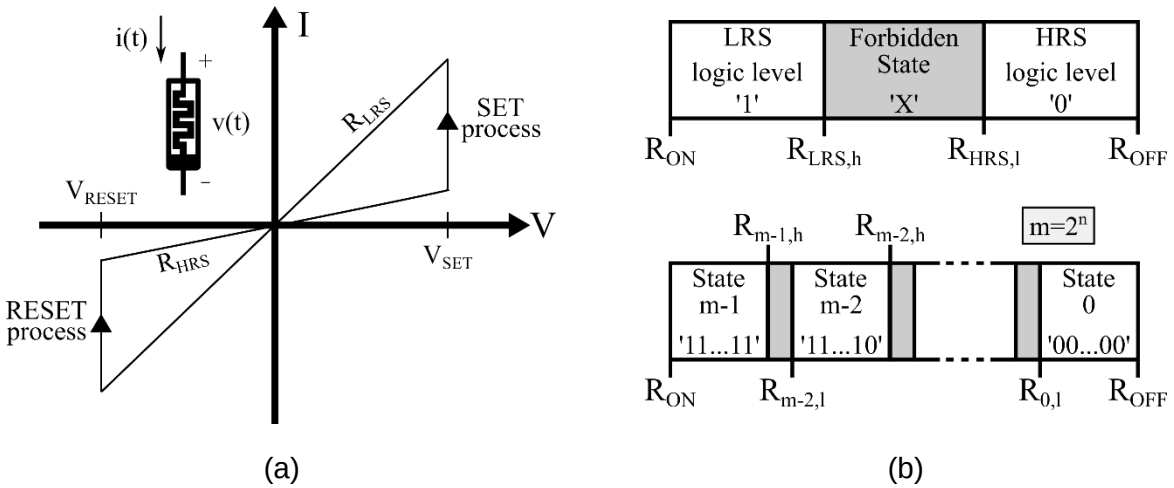
2.5 General Terms and Considerations

For the sake of uniformity and clarity, simplified memristor characteristics are assumed when explaining and discussing different applications and circuits in the following chapters of this Thesis. Such device behavior does not intend to embrace all different kinds of devices, nor to take into account all the richness of the switching dynamics. Therefore, when certain characteristics are required (because they are central to the function or impact in some circuit or system), they will be particularly specified.

Fig. 2-7a depicts the *v-i* behavior of our simplified memristor. The memristor considered here is bipolar, where the SET process occurs in the positive voltage region and the RESET process in the negative

voltage region. The actual voltages where the SET and RESET events occur are the V_{SET} and V_{RESET} voltage thresholds, respectively. The only assumptions made about the dynamics in the switching processes are about how far the memristor state evolves; when a voltage threshold is surpassed the resistance changes until it reaches the resistance limits of the device or is limited by another circuit element (e.g. a current compliance element). Hence, for further requirements about the dynamics (e.g. abrupt or gradual state evolution, delay in an application, required programming pulses to control programming operations) need an evaluation using simulations with realistic models.

About the resistance range demonstrated by the device, there are different values that are of particular interest. For example, the extreme values of the resistance range are R_{ON} and R_{OFF}, corresponding to the lowest and the highest value, respectively. In applications where memristor resistance must be modified in a gradual manner (in discrete intermediate levels), it is useful to define these levels as resistance bands rather than specific resistance values, as shown in **Fig. 2-7b**. For instance, when working with just two states (binary switching behavior), the resistance bands are usually called Low Resistive State (LRS) and High Resistive State (HRS). Each band covers part of the whole resistance range, representing a bit '1' or a '0'. Additionally, a forbidden state band separates both HRS and LRS bands. Such resistance bands are useful when dealing with sources of variability in the memristor states. When multi-bit codification per device is needed and as long as the control of resistive states is accurate enough, multiple resistance bands are defined. The scheme at the bottom of **Fig. 2-7b** shows how bands are organized in the memristor resistance range. Assuming that the data stored in memristors is *n*-bit long, there are 2^n bands separated by 2^n-1 forbidden states.



(a) (b)

Fig. 2-7. Simplified memristor and terminology used. (a) *i-v* curve of the simplified memristor indicating SET and RESET processes that lead to HRS and LRS. In (b), two examples of state codification for stored data in the memristor along the whole resistance range between R_{ON} and R_{OFF}. At the top, binary data is codified by distinguishing between two different resistance states (HRS and LRS), separated by a forbidden state gap. $R_{LRS,h}$ and $R_{HRS,l}$ are the high and low limits of LRS and HRS bands, respectively. At the bottom, *n*-bit data is codified by defining 2^n resistance state bands separated by forbidden state gaps.

2.6 Concluding Remarks

Memristor is a concept that already existed five decades ago, but the "modern" memristor (as a resistive switching device) considered today is still a hot topic. The predictions tell that these devices can

outperform conventional technologies in some aspects. Their simplicity beholds in their structure as two-terminal devices and their fabrication. Crossbar structures exploit these virtues by arranging multiple memristors in a regular and highly dense architecture. A diverse family of devices exist, each one with different switching mechanisms and performance metrics. Memristor models, which are crucial in the transference of fabrication advances to novel applications, exist to reproduce the different behaviors observed and also to cover different necessities, from accuracy to fast computation in results.

Despite their promising characteristics, there is one concern that cannot be neglected: memristor reliability. Previously, **Table 2-1** has shown the limited endurance that in general all memristors have compared with other technologies. Additionally, memristor variability is also worrisome in some of the resistive switching alternatives. Endurance limits the switching cycles the devices can stand, while variability limits the programming accuracy and the multi-bit codification. The next Chapter explains the different sources of reliability issues and how to embed their effects in models aiming to take into account in simulations.

Chapter 3 Modeling Sources of Unreliability in Memristors

Unreliability is still one of the major challenges in the development of memristor technology, which is also the reason why there are critical limitations for most of the potential future applications of memristors. Furthermore, the inherent stochastic behavior of memristors is by itself a source of uncertainty which should be taken into consideration.

This chapter discusses the three major reliability issues relevant to memristors, namely: (i) the random behavior of memristors, (ii) the switching failures, and (iii) the switching endurance. First, a brief review of the relevant literature is presented in order to explain the principal factors which affect the reliability of operation of memristors. Next, some experimental data are shown concerning the cycle-to-cycle variability in memristors. Finally, the strategy followed in the rest of this Thesis to model the variability and switching faults in circuit simulations, is thoroughly presented.

3.1 Common Sources of Reliability Issues in Memristors and Their Classification

According to IEEE, the definition of "reliability" is *the ability of an item to perform a required function under certain conditions for a given period of time* [60]. Therefore, what reliability is, in fact, it depends on the specifications defined for a target application, and also on the expected timeframe when these specifications must hold. By taking into account these two factors, three different types of unreliable behavior can be established: (i) random behavior, (ii) failure, and (iii) endurance.

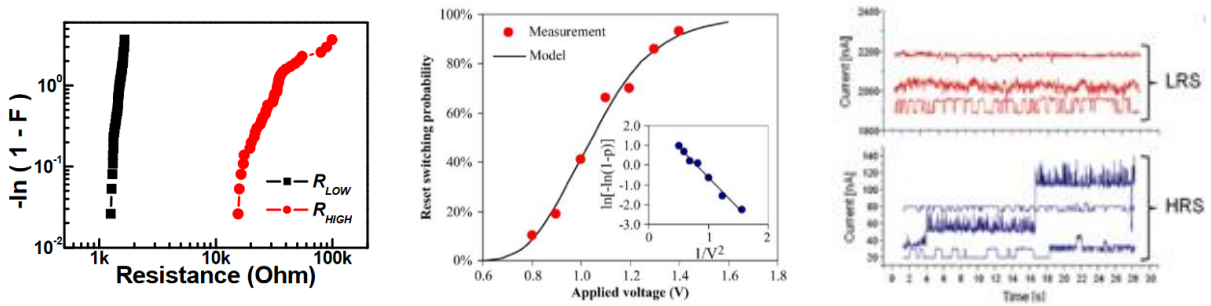
Random behavior refers to the uncertainty or lack of repeatability in a result of an action while applying the exact same conditions. In applications where accuracy is a critical parameter, randomness cannot be ignored. Three different characteristics related with random behavior are identified: state variability, stochastic switching and random telegraph noise [61].

In the context of state variability, memristors suffer from device-to-device (D2D) variations owing to the employed manufacturing processes, likewise it happens with other nanoelectronic devices as well. However, single devices also demonstrate intrinsic variability in their behavior which is often called cycle-to-cycle (C2C) variability. C2C variability is caused mainly by the inherent physical mechanism in the device. For instance, filament type suffers for considerable C2C variability due to the randomness nature in the creation and annihilation of the CF. The result is that when programming a LRS or HRS, instead of achieving the same resistance states, these states fall inside the range of a resistance distribution. For instance, in **Fig. 3-1a** several measurements of HRS and LRS after programming have been collected, depicting a distribution rather than two invariant states [62].

Switching events in memristors seem to follow a stochastic behavior rather than being deterministic, as being observed both in RESET [63] and SET processes [64]. The switching probability is related with the externally applied stimulus, e.g. voltage amplitude and pulse width for a voltage pulse, to change the resistive state of the cell. For instance, **Fig. 3-1b** shows measured RESET switching probabilities in a CuO_x device using different voltage pulses [63]; the higher the amplitude (or the longer the width) of

the applied voltage pulse, the higher the probability to observe a switching event in the device. However, an excessive amplitude also leads to more stressed devices; eventually this turns into a trade-off between the successful switching of the device state and the device endurance.

The last effect related to variability is the random telegraph noise (RTN). RTN is described as *discrete fluctuations in the current-/voltage time traces whenever the charge transport through a solid state device is controlled by the statistical capture/emission of electrons at electron trap sites or the statistical transposition of single lattice defects* [65]. Fluctuations caused by RTN may be larger in filament type devices; current paths through the atomistic scale of the CF are more affected to trapping/detrapping perturbations. **Fig. 3-1c** depicts the RTN fluctuations in current measurements in HRS and LRS after several SET/RESET processes [66]. When charges are trapped or untrapped, we observe that some discrete jumps in the current measurements according to these events.



(a) (b) (c)

Fig. 3-1. Contributions to memristor random behavior. (a) State variability observed for (extracted from [62]) (b) Measured RESET switching probabilities of a CuO_2 MIM structure versus applied voltage pulse amplitude for a fixed pulse width (extracted from [63]). (c) Random telegraph noise observed when sensing currents through memristors in LRS and HRS programmed in different cycles (extracted from [66]).

As far as "failure" is concerned, it is a condition in which a device fails to behave as it is supposed to. Permanent faults are mainly caused by defects in the manufacturing processes. Additionally, a potential exposure of the device to conditions of extreme temperature or improper externally applied stimulation (e.g. too high voltage or current) could lead to transient failures. Although there is not much information in the literature at this moment about the reasons and the occurrence of permanent failures, some information about the amount of HRS and LRS defects observed in a memristive crossbar array, can be found in [14]. The spatial distribution of faulty memristors appears to be random, as it can be observed in **Fig. 3-2a**. Furthermore, the number of switching failures is much higher for the HRS (approximately equal to 80 % of the total) than for the LRS. Apart from device-specific faults, layout defects in the peripheral circuitry may be also considered. In this context, we distinguish the work in [67] which focuses on the defects across a passive crossbar array architecture. Concretely, it considers open, bridge and short defects between different conducting elements, like column and row nanowires or even in the transistor terminals in the peripheral circuitry. Some examples both in row- and column-nanowires are depicted in **Fig. 3-2b**, as well as around the periphery in **Fig. 3-2c**.

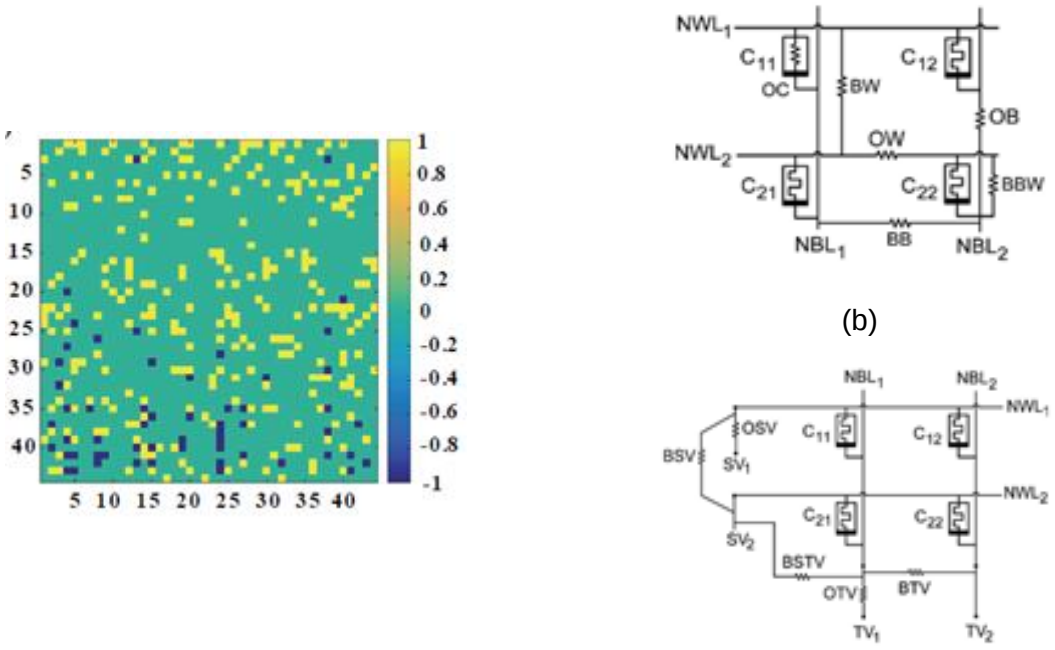


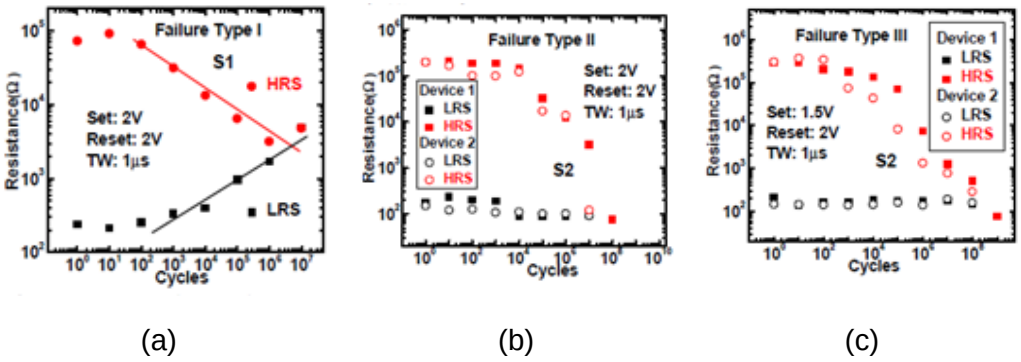
Fig. 3-2. Defects in memristor-based crossbar arrays. (a) Normalized conductance shift from the expected programmed conductance in a 64x64 memristor crossbar in [14]. The cross-points of the crossbar are represented by little colored squared tiles. Green tile refers to memristors whose conductance is properly controlled, whereas yellow and blue tiles are memristors with a sensed conductance larger or lower than expected, respectively. (b) Modelled defects inside the array of a crossbar architecture and (c) in the peripheral circuitry driving the array [67].

Generally, in the first switching cycles the newly formed devices, i.e. devices that have recently passed through the forming process, behave as expected. However, the total time while the devices will still switch state as expected, is something important to consider; this is generally referred to as device "endurance". Likewise, it happens with other types of memory cells, memristors as well allow a limited maximum number of writing cycles (where "cycle" refers to a SET followed by a RESET event, or vice versa). Owing to the stress seen by the device during the writing operations, some of the device properties gradually degrade and, at some point, the performance no longer meets the desired specifications for the target application. Concretely, such degradation has been observed in the HRS and LRS levels. For instance, in [13] the authors identified three degradation scenarios, which here are illustrated in **Fig. 3-3**. We observe that the HRS and/or LRS levels degrade due to several different causes. In each case, after a certain number of switching cycles, the HRS and LRS levels end up being indistinguishable (i.e. HRS ~ LRS).

3.2 Hands-on Experience with SDC Commercial Memristors

In order to demonstrate hands on experience in measurements of switching variability in real memristor devices, as demonstrated in [68], we collected data from self-directed-channel (SDC) memristors developed and commercialized in 16-pin ceramic DIP package by Knowm Inc. [19]. The target device was a BS-AF-W bipolar memristor with tungsten (W) dopant on a chalcogenide material. The SDC memristors [69] constitute an ion-conducting device type, a sub-class of electrochemical metallization (ECM) devices,

which uses a metal-catalyzed reaction within the device active layer to generate conductive channels (Ag ion transport routes) that contain Ag agglomeration sites, permanent under similar operating conditions. Therefore, they are different from conductively bridged (CBRAM) devices, which are based on the formation/dissolution of a conductive filament between the top and bottom electrodes in response to an applied voltage [70]. The collected data offer valuable experience on observation of variability, as shown next.



(a) (b) (c)

Fig. 3-3. Different failure mechanisms for bipolar filamentary memristors: (a) both HRS and LRS levels approach each other at an intermediate resistive level. (b) HRS degrades abruptly to LRS. (c) HRS degrades gradually towards LRS. Extracted from [13].

More specifically, the SDC Knowm memristor switching mechanism is bipolar; with a positive applied voltage above a threshold value, the device resistance may decrease to a LRS level, whereas the opposite polarity makes its resistance to rise to a HRS level. The device also has the property to phase change when higher voltages and currents are used, which is purposely avoided in the measurements as it provokes irreversible changes in the devices.

Knowm devices come fresh and they need an electroforming process to enhance their memristive behavior. The procedure to obtain electroformed memristors is next explained and the overall setup is illustrated in **Fig. 3-4a**. A slow voltage ramp rising from 0 V to 1 V was generated using the Agilent 4156C Semiconductor Parameter Analyzer and applied directly to the memristor, while fixing the compliance current to 1 µA to protect the device when it switched abruptly to LRS. Once electroformed, the experiments consisted in observing the behavior of several *i-v* cycles under the same stimulus. Concretely, the 4156C device applied a voltage sweep between -0.7 V and 0.5 V, in small steps of 2 mV at a pace of 100 ms for each step, repeatedly until 200 cycles were performed. In this second phase during the switching cycles, the current compliance was set to 10 µA, as recommended in the datasheet specifications.

Fig. 3-4b shows the obtained voltage and current data of a single memristor collected with the 4156C instrument in the form of an i-v plot. While observing this plot, it is easy to recognize the characteristic i-v hysteresis loops, although the left lobe is quite narrow. Further measurements on the same type of devices can be found in [71]. The reason behind the asymmetry in the input stimulus lies in that the SET and RESET processes are completely different. More specifically, while the SET process is abrupt and the device

apparently switches completely before the applied voltage reaches to 0.3 V, the RESET process requires a larger voltage span to complete, due to being a more gradual process with a slower resistance evolution.

There are some regions that are difficult to distinguish in the linear scale plots due to the R_{OFF}/R_{ON} ratio of memristors. Therefore, it is a common practice to present the i-v curves using the logarithmic scale for the current, as shown in **Fig. 3-4c**, where current from both lobes can be clearly appreciated and thus can be compared more easily. The negative lobe size is quite small compared to the other lobe. RESET process starts at smaller voltage magnitudes compared to the SET process. However, the transition from LRS to HRS is slower, thus requiring higher negative voltages to enable reaching to R_{OFF}. Additionally, while the SET process produces a single and abrupt current jump, the RESET process gradually drops the current in several steps. Variability seems to be worst in the RESET process, where switching events are more scattered. Finally, the current axis in **Fig. 3-4c** is converted to a resistance axis in **Fig. 3-4d**, by simply dividing the voltage and current data points. This last plot is useful to illustrate the evolution of the resistance and/or the HRS and LRS levels, with an R_{OFF}/R_{ON} ratio higher than 100. It also shows how R_{OFF} is not a purely ohmic state, although it seems to be only lightly dependent on the voltage. On the other hand, ohmic behavior for R_{ON} is not properly shown here, because of the application of the compliance current by the 4156C instrument.

Moreover, some characteristic memristor parameters may be extracted from the measured data, such as the R_{OFF}, R_{ON} values, as well as the voltages required for SET and RESET, i.e. V_{SET} and V_{RESET}. It is worth mentioning also that the validity of the parameters obtained is bound to the conditions under which the memristor(s) were formed and tested. Therefore, it is not guaranteed that these parameter values can be directly included in a model to faithfully replicate the dynamic device behavior in generic simulations for different conditions than the ones of these measurements. Nevertheless, the extracted parameters are useful to have some insight about the magnitude of C2C variability for this device. For this reason, R_{OFF} and R_{ON} values are calculated in each cycle through the current obtained at 50 mV (when decreasing) and -50 mV (when increasing), respectively. Furthermore, V_{SET} and V_{RESET} are defined as the voltages where the measured resistance slope (in logarithmic scale) is maximum. Once obtained every parameter for each cycle, mean and standard deviation is calculated for each parameter and the results are shown in **Table 3-1**. Higher variability is observed in R_{OFF} values compared to R_{ON}. The parameter that is affected the most by variability is V_{RESET}, which is something reasonably expected because of the increased randomness of the RESET process observed in the measurements.

3.3 Incorporating Reliability Phenomena in Memristor Device Models

The ability to model precisely the memristor behavior is essential to demonstrate their usefulness in several applications. However, being able to incorporate in memristor models different phenomena related to variability, is also an essential tool so as to evaluate the impact, reliability or tolerance of an application to some of these issues. Unfortunately, so far most memristor models do not include such features, so they need to be incorporated separately applying different strategies. Therefore, in this section

both memristor variability and failure modelling are discussed. On the other hand, device endurance (or degradation) modelling is not covered.

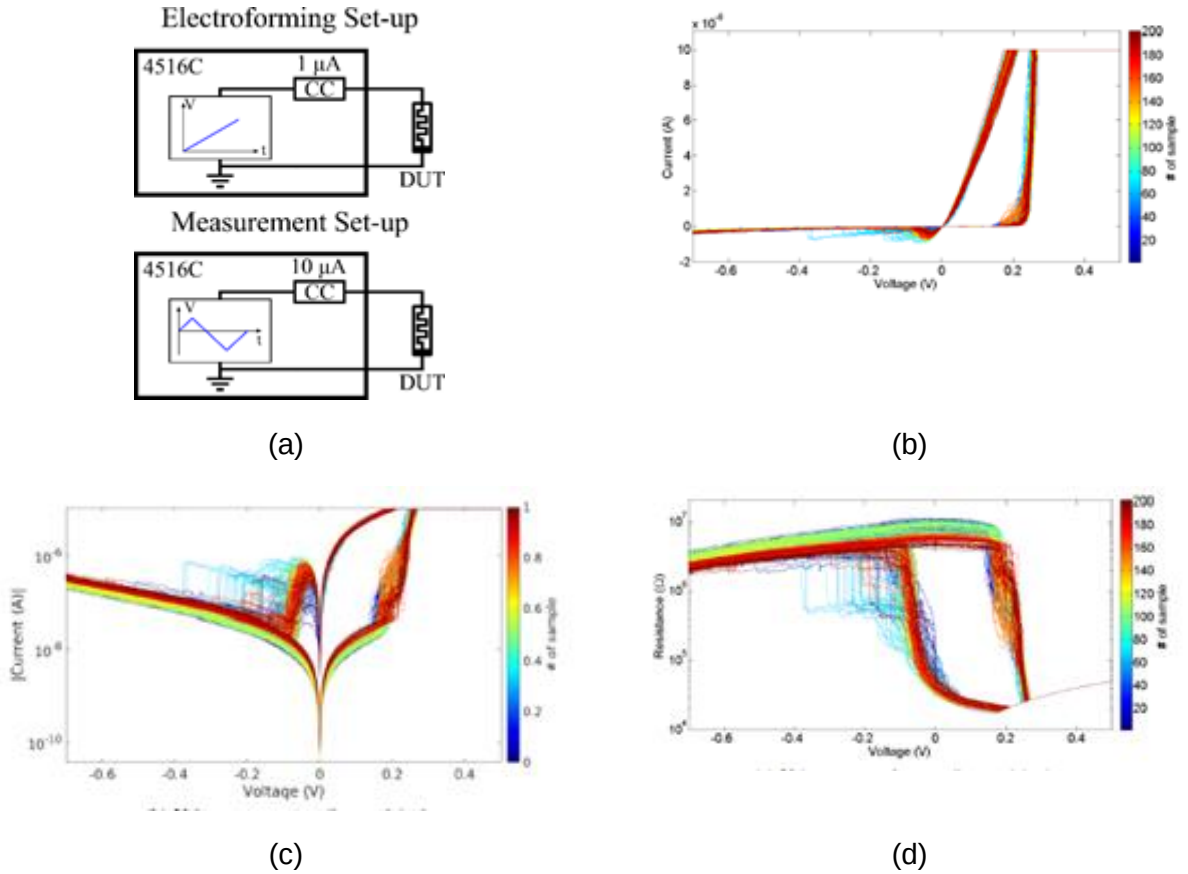


Fig. 3-4. Data measurements for *i-v* hysteresis loops on Knowm commercial SDC memristors as DUT. (a) Experimental set-up for electroforming of memristors and to perform the C2C measurements. When electroforming, the DUT is supplied with a 1 µA current-limited step voltage function that sweeps from 0 to 1 V. The measurement of *i-v* curves uses a 10 µA sweeping voltage from -0.7 V to 0.5 V. Both sweeping voltages are stair-shaped signals with a step resolution less than 2mV and a duration of 100 ms per step. (b) *i-v* curves. (c) *i-v* curves in logarithm scale. (d) *R-v* curves.

Table 3-1. Parameter extraction from measurements on SDC Knowm memristors

$R_{ON,\mu}$	29 kΩ	$V_{SET,\mu}$	206.9 mV
$R_{ON,\sigma}$	2.656 kΩ	$V_{SET,\sigma}$	32 mV
$R_{OFF,\mu}$	7.388 MΩ	$V_{RESET,\mu}$	- 117.6 mV
$R_{OFF,\sigma}$	1.742 MΩ	$V_{RESET,\sigma}$	- 80.6 mV

3.3.1 Modelling the Memristor Variability

As it has been reviewed previously, there are different factors that contribute to variability, classified into C2C and D2D variability. Although each factor can be modelled separately, doing so is a rather complicated task which is out of the scope of this work, mostly because of the need for deeper

understanding on the physical background behind each contribution. Additionally, modelling each effect separately may be impractical as it adds computational overhead in the simulation.

Our approach is based on the observation that all the contributions in variability are seen as random uncertainty of parameters. Instead of focusing on modelling each contribution separately, all contributions were wrapped up into two general terms; i.e. C2C and D2D variability, being the only random events. It is important to distinguish how each type of variability should be implemented, so next we describe this in more detail. D2D variability is permanent and implies a different behavior of the device from the very beginning of its lifetime. Hence, in simulations of any device modeled with such effects, the latter have to be applied at the instance of the netlist, or during the initialization at the beginning of the temporal simulations. On the other hand, C2C variability generates uncertainty during the entire temporal simulation, as we observed previously in measured currents that were affected by RTN. Therefore, the parameters of the model affected by C2C variability must be altered constantly. The choice of model parameters that need to be modified in order to incorporate D2D and C2C variability, will depend entirely on the selected device model.

In this context, next we show how D2D and C2C variability can be introduced in the Verilog-A Stanford/PKU model [56], which is the model chosen to be used in this study. As mentioned in the previous chapter, this model already implements some variability features. However, they are minimal leaving us a lot of space for improvements.

More specifically, the model takes into consideration C2C variability induced in the state variables *g* (the tunneling gap distance between the electrode and the tip of the conductive filament (CF)) and *w* (the CF width), which are not only determined by the change of state equations but also by random quantities $\delta g \cdot \chi(t)$ and $\delta w \cdot \chi(t)$, as shown in eq. (12) and (13). $\chi(t)$ is a random variable that follows a zero-mean Gaussian distribution with a standard deviation equal to 1, N(0,1). So, $\delta g \cdot \chi(t)$ and $\delta g \cdot \chi(t)$ are Gaussian distributions N(0,δg) and N(0,δw), respectively. It should also be considered that different memristors might demonstrate different amounts of variability, hence it is interesting to include different scales of statistical distributions for different parameters. In this direction, in addition to the aforementioned implementations, we included a factor *k* that multiplies the default standard deviations δg and δw in order to capture different levels of C2C variability. The results are new random quantities which follow a Gaussian distribution N(0,$k \cdot \delta g$) and N(0,$k \cdot \delta w$). This allows us to cope with higher levels of uncertainty observed in other devices, like the Knowm memristors which were characterized experimentally in the previous Section. For instance, **Fig. 3-5** shows the resistance distributions after 5000 cycles of programming HRS and LRS for different values of *k*. As *k* increases, the distributions are wider.

$$g = \left(\int_0^t \frac{dg}{dt} + \delta g \cdot \chi(t) \right) dt \tag{12}$$

$$w = \left(\int_0^t \frac{dw}{dt} + \delta w \cdot \chi(t) \right) dt \tag{13}$$

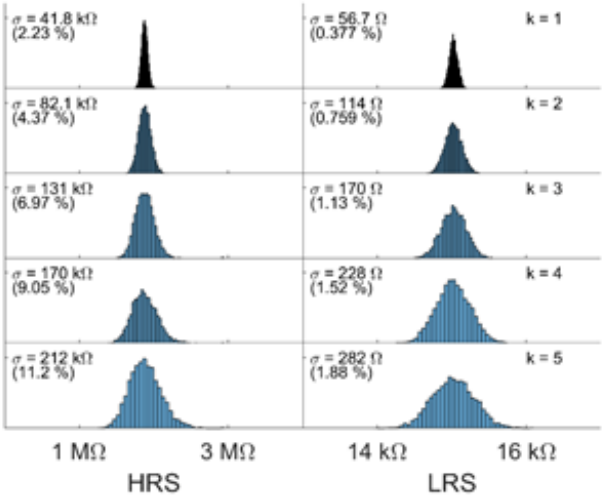


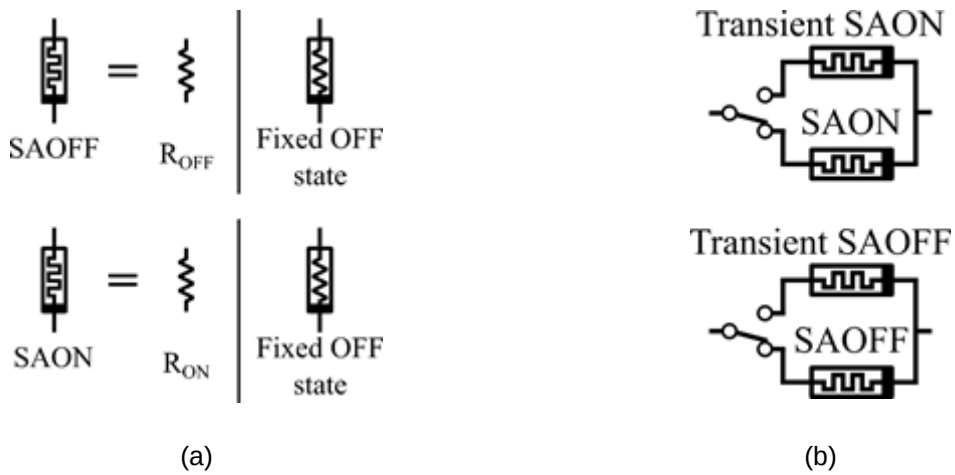
Fig. 3-5. Probabilistic distribution of 5000 total programming events of HRS and LRS using the Stanford/PKU model for different values of *k* factor. The resulting standard deviation and the percentage of standard deviation with respect to the average are shown for each distribution.

3.3.2 Memristor Fault Modelling

Fault modelling in memristor applications can be quite challenging because there are few references reporting faults in memristor devices so far. When it comes to a defective memristor, there are two possible scenarios: *stuck-at* faults and *transient* faults. Stuck-at faults have been discussed previously; a memristor might accidentally get stuck at HRS or LRS permanently. On the other hand, transient faults are important because they are not noticed initially. Instead, the device starts working properly as expected but at some point it becomes faulty, i.e. stuck-at-OFF (SAOFF) or stuck-at-ON (SAON).

The modelling of stuck-at faults is quite straightforward. We provide two different options, as illustrated in **Fig. 3-6a**. The first one concerns replacing the memristor by a linear resistor whose resistance is equal to R_{OFF} as equivalent to a SAOFF fault, and R_{ON} as equivalent to an SAON fault. This solution is convenient owing to its simplicity and does not impact the computational time in simulations. A second alternative consists in using a modified memristor model as a faulty memristor. In this case, the dynamics are removed, i.e. preventing it from changing its state. In other words, the resulting element is now a nonlinear resistor. This last solution allows preserving the non-linear *i-v* relationships observed in the device model without modifying the resistance value.

In the case of a transient fault, it is assumed that a memristor eventually appears to be in SAOFF or SAON. Therefore, the model used for such fault must allow setting the moment when the fault is actually provoked. To this end, a transient faulty memristor is modelled using a voltage-controlled switch, which switches between a functional memristor and an SAOFF (or SAON) memristor, which is modelled as explained previously. **Fig. 3-6b** shows the circuit schematics corresponding to this solution.



(a) (b)

Fig. 3-6. Two memristor faults modelled at circuit level. (a) Stuck-at-OFF and stuck-at-ON, modelled by either a fixed linear resistor or a nonlinear resistor. (b) Transient stuck-at faults, modelled with a voltage-controlled switch that commutes between a functional memristor and a faulty memristor.

3.4 Concluding Remarks

Memristors deal with defects and uncertainty phenomena, whose origin and the effects produced by them have been reviewed in this chapter. Not only are caused by manufacture limitations, but their physical mechanisms inherently cause variations in the behavior. If some of these issues are critical or not for an application, it depends on the application requirements and the memristor characteristics. For instance, if precise, fine-grained control of the memristor states is required by specification, it may be a challenge for a device with significant variability.

Furthermore, we have exposed ways of model C2C and D2D variability, as well as zero-time and transient faults. Now that all the tools required for simulation are known (memristor, memristor variability and fault models), the next step is to use them to analyze the impact of uncertainty in applications. The following chapters deal with different kind of applications and solutions to mitigate some of these unwanted effects. Concretely, memory systems are covered in the next chapter, which is one of the earliest applications where memristors have been involved, and how to detect faults in a memory cell.

Chapter 4 Fault Testing in Memristor-Based Memory Systems

Memristors are devices whose resistive state is inherently nonvolatile, thus being potential candidates to be used as memory cells for low-power purposes. Additionally, crossbar arrays are structures that take advantage of the small footprint of memristors, allowing to achieve the maximum possible integration density, thus to maximize storage density of memristive memories. Everything considered, memristor gathers enough promising characteristics so as to be considered a key memory device candidate for future applications. Unfortunately, the reliability issues discussed in the Chapter 3 remain one of the major obstacles for this technology to make it to the market as a real and competitive alternative. Besides expecting better and more reliable devices that alleviate these drawbacks, finding solutions to detect, correct and/or tolerate errors in memristive memories is a mandatory issue. This Chapter presents a review of the possibilities that memristors offer as part of memory systems. Next, a novel strategy to test faulty cells in memristor-based memory arrays is presented (published in [72]). Such strategy spans the whole space from the design of the memory circuit to the algorithm that detects different kinds of known faults for this type of devices.

4.1 Potential of Memory Systems Based on Memristors

The suitability of memristor as a potential candidate device compared to other emerging technologies, has been projected in the near past according to [2], considering in such comparison both conventional memories (DRAM, SRAM, NOR and NAND) as well as emerging memory devices (FeRAM, STT-RAM, PCRAM and RRAM). Some predictions for 2020 highlight the smallest cell area of memristors ($4\ F^2$), as well as the lowest writing voltages (< 0.5 V). Read and write/erase times (about 1 ns and < 1 ns, respectively) are much better than in other emerging technologies but slightly slower compared to performance of conventional SRAM (as low as 0.2 ns). The major challenges are situated in the write endurance of the cells, where DRAM or SRAM may prevail by 4 orders of magnitude, and certainly the memristors are far from being placed in the first band among other emerging technologies.

Furthermore, achieved memory density may reach beyond what is allowed by the physical footprint in 2D structures. This, owing to multi-level storage in memories which store more than one bit of information per memory cell (i.e. per memristor), allowing higher quantities of stored data in the same crossbar array. To this end, careful distribution of the achieved resistance window is required, separated in bands corresponding to different stored combinations, along with controlled programming processes and variability of the different resistive states. In addition, a higher HRS to LRS ratio allows more states to be allocated. For instance, recently it was reported that for some devices it is possible to store more than 6 bits of information per memristor, using a proper programming methodology [73]. On the other hand, 3D-stack technology allows to integrate two or more layers of crossbar arrays in a stacked manner, thus scaling further down from planar cell area $4F^2$ to $4F^2/N$, if N is the number of total stacked crossbar arrays (layers) [74].

4.2 Online Testing Strategy for Faults in 1T1R Crossbar Memory

This Section presents a procedure to test a one-transistor-one-memristor (1T1R) crossbar-based memristive memory array during normal operation of the system, published in [72]. Such procedure focuses specifically on the different malfunctions of the memory cells. To this end, it includes a simple fault model for the 1T1R cell, as well as the interferences between cells when faults occur.

All the presented results and data in this Section were obtained by means of simulations only, conducted with the Cadence Virtuoso ADE software tool. The memristor model used was the Stanford/PKU RRAM model implementation in Verilog-A [56], using the all the default values listed in **Table 2-2** in Chapter 2. Regarding the transistor, a 65 nm predictive model was adopted from [75].

4.2.1 Read and Program Operation to a Single Cell

As a first step in the design process, we focus on the operation of the 1T1R cell, and the strategies to follow in order to read and write information from/to it. In this direction, the cell behavior must be well-studied and also the stimulus and circuitry used for these purposes must be chosen appropriately. The circuit schematic of a 1T1R cell is shown in **Fig. 4-1**. A 1T1R memory cell contains a memristor in series with a transistor.. The cell essentially consists of a memristor connected in series with an NMOS transistor which acts as a selector device. Hence, the 1T1R cell has three terminals: the top electrode of the memristor *r*, the source terminal of the transistor *c*, and the gate terminal of the transistor *sel*. Labels *r*, *c* and *sel* we selected so as they stand for *row*, *column* and *select*, in reference with the crossbar architecture and the corresponding function they are used for.

The transistor allows the cell to be selected or unselected if *sel* is '1' or '0', respectively. Only when selected, the transistor is conducting and the cell can be read or programmed. On the other hand, by keeping the select transistor cut-off in an unselected cell, we prevent any read and program operations from taking place in that cell, but most importantly, we also prevent sneak-path currents from passing through it.

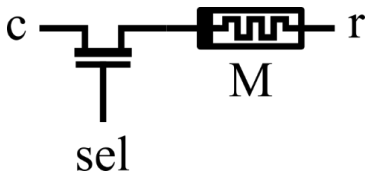


Fig. 4-1. A 1T1R memory cell contains a memristor in series with a transistor.

The read-out operation is such that makes the state of the memristor accessible to other subsystems, without disturbing the stored data. Reading usually means that the resistive state of the memristor is converted to another magnitude able to be sensed, such as voltage or current. So, the requirements for a proper design of the read operation are as follows: (a.) *the operation must not disturb the memristor state*, and (b.) the applied circuit must be able to distinguish between different levels of resistance, assuming that

data is codified in discrete resistive levels within the entire valid resistance window of the device [R_{MIN}, R_{MAX}].

There are different techniques proposed in the literature which could be applied to sense the memristor state. We particularly used the scheme shown in **Fig. 4-2**. Schematic of the circuit used in the read operation of a single memristor cell. *RD_OUT* is either '0' or '1' if R_M is HRS or LRS, respectively., which concerns a comparator that senses the difference between a reference voltage, created via a voltage divider, and a read voltage from a voltage divider between a resistor and the target memristor. Assuming that we only need to distinguish between two states, namely '0' and '1', a comparator is sufficient and is considered a fast enough solution. The read operation is performed by applying a voltage pulse across the voltage dividers. R_{REF} acts as a threshold resistance to evaluate if the value of the resistance of the memristor R_M is HRS or LRS. If R_M is higher than R_{REF}, *RD_OUT* outputs a '0' logic level. Otherwise a '1' logic level is given. In order to carry out this state conversion during read-out without altering the stored cell content, it is required that the applied voltage pulse complies with certain properties. For example, the pulse amplitude V_{READ} needs to be low enough, so that the voltage drop on the memristor terminals is not high enough to cause any significant change in the state of the memristor. Additionally, the pulse width must be only long enough to cope with the parasitic capacitance of memristor, transistors and nanowires. Taking into account all these specifications, in our circuit design the reading circuit used a resistor R_{REF} = 200 kΩ and a voltage pulse whose amplitude was V_{READ} = 1 V and its duration was 10 ns.

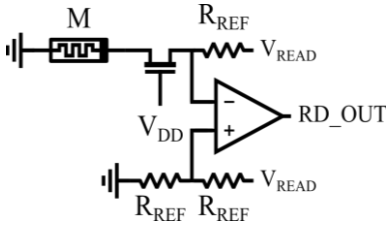


Fig. 4-2. Schematic of the circuit used in the read operation of a single memristor cell. *RD_OUT* is either '0' or '1' if R_M is HRS or LRS, respectively.

The goal of programming a memristor is to precisely set the device state into a HRS or LRS. It is worth repeating here that finding proper conditions for a reliable programming performance is a key factor. Moreover, the programming can be dealt with as if it were two operations instead of one: i.e. the operation to program a HRS state and a LRS state, respectively. Each one has different requirements as the SET and RESET mechanisms are not exactly the same.

Fig. 4-3a shows the general programming schemes used in each case. On the left side, for the LRS programming scheme, a positive voltage pulse with amplitude V_{ON} is applied to the memristor. Furthermore, an additional series transistor is used in order to limit the current passing through the memristor for protection purposes because too high currents could damage the device once it abruptly switches to LRS during a SET process. In this direction, the gate voltage was selected V_{GCC} = 1.28 V such that a 100 μA compliance current was achieved. On the right side, for the HRS programming scheme, a negative voltage

is applied to the memristor (note the different polarity of the memristor in this topology) by means of a voltage pulse with amplitude V_{OFF}. Unlike in the previous case, here there is no need for an extra transistor to limit the current in the RESET process because, once the resistance of the memristor starts to increase, at the same time the current will decrease and this way will never cause damage to the device.

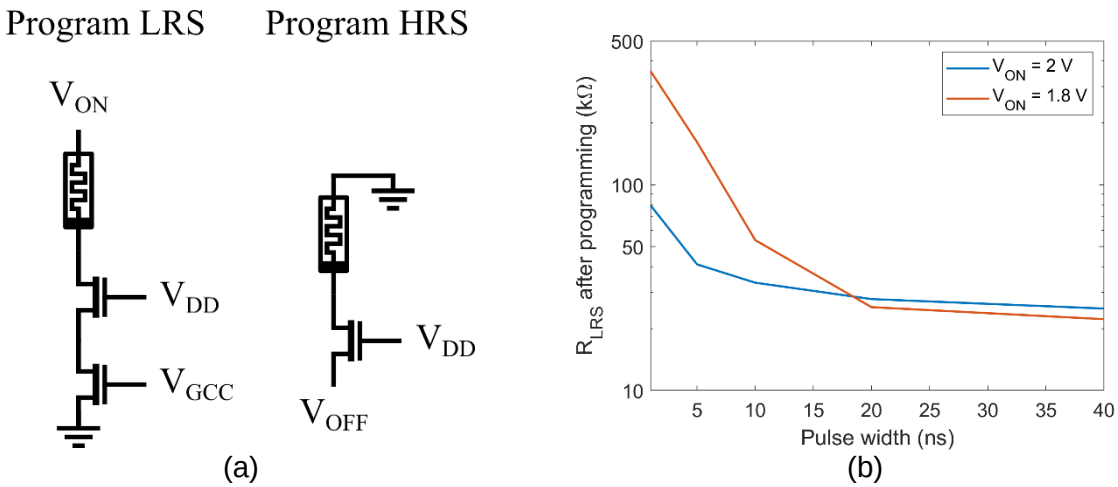


Fig. 4-3. (a) Programming schemes for HRS and LRS, concerning a memory cell that stores 1-bit of information. (b) The LRS achieved after a SET operation considering different width and amplitude V_{ON} of the applied voltage pulse.

In order to demonstrate how the applied programming voltage pulses should be tuned, several LRS programming operations were performed using two different V_{ON} amplitudes, namely 1.8 V and 2 V, while probing different pulse widths. After each programming event, a read operation was held. Read-out results are shown in **Fig. 4-3b**. We observe that with lower amplitude V_{WRITE}, larger pulse widths are needed in order to reach the same LRS state. Nevertheless, we decided to use the same voltage pulse for both programming schemes in order to simplify the circuitry; $V_{ON} = V_{OFF} = V_{WRITE} = 2$ V with 120 ns width pulses was chosen.

Once the read and the read/program operations were designed, all circuits presented so far were combined in a unique complete peripheral circuit for read/program operations to a single cell, as depicted in **Fig. 4-4**. This circuit is able to perform all operations described previously, by enabling or disabling some of the devices and blocks of the peripheral circuitry. More specifically, two inverter drivers provide 0 V and V_{WRITE} voltage levels available on both cell terminals, needed for HRS/LRS program operations. When input $d = 0$, then r and c nodes are V_{WRITE} and 0 V, respectively. Note that one driver is a tri-state inverter and the aforementioned scenarios only occur with $en = 1$. Setting $en = 0$ disconnects the output of the tri-state inverter from the voltage supplies, resulting in a high-impedance output node. Consider that it is necessary that the output is kept in high-impedance when the PMOS transistor connected to node c is conducting, i.e. when $read = 0$, to not interfere with the V_{READ} voltage. The rest of the circuit is adapted from **Fig. 4-2** with the exception that R_{REF} are implemented using PMOS transistor in the saturation region with a properly tuned gate voltage V_{GP}, to achieve channel resistance approximately equal to 200 kΩ. This implementation has the advantage of disconnecting V_{READ} voltage (setting read = 0) from node c when the

tri-state inverter output is enabled (*en* = 1). Using this scheme, read and program operations were performed by setting signals to proper levels, as they are detailed in **Table 4-1**.

Table 4-1. Signal setting for each operation in a single 1T1R cell

OPERATION	en	$\overline{en}$	d	$\overline{d}$	$read$
READ	0	1	1	0	1
PROGRAM HRS	1	0	1	0	0
PROGRAM LRS	1	0	0	1	0

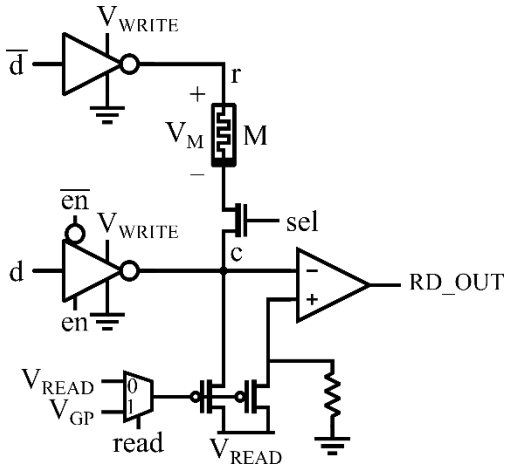


Fig. 4-4. Complete peripheral circuitry for read/program operations performed to a single 1T1R cell.

4.2.2 Overview of target memory system

The complete memristor-based memory system considered in this study is shown in **Fig. 4-5a**, and comprises of the 1T1R crossbar array described previously, along with the peripheral circuitry shown in **Fig. 4-4**. As its core, the system has a 1T1R crossbar of size *m* x *n* (undetermined). The row nanowires are labelled r_1, …, r_m; they are connected to the top electrode of the cross-point memristors. Likewise, the column nanowires c_1 … c_n connect to the source terminal of the select transistors. Additionally, all the gate terminals of the select transistors found in the same row (the same memory word) are connected between them by select nanowires labelled sel_1 … sel_m. Using this notation, a cell is identified as (*i*,*j*) by its position in the array defined in terms of the row r_i and column c_j where it is connected.

The peripheral circuitry consists of signal drivers and a sensing circuit. Regarding the signal drivers, there are *m* row inverter drivers plus *n* column tri-state inverters. The inverters provide the V_{WRITE} voltage and ground that are required in program and read operations. Each inverter input terminal has a different input, d_{r1} … d_{rm} and d_{c1} … d_{cm}. The *en* signal for the column tri-state inverters is shared, making it possible to enable all of them simultaneously. On the other hand, the sensing circuit contains *n* amplifiers, each one sensing one column nanowire, whereas the R_{REF} resistors shown in **Fig. 4-2** are here implemented using PMOS transistors. The reference voltage which is obtained from the voltage divider connected to the positive input of the amplifier, is shared among all the amplifiers.

All the memory cells belonging to the same row in the array are selected when enabling the corresponding *sel* signal (sel_i = 1). The same occurs with the shared *en* signal for the column tri-state inverters and the shared wire for the PMOS gates in the sensing circuit. This configuration for the crossbar allows performing read, program HRS and LRS operations in every memristor of a row instead of a single cell, in a single step for each operation. The array operations, which extend the operations previously described in **Table 4-1** for a single cell, and the set of signals required for each operation are listed in **Table 4-2**. All operations now use input arguments to determine the location of the cells to be read or written/programmed. For instance, READ ROW *i* gives in the outputs the stored state in the *n* cells of row *i* through the *n* amplifiers outputs, given the input argument *i* which is the target row number ($1 \leq i \leq m$). **Fig. 4-5b** depicts in detail the logic levels of all the signals applied to the system to perform a READ ROW operation. The input of each column inverter is fed with an 'X' logic level, which means a 'don't care' value, i.e. the value is not important due to the inverter high-impedance state (*en* = 0). PROGRAM HRS b_n ... b_0 ROW *i* and PROGRAM LRS b_n ... b_0 ROW *i* operations program the memristors found in the row *i* at HRS or LRS respectively. The cells to be programmed are indicated by the input arguments b_n ... b_0. A b_j = 1 means a request to program the cell in position (*i*, *j*), whereas b_j = 0 means the cell (*i*, *j*) to keep its current state. In order to make sure that the memristors that are not supposed to change indeed keep their state, the voltage drop across their terminals must be negligible. Hence, the column and row driver must output the same voltage levels. **Fig. 4-5c** and **Fig. 4-5d** show some examples of the execution of PROGRAM HRS and LRS operations in a particular row within a 5x5 crossbar, respectively. Again, 'X' logic levels appear this time supplied in non-selected rows because transistors in those rows already prevent conduction through these cells.

4.2.3 Cell faults considered in the circuit

The proposed online test strategy accounts only for faults occurring in the memory cells. Because the cell is comprised of both transistor and memristor devices, faults in either of the devices are considered. In Chapter 3, memristor faults M_{SAON} and M_{AOFF} were already described. Regarding the select transistor, two faults are taken into account: when the transistor is stuck-at-on (T_{SAON}) and stuck-at-open (T_{SAOPEN}). When a cell is faulty with a T_{SAON}, there is a short-circuit between drain and source. This case is modelled at circuit level by substituting the faulty transistor with a resistor R_{SAON} which represents a very low resistive connection and practically connects directly the memristor to the column nanowire of the crossbar. This way, the cell can be programmed and read, as it is kept permanently selected. On the other hand, a T_{SAOPEN} fault means that the transistor is always cut-off. In this case, we model such fault by substituting the transistor with a high value resistor R_{SAOPEN}. Both types of faults imply the loss of capability to deselect the cell. However, whether read and program operations are feasible or not, it actually depends on the fault type. The summary of all the faults considered is given in Table **¡Error! No se encuentra el origen de la referencia.**. Another relevant assumption in this study is the single fault scenario, i.e. the case of having only one fault at any moment in the entire crossbar. Next, the detection of every fault is discussed in order to elaborate a strategy to enable detection of faults in a 1T1R crossbar-based memory system.

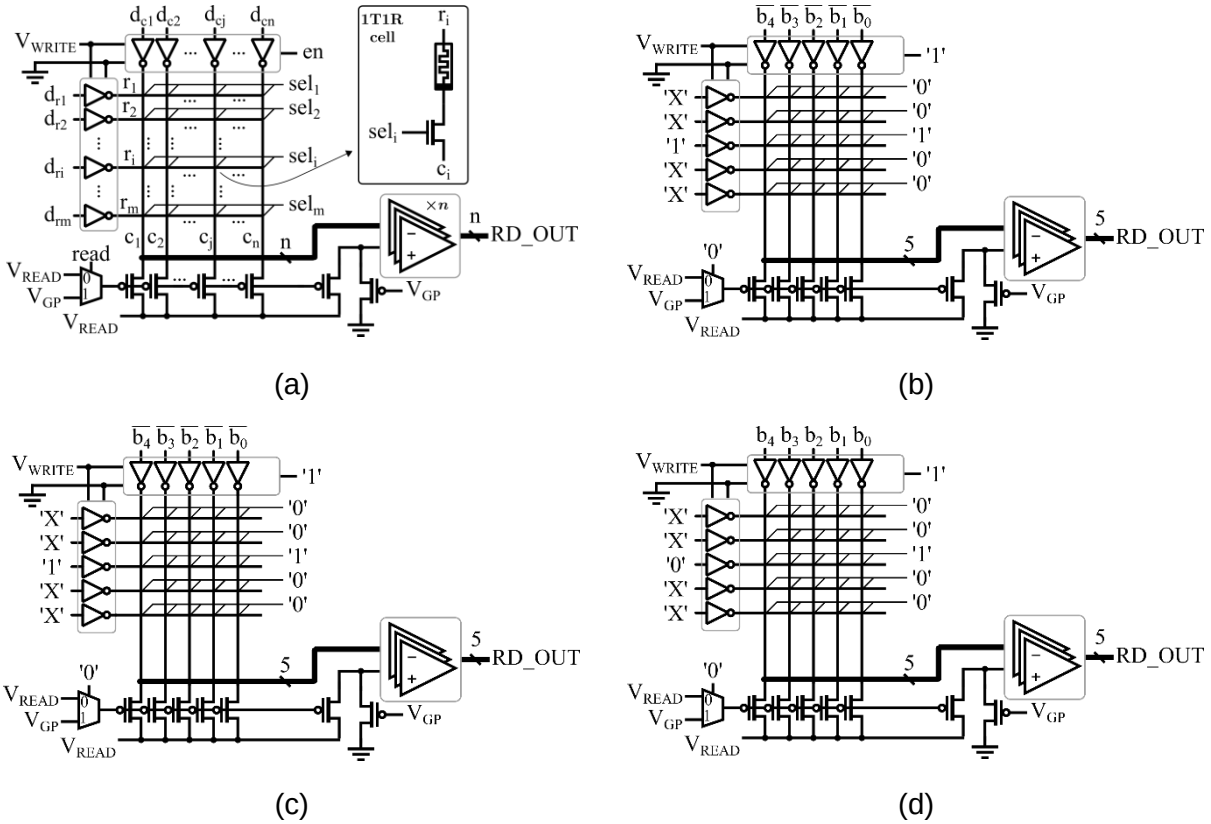


Fig. 4-5. Schematics of the target 1T1R crossbar memory system. (a) The complete memory system overview. (b) READ ROW operation. (c) PROGRAM HRS operation to row 3. (d) PROGRAM LRS operation to row 3.

Table 4-2. Signal setting for each operation in a single 1T1R crossbar memory system.

	en	*read*	$d_{r1} \dots d_{ri} \dots d_{rn}$	$d_{c1} \dots d_{cn}$
READ ROW *i*	0	1	1 … 1 … 1	X … X
PROGRAM HRS $b_{n-1}...b_0$ **ROW** *i*	1	0	X … $1^{(i)}$ … X	$\bar{b}_{n-1} \dots \bar{b}_0$
PROGRAM LRS $b_{n-1}...b_0$ **ROW** *i*	1	0	X … $0^{(i)}$ … X	$b_{n-1} \dots b_0$

Table 4-3. Single faults considered in 1T1R memory cells.

Fault	**Name**	**Description**
Transistor stuck-at-on	T_{SAON}	Channel remains conductive; low resistance R between drain and source
Transistor stuck-at-open	T_{SAOPEN}	Channel remains cut-off; high resistance R between drain and source
Memristor stuck-at-ON	M_{SAON}	Memristor kept in LRS; low resistance R_{ON}
Memristor stuck-at-OFF	M_{SAOFF}	Memristor kept in HRS; high resistance R_{OFF}

4.2.3.1. Detection of T_{SAOPEN} and M_{SAOFF} faults

The detection of T_{SAOPEN} and M_{SAOFF} is quite straightforward. Both faults make the 1T1R cell to be sensed as HRS, either because the transistor is cut-off (exhibiting a very high channel resistance) or the

memristor is stuck-at a HRS. These resistance values are higher than the selected R_{REF}, hence sensed as HRS in a read operation independently of the state programmed in the memristor. Therefore, by programming a memristor to LRS and then immediately reading it, it is easy to detect a T_{SAOPEN} or M_{SAOFF} fault if a HRS is sensed instead of the expected LRS

4.2.3.2. Detection of M_{SAON} faults

Similarly, M_{SAON} faults are also easy to detect. In fact, they follow the opposite behavior and detection than what we described for M_{SAOFF}. More specifically, in this case the 1T1R cell is sensed as LRS after a read operation regardless of any previous programming operations to store a particular state. The detection of this fault is complementary to the T_{SAOPEN} and M_{SAOFF} detection; i.e., the cell is programmed to HRS and then read, revealing a M_{SAON} fault if the read operation senses a HRS instead of the expected LRS.

4.2.3.2. Detection of T_{SAON} faults

Unlike the rest of fault types, T_{SAON} detection follows a completely different strategy. In this case, the issue is the lack of control for accessing the 1T1R faulty cell; concretely, it is always selected. Consequently, it can be successfully programmed to HRS or LRS and read without raising any inconsistency. Nevertheless, since it is always selected, unfortunately it can interact with other cells and operations.

To understand the interference of the T_{SAON} fault, **Fig. 4-6** depicts a scenario consisting in a 3x3 crossbar array with a single T_{SAON} fault located in (2,2) position. The example also assumes that the memristor in the faulty cell is in LRS. The system aims to read the content in row 1, i.e. the operation READ ROW 1. Cells in row 1, (1,1), (1,2) and (1,3), have their memristors in states HRS, HRS and LRS, respectively. However, by applying the signal setting in **Table 4-2**, not only cells in row 1 are sensed but also (2,2) is sensed due to the current path that the T_{SAON} fault allows and the 0 V supplied at r_2 node. Green current paths mark the expected sensing currents while the red one is the current path allowed by the T_{SAON} fault. Note that in column 2 the read circuit senses both cells (1,2) and (2,2) in parallel, showing an equivalent resistance similar to a LRS of a memristor. The outcome in the read operation is cells (1,1) and (1,3) are correctly senses as '0' and '1', respectively, while cell (1,2) is wrongly sensed as '1'. Although this example covers the READ ROW 1 operation, READ ROW 3 outcome is the same: the sensing of cell (3,2) is also interfered by cell (2,2), hence cell (3,2) reading may be misleading.

Before introducing the method to detect a T_{SAON} fault, there are two other situations to contemplate. The first one is related to the previous assumption in which the memristors cell (2,2) is in LRS. Indeed, if the memristor cell with the T_{SAON} is in HRS, an incorrect evaluation is not necessarily true as the read circuit may be able to distinguish between a memristor in LRS and two parallel memristors. The second situation is about performing the READ ROW operation in the row where the T_{SAON} fault is located. In this case, the output is correct as there are no other faulty cells interfering (recall the single fault assumption).

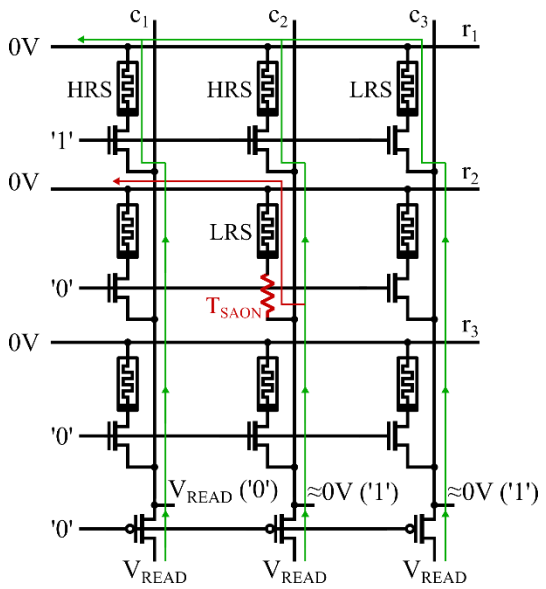


Fig. 4-6. Example of a 3x3 crossbar and the interference of a T_{SAON} fault in cell (2,2) while performing a READ ROW operation. The content of the cells (1,1), (1,2) and (1,3) is HRS, HRS and LRS, respectively. The operation READ ROW 1 should lead to '0', '0' and '1' read output. However, the interference of the faulty cell (2,2), which in addition its memristors is in LRS, modifies the read output in the column 2. Interference cause (T_{SAON} modelled fault) and effect (current sensing cell (2,2)) are marked in red.

These three situations give hints that will help to develop a method to detect a T_{SAON} fault in an unknown cross-point of the crossbar. The approach is to exploit the interference seen in **Fig. 4-6**, i.e. misleading readings in different rows localized at the same column when the content of the cells is previously known. The detection is possible by following the steps illustrated in **Fig. 4-7**, using as an example the 3x3 crossbar shown in **Fig. 4-7a**. Each step is accompanied with the operation that is being performed, a diagram of the crossbar state and an auxiliary external memory for the T_{SAON} fault detection. In the crossbar, the cells are represented by dots at the cross-points of row and column nanowires. The border color of the dot indicates if the cell is selected (blue), suffering a T_{SAON} fault (red) or non-selected (black), while the inside color means the state of the memristor in the cell (orange for HRS and green for LRS). Regarding the auxiliary external memory, its size matches the crossbar. It stores the results of the readings. When a space is blank, it means that the content is not meaningful for the fault detection process.

As a first step, we want to program to LRS the memristor of the cell with the T_{SAON}, although its localization is unknown. Because of the lack of controllability in the selector device (it is always turned on), the fastest way to program the faulty cell is to slightly modify the PROGRAM LRS ROW operation. Recall that the output of all the unselected row inverters is 'X', meaning there can be either '1' (V_{WRITE}) or '0' (0 V). If the 'X' is '0', as in the selected row, the faulty cell can be programmed to LRS. Of course, to not miss the column where the faulty cell is, all the row must be programmed to LRS, i.e. perform a PROGRAM LRS 11…11 ROW *i*.

The second step involves the HRS programming of the row we are about to read. Unlike the first step, the programming of the faulty cell must be avoided, which is already programmed at LRS. In this case, this is also achieved by modifying the 'X' values in the PROGRAM HRS ROW operation. However,

in this occasion the ‘X’ is assigned a ‘1’, which is the opposite logic value that is supplied to the selected row. Hence, when the operation PROGRAM HRS 11…11 ROW *i* is ordered all memristors in row *i* are switched to HRS but any other memristor in other rows show a negligible voltage drop. **Fig. 4-7c** depicts this step, where only the cells in the selected row 1 are programmed to HRS.

The third step consists in reading the contents of the row and store them in the external memory. Expecting not faults, the reading output should be ’00…00’. However, if a cell with T_{SAON} fault is present in another row, that cell will interfere in the reading operation causing a misleading output ‘1’ instead of an expected ‘0’. The situation is the same as shown in **Fig. 4-6**, and in the current example is illustrated in **Fig. 4-7d**. These three steps are repeated for each row in the crossbar. For instance, the state of the crossbar after reading row 2 and row 3 in the example is shown in **Fig. 4-7e** and **Fig. 4-7f**, respectively. Because the fault is allocated in row 2, there is no interfere from other rows in this reading and the results is ‘000’. The reading in row 3 is exactly the same as in row 1.

After all steps are completed for all rows, the readings stored in the external memory are checked. A pattern consisting in a column full of ‘1’s and a single ‘0’ (while the rest of columns are ‘0’) indicates a T_{SAON} fault, as shown in **Fig. 4-7f** (marked in red). Concretely, the single ‘0’ in that column comes from the cell suffering the fault.

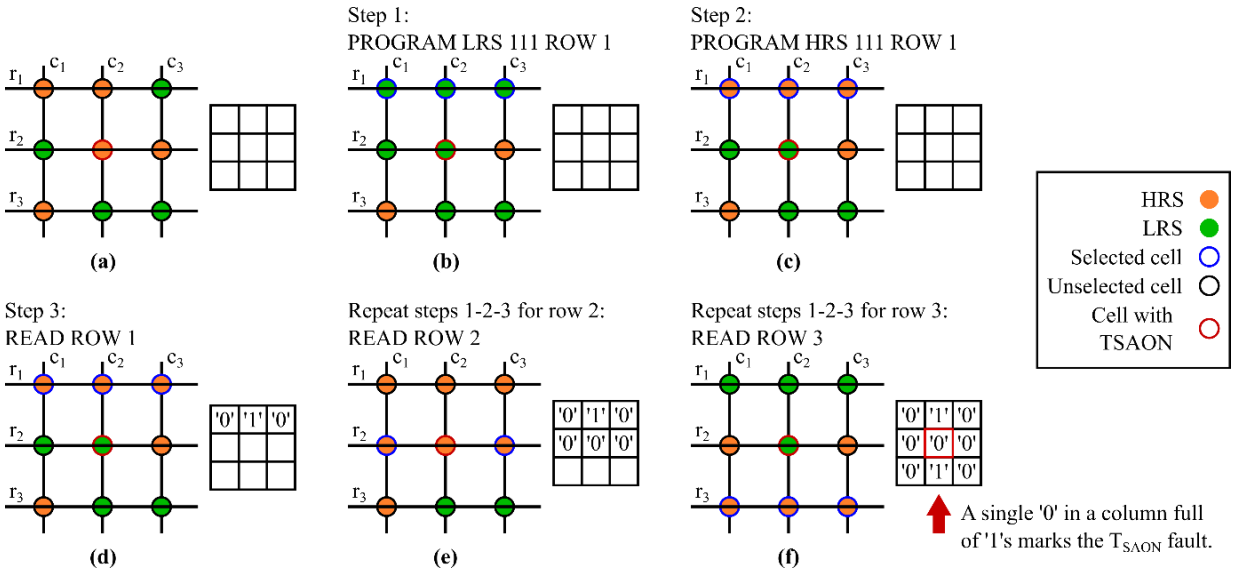


Fig. 4-7. Method for detecting a single T_{SAON} fault in a cell using as an example a (a) 3x3 crossbar with cells already storing some random data. Next to the crossbar, a grid represents an external auxiliary memory that stores read outputs. The steps for the methods are (b) a PROGRAM LRS 111 ROW 1 operation (in which the cell (2,2) is also switched), (c) a PROGRAM HRS 111 ROW 1 operation, (d) a READ ROW 1 operation. These three steps are repeated for the rest of the rows. For instance, system picture after reading row 2 and row 3 are shown in (e) and (f), respectively. After the last row is read, the memory content is used to detect a T_{SAON} fault in cell (2,2), showing the content ‘0’ in a column full of ‘1’s.

4.2.4 A novel online test strategy

Combining all the detection methods previously explained, a compact strategy that checks the memory for all the considered faults is presented. The online test strategy presented here was inspired by built-in self-test (BIST) strategies [76] and constitutes a robust process capable of checking all cells in the 1T1R crossbar and of detecting different types of faults (listed in **Table 4-3**), given the assumption of a single fault in the system. According to the proposed testing concept, some known states are programmed in

certain cells and are read immediately afterwards. Any inconsistency found between the expected states previously programmed and the read-out results are gathered, being valuable information in order to determine if there is any faulty cell.

The complete procedure for the proposed online testing method is shown in the form of pseudo-code in **Algorithm 4-1**, mixing both specific operations of the crossbar memory as well as logic operations involving data stored in external memory elements. Regarding these memory elements, two of them are present in the system: a register named *s* (of size *n*), and an array named *a* (of size *m* x *n*), serving for different purposes explained later. More specifically, the algorithm performs the following steps at each row *i* of the target crossbar array:

1) Read the content of row *i* and store their states temporarily in register $s = s_{n-1} \dots s_0$ (line 2).
2) Program all memristor in row *i* to LRS. as in the example of **Fig. 4-7b** (line 3).
3) Read again row *i*, store output in the auxiliary memory *a*, and check if the content of the entire row is '1' by reading again row *i*. A cell sensed as a '0' in the read output is successfully detected to suffer a M_{SAOFF} or T_{SAOPEN} (lines 4-9).
4) Program all memristor in row *i* to LRS (line 10). No other cells (even a cell with a T_{SAON} fault) is programmed (as shown in example **Fig. 4-7c**).
5) Read again row *i*, store output in the auxiliary memory d and check if the content of the entire row is '0' by reading again row *i*. A cell sensed as a '1' in the read output is may be interpreted as a cell suffering a M_{SAON} fault or an interference (recall **Fig. 4-6**) caused by another cell with T_{SAON} fault (lines 11-16).
6) Retrieve previous content of row *i*, which is stored in register *s*, by performing LRS and HRS programming operations in row *i* according to the data stored (lines 17-18).
7) By consulting the auxiliary memory, *a* check for a column with *n*-1 cells interpreted as faulty (a_{ij} = '1') in the step 5. If so, the only cell not detected as faulty is actually the one faulty (exemplified in **Fig. 4-7f**); the other cells in the columns are not faulty, when sensed they were interfered by the faulty cell. Otherwise, the any detected fault is due a M_{SAON} fault (lines 20-24).

Algorithm 4-1. Online Testing Procedure

```
1    for i from 0 to m - 1 do    // for every row
2        s_{n-1} … s_0 ← READ ROW i    // store in s the data stored in the row
3        PROGRAM LRS 1...1 ROW i    // a cell with T_SAON fault in other rows is also programmed
4        a_{i,n-1} … a_{i,0} ← READ ROW i
5        for j from 0 to n – 1 do    // for every column
6            if a_{i,j} = '0' then    // check for any cell sensed as HRS instead of LRS
7                cell (i,j) is faulty    // M_SAOFF or T_SAOPEN fault detected
8            end if
9        end for
10       PROGRAM HRS 1...1 ROW i    // other cells are not programmed
```

```
11        a_{i,n-1} … a_{i,0} ← READ ROW i
12        for j from 0 to n – 1 do    // for every column
13            if a_j = '1' then    // check for any cell sensed as LRS instead of HRS
14                cell (i,j) is faulty    // M_SAON or interference due a T_SAON fault is detected
15            end if
16        end for
17        PROGRAM LRS s_{n-1} ... s_0 ROW i    // retrieve original data of row i
18        PROGRAM HRS s̄_{n-1} ... s̄_0 ROW I    // retrieve original data of row i
19    end for
20    for j from 0 to n – 1 do    // for every column
21        if (n-1 cells in column j are faulty except cell (k,j)) then
22            cell (k,j) is faulty    // T_SAON fault detected
23            other cells in column j are not faulty    // sensed as faulty due to cell (k,j) interference
24        end if
25    end for
```

Although not mentioned in the algorithm to simplify the whole explanation, further registers are necessary to keep track of which cells are detected as faulty or are candidates. Size of all memory elements (especially array *a*) will increase as the memory array size increases. In this context, the algorithm could be modified to overcome this issue, e.g. by only storing information about the locations of the cells detected or interpreted as faulty (lines 7, 14, 22 and 23), thus avoiding the storage of all the outputs from read operations.

4.2.5 Simulation results and discussion

We designed the different parts of the 1T1R crossbar memory system shown in **Fig. 4-5a**. Inverters and pass transistors were built using a predictive transistor model for 65 nm node [75]. Moreover, the comparators and the multiplexer were implemented using Verilog-A behavioral description. Parameter values used in simulations were V_{WRITE} = 2 V, V_{READ} = 1 V, R_{REF} = 200 kΩ and R_{SAON} = 1 kΩ, where R_{SAON} is the resistance of the transistor when a M_{SAON} is simulated. Memristors were always initialized in high resistive state (HRS). In order to check the effectiveness of the suggested online testing procedure, rather than simulate a whole system, it sufficed to check specific parts and cases. Two different temporal simulations were performed: one for the verification of a single cell and the peripheral circuitry, and another to illustrate the interference of a T_{SAON} fault in a reading operation.

The first simulation concerned the circuit that reads and programs a single 1T1R cell, shown in **Fig. 4-4**. The inverters were properly designed such as to drive enough current to program successfully the memristor. The overall circuit operation is depicted in **Fig. 4-8**; although not explicitly shown, *sel* = '1' through all the simulation. As mentioned previously, the state of the memristor in the 1T1R cell is initialized at HRS at the beginning of the simulation. Therefore, a first READ operation (*read* = '1', *en* = '0', *d* = '1' and $\bar{d}$ = 'X') confirms that the cell is at HRS. Next, a PROGRAM LRS operation is performed (*read* = '0', *en* = '1', *d* = '0' and $\bar{d}$ = '1'), achieving a voltage similar to V_{WRITE} across the memristor, and the READ

operation again shows that the memristor was programmed successfully at LRS. Finally, a HRS is programmed (*read* = '0', *en* = '1', *d* = '1' and $\bar{d}$ = '0') and the read output is HRS again, as expected. The program pulse width was 120 ns, long enough to permit the memristor to switch completely its state. On the other hand, the read pulse width was 10 ns, short enough to make sure the device state is not altered during the read phase. Given the aforementioned properties of the applied read/program voltage pulses, the HRS value was 1.5 MΩ and the LRS was 21 kΩ (see memristor model details in Chapter 2).

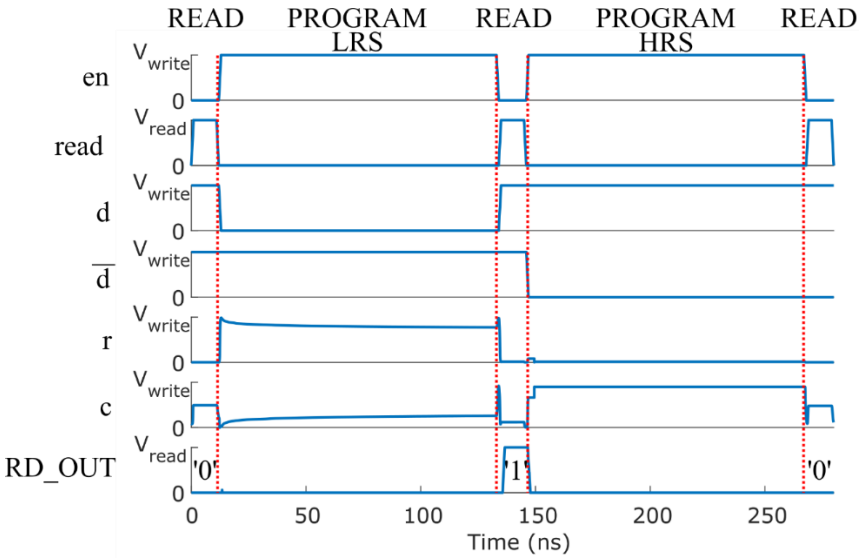


Fig. 4-8. Temporal simulation of the verification of circuit in **Fig. 4-4**, in which operations READ, PROGRAM LRS, READ, PROGRAM HRS and READ are sequentially performed. RD_OUT logic level is also displayed in READ operations.

Given that all the single cell program/read operations in simulation worked as expected, we proceeded to simulated specific cases related to the faults to be detected. More specifically, the most interesting case of fault testing concerns the detection of T_{SAON} faults at the select transistors, since in such case the corresponding memristor interfere with another cell found in the active crossbar row. Instead of simulating the entire system including just one faulty cell, in the simulated circuit we included only the target cell to be read and the interfering faulty memory cell. Thus, based on the scenario shown in **Fig. 4-6**, only the cells (1,2) and (2,2) were simulated using the equivalent circuit for the second column of the crossbar shown in **Fig. 4-9a**, where the T_{SAON} fault was modelled using the resistor R_{SAON}.

The time-evolution of all the signals related to this simulation are detailed in **Fig. 4-9b**. Recall that both memristors were initialized at HRS. The simulation starts by reading the content of the cell (1,2). Therefore, since cell (2,2) is in HRS, the interference is not significant/harmful and the sensed output *RD_OUT* = '0' confirms that memristor in this cell is in HRS. Then, the cell (1,2) is programmed to LRS; additionally, cell (2,2) is also switched to LRS because it suffers from a T_{SAON} fault. The next read operation is also successful sensing LRS (*RD_OUT* = '1'). In fact, the interfering device is in fact contributing to the correct sensing of a LRS, decreasing the equivalent resistance sensed by the read circuit. Finally, the cell (1,2) is programmed back to HRS. Unlike the program LRS, cell (2,2) remains in LRS and does not switch (due to the modification of the PROGRAM HRS ROW operation as explained in the previous subsection).

When checking for the new programmed state of cell (1,2), in the read operation cell (2,2) interferes and as a result the cell (1,2) is wrongly marked as faulty due to the unexpected sensing output *RD_OUT* = '1'.

As the algorithm repeats this procedure with the rest of the rows, at the end all cells found in column 2 will be wrongly marked as faulty (as proved in the simulation with the cell (1,2)) except cell (2,2). Consequently, if this is the case (see lines 20-25 in pseudo-code), then the last instruction should swap the marks for faulty and non-faulty cells in column 2, to eventually identify the cell with the T_{SAON} fault.

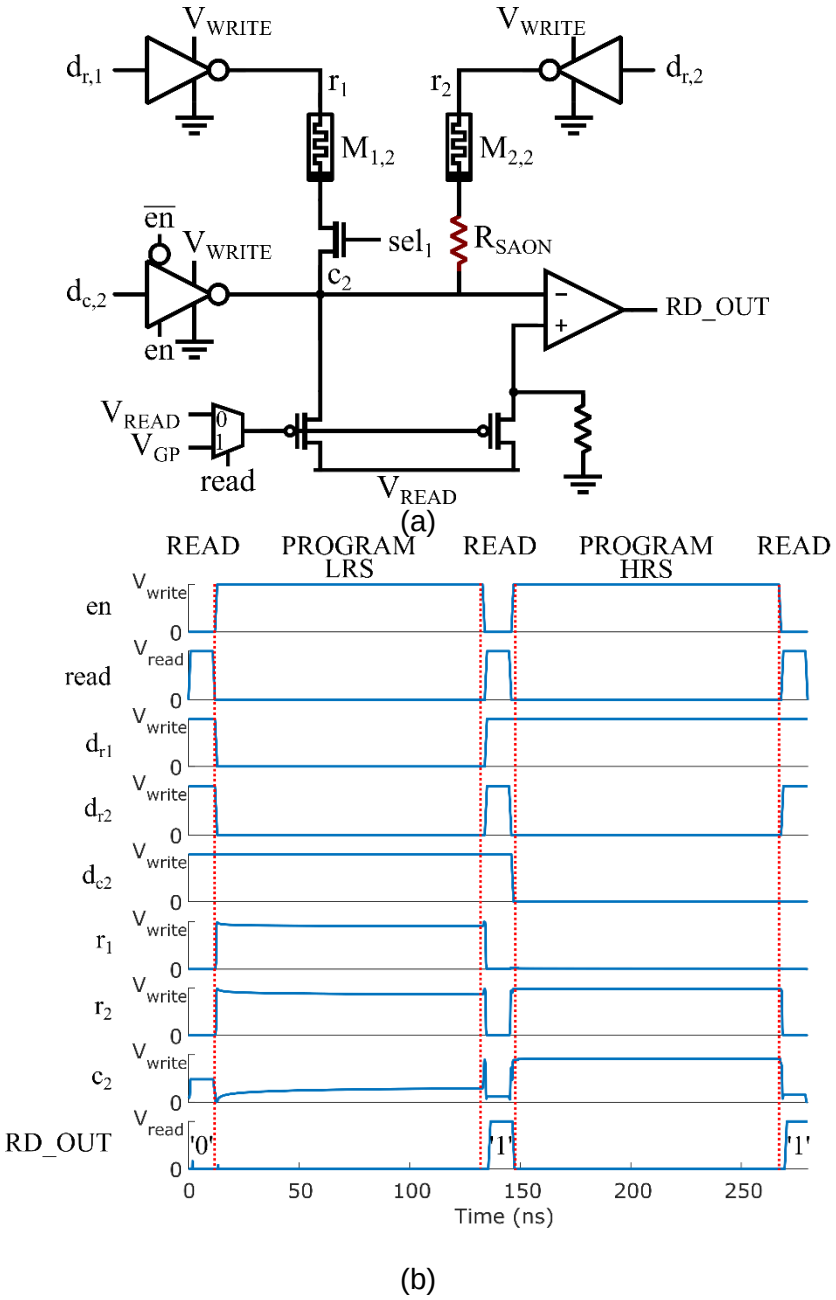


Fig. 4-9. Verification of the interference caused by a cell suffering a T_{SAON} fault, based on the example of **Fig. 4-6**. (a) Equivalent circuit of the column 2 in **Fig. 4-6**, where $M_{1,2}$ and $M_{2,2}$ are the memristors of cells (1,2) and (2,2), respectively. (b) Temporal simulation when applying the sequence of operations READ, PROGRAM LRS, READ, PROGRAM HRS and READ. RD_OUT logic level is also displayed.

4.3 Concluding Remarks

Memories are one of the main applications in which memristors are expected to be a decisive technology. Memristors provide multiple advantages to compete with conventional technologies: crossbar

structures that provide massive device density, 3D structures that and multi-bit memories. All these features have been covered in the first part of this Chapter.

In the context of the reliability issues that face memristors, an on-line test strategy has been presented here able to detect a single fault in a 1T1R crossbar-based memory systems. The testing procedure can be performed when required, halting the normal operation. The algorithm proposed is able to detect four different types of faults presented in the 1T1R, while preserving the data stored in the memory. In particular, the simulation results confirm the interference from a cell with a T_{SAON} fault, which is the unique fault that induces two different cells to interact. Once known and verified the behavior of this fault, the whole algorithm of the test is confirmed to be effective.

In the next Chapter, memristor-based digital computing applications are covered. One of their most interesting features is the use of a crossbar memory where the data is not only stored but also computed. Hence, fault detection methods like the one presented before may apply to memristor-based digital computing systems. Instead of considering faults in the system, the focus is on the sensitivity of C2C and D2D variability in memristors when evaluating different logic gates.

Chapter 5 Addressing Variability Issues in Memristor-based Digital Computing Circuits

Besides digital memories, another application field where memristors were quickly identified as promising candidate devices, is in digital logic circuits. In fact, the memristor can be viewed as a two-terminal programmable resistive switch, thus being an element that can be used to build logic gates and digital computing systems. However, memristor offers other interesting features too. For instance, the nonvolatile data storage capability opens new possibilities to nonconventional forms of computing. In spite of such new possibilities, there are certain challenges that memristor-based logic might face, some of them being in common with other memory system technologies; e.g. limited write endurance and variability issues. In this Chapter, some widely known memristor-based logic styles of the recent literature and their relevant issues, are reviewed. Next, a set of selected logic styles are further explored at circuit level and analyzed, with respect to the impact of variability of memristors on their performance. Finally, a novel logic style is thoroughly described, that addresses design limitations owing to variability and thus it represents a viable solution to future memristor-based in-memory computing systems. Such works have been previously presented in [77], [78] and [79].

5.1 Review of Memristor-based Digital Computing Circuits

Since the claim of the fabrication of the first "modern" physical memristor in early 2008 by Hewlett Packard Labs., digital logic circuits based on memristor is a topic that has been extensively explored by the research community. In general, the contributions relevant to constructing logic circuits using memristors as switching elements, have been mostly focused on developing different logic design styles with results based primarily on SPICE simulations using behavioral (phenomenological) device models other than models based on device physics.

By logic style we refer to a design methodology that can produce all kinds of logic functions, and primarily universal ones such as NAND, NOR, or complex ones such as XOR. In the realm of MOSFETs, some examples of logic styles are *static CMOS logic*, *ratioed logic*, or *pass transistor logic*. Memristor-based logic is not much different from such conventional logic styles, and there are various proposed logic styles; each one aims to provide different functionalities, as to simplify the design of the circuitry, to reduce the latency to compute a function or to lessen the stress in the devices by using low voltages in the computation. that aimed to improve performance and synthesis results or address specific problems, to name a few. Memristor-based logic styles can be classified with respect to different features. Two classification criteria were used in **Table 5-1**, concerning *stateful* or *non-stateful* logic, and compatibility with the crossbar geometry.

Table 5-1. Classification of known memristor-based logic design styles.

	Crossbar	**Non-crossbar**
Stateful	Logic Implication [80] Converse Non-Implication [81] Memristor-Aided Logic (NOR) [82] Scouting Logic [83]	Memristors As Drivers [84] Memristor-Aided Logic (other gates) [82] CMOS-like Logic [85]
Non-stateful	-	Memristor Ratioed Logic [86]

We call stateful all the logic style families where memristors fulfill two functions: on one hand, they hold the input data used in the computation encoded in their resistive state; one the other hand, they are directly involved in the circuit that performs the logic computation taking into account their internal resistive state. The common characteristics of this form of logic are as follows:

- Memristors are previously programmed codifying input data in the form of resistance.
- The computation consists in the voltage/current stimulation of a circuit that includes memristors storing the input data; the output may be obtained in two ways:
    - Sensing a voltage/current produced by the circuit; such magnitude contains information about the output. This output can be subsequently stored by programming a memristor.
    - Conditional programming of a memristor due to the voltage/current stimulation used during the computation, which eventually holds the output data.
- Output data can be retrieved, i.e. its content can be read, by other subsystems.
- It is essentially sequential logic, where each step constitutes either a memory or a logic operation.

In Von Neumann computer architectures (see **Fig. 5-1a**) computing and memory tasks take place in separate modules which communicate through a common bus. Moreover, because CPU operates at a rate which is higher than the throughput achieved between memory and CPU modules (what is known today as the memory wall problem), the communication bus became a real bottleneck in such systems. On the contrary, stateful logic enables non-conventional computing styles and architectures where computing and memory operations are held in the same physical place. This new paradigm is called *Computing in-memory* (CIM) and its scheme is generally described in **Fig. 5-1b**. In CIM, memory and computing are not performed in separate modules connected by a bus, but instead both functions are performed in the same physical space and are practically controlled by a memory controller/CPU. However, usually not all the computing performance is exclusively located inside the CIM block, but there are functions that are held using dedicated peripheral circuit blocks. These architectures are called *near Computing In-Memory* (see **Fig. 5-1c**) and usually concern the fact that data goes in and out the nonvolatile memory module, while still avoiding the bottleneck of the Von Neumann architecture.

As it has been widely discussed previously in Chapter 2, crossbar structures can take full advantage of the favorable memristor features. Therefore, reasonably several logic styles proposed in the literature aimed at supporting crossbar compatibility, meaning that they can be performed inside a crossbar array. Hence, rather than using a special/dedicated circuit topology for exclusive memory/computing tasks, in fact the crossbar array serves multiple purposes, ranging from memory to digital computing while taking advantage of different compatible logic styles. Therefore, crossbar compatibility is a valuable characteristic of a logic style, given that the crossbar array is the most promising circuit architecture for future resistive memories. However, it is also known that crossbar arrays require the operations to be highly controlled due to the sneak-path currents or parasitics that could become dominant in large crossbar arrays.

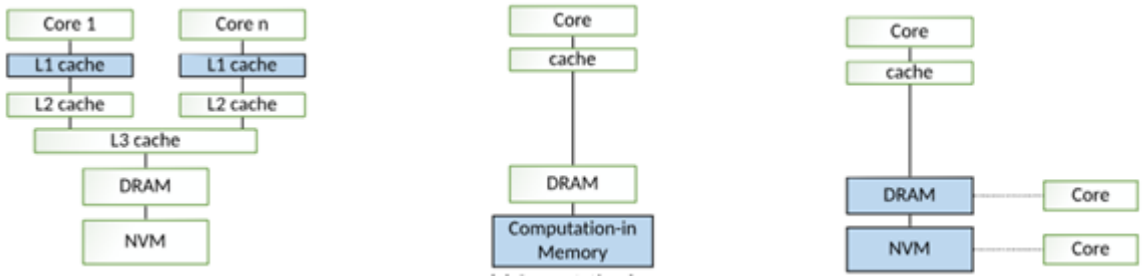


Fig. 5-1. Computing architectures and strategies related to the distance between computing and memory subsystems. (a) CPU-centric approach, (b) Computing-In-Memory and (c) Near Computing-In-Memory.

5.1.1 Logic Implication (IMPLY Logic)

One of the first logic design styles proposed in the field of digital computing applications of memristors, was the logic implication or material implication (or simply IMPLY logic) [80]. This style is called after the logic operator that has the same name: IMPLY, which is a binary operator written as $y = p \rightarrow q$ whose truth table is shown in **Table 5-2**. The output variable y is '1' if p = '0', otherwise $y = q$. Therefore, it can also be described as $y = \neg p + q$. Unlike other typical logic operators, such as AND, OR, NOR, NAND, XOR…, ETC., logic implication is not commutative, i.e. $p \rightarrow q$ and $q \rightarrow p$ has different outcomes.

Table 5-2. Logic Implication truth table

p	q	$y = p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Logic implication can be implemented as an stateful logic operation using a circuit shown in **Fig. 5-2a**. The memristors m_p and m_q represent the inputs p and q, respectively, which are assumed to be previously stored in these devices, where '1' is encoded as LRS and '0' is encoded as HRS. Memristors

bottom terminals are tied to a R_G resistor and top terminals are connected to the driver circuit, where R_G is a resistor whose value is selected between HRS and LRS of the memristors.

The circuit operates as follows: both m_p and m_q drivers apply voltage pulses of positive amplitude V_{COND} and V_{IMP}, respectively. V_{COND} is lower than V_{SET}, preventing m_p to be switched, whereas V_{IMP} is higher than V_{SET}, thus providing m_q the possibility to be programmed to LRS. However, m_q is only switched to LRS if m_p is in HRS (p = '0'), when most of V_{COND} falls across m_p ($R_{HRS} > R_G$) and also most of V_{IMP} falls across m_q. In any other case m_q should not switch, either because m_q is already in LRS or m_p is LRS, forcing a high voltage in G node that makes the voltage drop in m_q insufficient to switch. As a result, the logic implication operation is performed on the two memristors and its result is stored in m_q, i.e. $q' = p \rightarrow q$, where q' refers to q content after the evaluation of the operation. With a proper design of the applied voltage pulses, the circuit operates as it is described in **Fig. 5-2b**, i.e. m_p keeps its state but m_q switches from HRS to LRS only if both m_p and m_q are initially both at HRS.

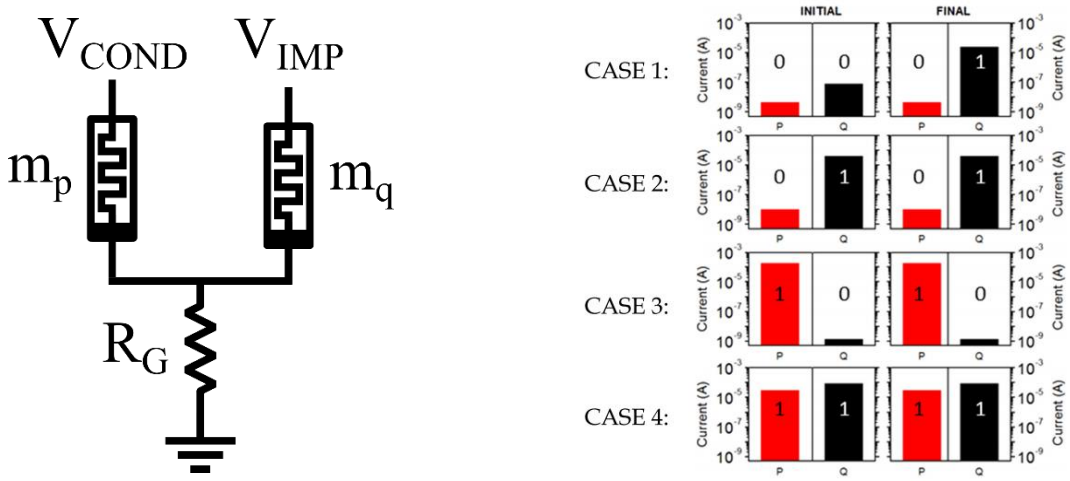


Fig. 5-2. IMPLY operation. (a) Schematic of the circuit that performs the IMPLY logic operation. (b) Experimental verification of an IMPLY logic operation using Ni/HfO2/Si unipolar devices. Sensing currents before (initial) and after (final) the operation according to the four cases in the logic implication truth table (see Table 5-2). Extracted from [87].

The selection of the input voltage pulse properties is critical for the correct function of the IMPLY operations. In this direction, we explore the design constraints for V_{COND} and V_{IMP} analyzing the circuit in **Fig. 5-2a** according to the expected m_p and m_q states before and after the IMPLY operation, while assuming that $R_{HRS} >> R_G >> R_{LRS}$. Prior to discussing further each case in the corresponding truth table, the voltage drop across every memristor is calculated. Applying Kirchoff Current Law in node G, voltage V_G is obtained as follows:

$$V_G = \frac{(1/R_p)V_{COND} + (1/R_q)V_{IMP}}{(1/R_p) + (1/R_q) + (1/R_G)} \tag{14}$$

Hence, voltages across m_p and m_q, V_p and V_q, are immediate and are given by the following equations:

$$V_p = V_{COND} - V_G = \frac{(1/R_q + 1/R_G)V_{COND} \; - \; (1/R_q)V_{IMP}}{(1/R_p) \; + \; (1/R_q) + (1/R_G)} \tag{15}$$

$$V_q = V_{SET} - V_G = \frac{-(1/R_p)V_{COND} \; + \; (1/R_p + 1/R_G)V_{IMP}}{(1/R_p) \; + \; (1/R_q) + (1/R_G)} \tag{16}$$

Next, all cases in the truth table of **Table 5-2** are evaluated to meet the design constraints for V_{COND} and V_{IMP} amplitudes. For each case, one condition related with the evolution of m_p and m_q in the IMPLY operation is given. Taking into account that m_p keeps their state throughout the IMPLY operation, one condition must ensure that V_p at the beginning of the operation does not surpass V_{SET} or V_{RESET} (depending on m_p state) and m_p does not switch. Meanwhile, m_q state evolution in the circuit reflects the differences between q to q' in the truth table; therefore, a second condition must force V_q the evolution from q to q' at the beginning of the operation. For each condition, (15) and (16) are used and the assumption $R_{HRS} >> R_G >> R_{LRS}$ is used to simplify the resulting constraints. Depending on the input combination, we have the following cases:

- Case 1: $p = q = 0$, resulting in $q' = 1$. m_p keeps its state and m_q experiences a SET process:

$$V_p < V_{SET} \Rightarrow \frac{(1/R_{HRS} + 1/R_G)V_{COND} \; - \; (1/R_{HRS})V_{IMP}}{(1/R_{HRS}) \; + \; (1/R_{HRS}) + (1/R_G)} \approx V_{COND} < V_{SET} \Rightarrow$$
$$\Rightarrow \boldsymbol{V_{COND} < V_{SET}} \tag{17}$$

$$V_q > V_{SET} \Rightarrow \frac{-(1/R_{HRS})V_{COND} \; + \; (1/R_{HRS} + 1/R_G)V_{IMP}}{(1/R_{HRS}) \; + \; (1/R_{HRS}) + (1/R_G)} \approx V_{IMP} > V_{SET} \Rightarrow$$
$$\Rightarrow \boldsymbol{V_{IMP} > V_{SET}} \tag{18}$$

- Case 2: $p = 0$ and $q = 1$, resulting in $q' = 1$. Both m_p and m_q keep their state:

$$V_p < V_{SET} \Rightarrow \frac{(1/R_{LRS} + 1/R_G)V_{COND} \; - \; (1/R_{LRS})V_{IMP}}{(1/R_{HRS}) \; + \; (1/R_{LRS}) + (1/R_G)} \approx V_{COND} - V_{IMP} \; < V_{SET} \Rightarrow$$
$$\Rightarrow V_{COND} - V_{IMP} \; < V_{SET} \tag{19}$$

$$V_q > V_{RESET} \Rightarrow \frac{-(1/R_{HRS})V_{COND} \; + \; (1/R_{HRS} + 1/R_G)V_{IMP}}{(1/R_{HRS}) \; + \; (1/R_{LRS}) + (1/R_G)} \approx 0 > V_{RESET} \Rightarrow$$
$$\Rightarrow 0 > V_{RESET} \tag{20}$$

- $p = 1$ and $q = 0$, resulting in $q' = 0$. Both m_p and m_q keep their state:

$$V_p > V_{RESET} \Rightarrow \frac{(1/R_{HRS} + 1/R_G)V_{COND} \; - \; (1/R_{HRS})V_{IMP}}{(1/R_{LRS}) \; + \; (1/R_{HRS}) + (1/R_G)} \approx 0 > V_{RESET} \Rightarrow$$
$$\Rightarrow 0 > V_{RESET} \tag{21}$$

$$V_q < V_{SET} \Rightarrow \frac{-(1/R_{LRS})V_{COND} \; + \; (1/R_{LRS} + 1/R_G)V_{IMP}}{(1/R_{LRS}) \; + \; (1/R_{HRS}) + (1/R_G)} \approx V_{IMP} - V_{COND} < V_{SET}$$

$$\Rightarrow \boldsymbol{V_{IMP} - V_{COND} < V_{SET}} \quad (22)$$

- $p = q = 1$, resulting in $q' = 1$. Both m_p and m_q keep their state:

$$V_p > V_{RESET} \Rightarrow \frac{(1/R_{LRS} + 1/R_G)V_{COND} \; - \; (1/R_{LRS})V_{IMP}}{(1/R_{LRS}) \; + \; (1/R_{LRS}) + (1/R_G)} \approx \frac{V_{COND} - V_{IMP}}{2} > V_{RESET} \Rightarrow$$

$$\Rightarrow \boldsymbol{V_{COND} - V_{IMP} > 2V_{RESET}} \quad (23)$$

$$V_q > V_{RESET} \Rightarrow \frac{-(1/R_{LRS})V_{COND} \; + \; (1/R_{LRS} + 1/R_G)V_{IMP}}{(1/R_{LRS}) \; + \; (1/R_{LRS}) + (1/R_G)} \approx \frac{V_{IMP} - V_{COND}}{2} > V_{RESET}$$

$$\Rightarrow V_{IMP} - V_{COND} > 2V_{RESET} \quad (24)$$

Among the above mentioned conditions found through our analysis, some of them result redundant or do not give any relevant information (e.g. that resulting in $V_{SET} > 0$, which is known by default). The set of conditions (17), (18), (22) and (23) (marked in bold) are enough to define V_{COND} and V_{IMP}. Another important aspect in the design of this logic style is that a high R_{OFF}/R_{ON} ratio is very desirable for the memristors so that the IMPLY operation becomes feasible.

The key characteristic of this logic style is that the IMPLY operator plus the FALSE operator (i.e. the RESET process of a memristor), form a universal/complete set of functions. Therefore, any complex logic function can be built using sequential IMPLY and FALSE operations. During the computation of a logic function, the number of memristors needed for it can be quite different. Moreover, there are three possible roles that memristors can adopt: being *input*, *output* and *auxiliary* devices. More specifically, input memristors store the input data, whereas output memristors store the result of computation, i.e. the output of the logic operation. Auxiliary memristors mainly store intermediate operations required during the execution of complex logic functions that consist of several sequential (cascaded) operations. A NOR*n* gate (i.e. a NOR gate of *n* inputs) needs *N*+1 memristors to be implemented, corresponding to *N* inputs and 1 output; it requires t. On the other hand, a XOR2 function may be built using 4 or 5 memristors, depending if we are interested in preserving the input data after completing the function computation or not. By preserving the data, here we mean that the memristors holding the input data will keep their resistive states.

Furthermore, high fan-in IMPLY operations, i.e. using more than two inputs, have been also studied [88]. In such case, there are multiple inputs stimulated with a V_{COND} pulse, whereas the memristor to hold the output is stimulated with a V_{IMP} pulse, as shown in **Fig. 5-3**. If every input memristor is in HRS, the logic output is '1'. However, if at least one input memristor is in LRS, the output memristor should not switch to LRS (similar to what happens in a conventional IMPLY, for m_p being a single input and m_q treated as an output memristor). The operation performed now is $y = \text{OR}(x_1, x_2, .., x_n) \rightarrow y$, or $y \equiv \text{NOT}(\text{OR}(x_1, x_2, \cdots, x_n)) + y$. The main difference is that the selected parameters V_{COND}, V_{SET} and R_G may depend on the number of inputs used in the operation. The high fan-in IMPLY allows to involve several input data in one operation and compute a the NOR logic function, which is a useful complement to the logic implication. For instance, a NOR*n* is performed in at least 2 + 2*n* steps using IMPLY + FALSE operations, while using high fan-in

IMPLY only needs 2 steps. Therefore, these high fan-in IMPLY operations can be combined with binary IMPLY operation of **Fig. 5-2** and the complementary representation of input data to drastically improve the number of steps required in a computational task [88].

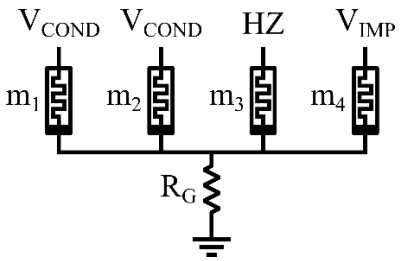
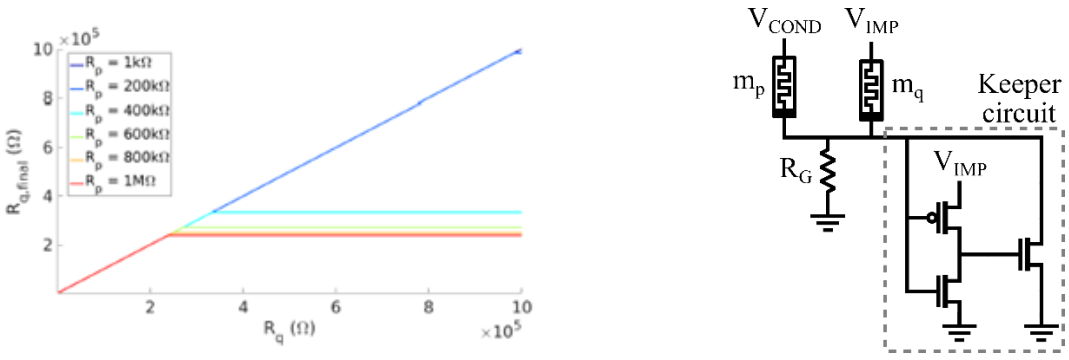


Fig. 5-3. Multi-input IMPLY logic gate. In the example, memristors m_1 to m_4 store inputs x_1 to x_4. By applying V_{COND} at m_1 and m_2 top terminals, V_{IMP} at m_4 top terminal and avoiding significant current to pass through m_3 (HZ stands for high impedance output provided by a driver), the logic function performed is $x_4' = OR(x_1, x_2) \rightarrow x_4$.

The IMPLY logic scheme suffers from an issue often called *state-drift* [81] [89] [90]. More specifically, when the conditional SET occurs, the memristor m_q starts to switch from HRS to LRS. As the resistance decreases, the voltage drop across the m_q also decreases. Eventually, the voltage drop falls below the voltage threshold of the device and, given that it is $R_G >> R_{LRS}$, the memristor does not complete the switching process towards LRS, thus being kept in an unwanted intermediate resistive state. For example, **Fig. 5-4a** depicts the final state of m_q after an IMPLY operation was performed for different initial m_p and m_q states, in a simulation using the VTEAM model [54] and resistance parameters R_{ON} =1 kΩ and R_{OFF} = 1 MΩ. The final R_q is stuck at values between 200 kΩ to 400 kΩ, instead of ending in lower levels, corresponding to the LRS region. Some solutions to this problem have already been proposed in the literature so as to avoid this limitation. For instance, one alternative is the inclusion of a voltage keeper circuit, depicted in **Fig. 5-4b**, in parallel with R_G [89]. When the circuit senses that the voltage at node G is too low, it enables the transistor that connects node G directly to ground and thus helps to maintain the voltage across m_q until it successfully finishes the SET process. The downside is certainly the increase of circuit area. Another solution is to use a complementary logic operation with a RESET process instead of the problematic SET process, suggested in [81]. The latter is further explained in the following Section.

5.1.2 Converse Non-Implication (CNIMP)

As it can be figured out, the conditional SET process in the IMPLY operation is limited by the existence of R_G. Therefore, an alternative to bypass this limitation and still not increase the circuit area is to use an operation with a conditional RESET process instead. In this direction, we have the converse nonimplication (CNIMP) [81] being a variant of the IMPLY that makes possible addressing the state drift issue. It is named after the binary operator that it computes, whose truth table is shown in **Fig. 5-5a**. This new operation is written as $y = p \nleftarrow q$ or alternatively $y = (\neg p)q$.

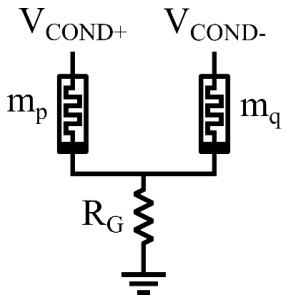


(a) (b)

Fig. 5-4. The state-drift problem observed in IMPLY logic operations. (a) Simulation of the IMPLY logic gate of **Fig. 5-2a** initializing resistance state of m_p (see legend) and m_q (*x*-axis) to different levels. The switching to the new m_q state ($R_{q,final}$) is plotted (*y*-axis), using the VTEAM model [54] and resistance parameters R_{ON} =1 kΩ and R_{OFF} = 1 MΩ. (b) One solution to the state-drift problem is to keep the voltage V_G grounded using a voltage keeper while sensing the G node.

The corresponding circuit, depicted in **Fig. 5-5b**, remains the same as the IMPLY logic style; the main difference concerns the used voltage pulses. More specifically, the amplitudes of the voltage pulses now applied, i.e. V_{COND+} and V_{COND-}, have different characteristics. Concretely, it is $0 < V_{COND+} < V_{SET}$ and $V_{RESET} < V_{COND-} < 0$. With a careful design of the circuit, the memristor m_q experiences a RESET process only if both memristors are in HRS. Likewise in the case of IMPLY, the operation performed is the $q' = p \nleftarrow q$, where the m_q memristor stores the logic output.

p	*q*	*y*	m_p	m_q	m_q'
0	0	0	HRS	HRS	LRS
0	1	1	HRS	LRS	LRS
1	0	1	HRS	HRS	HRS
1	1	1	LRS	LRS	LRS



(a) (b)

Fig. 5-5. Converse Non-implication logic style. (a) CNIMP truth table and (b) the circuit scheme used.

Although the design is similar to the one shown for the IMPLY logic gate, the conditions must abide for the truth table of the CNIMP operator. As before, it is important to figure out the expressions corresponding to the the voltages across the memristors m_p and m_q for this analysis, which we can adapt from (15) and (16) just by substituting V_{COND} and V_{IMP} by V_{COND+} and V_{COND-}, respectively, as follows:

$$V_p = \frac{(1/R_q + 1/R_G)V_{COND+} - (1/R_q)V_{COND-}}{(1/R_p) + (1/R_q) + (1/R_G)} \quad (25)$$

$$V_q = \frac{-(1/R_p)V_{COND+} + (1/R_p + 1/R_G)V_{COND-}}{(1/R_p) + (1/R_q) + (1/R_G)} \quad (26)$$

For each input data combination, V_p and V_q are bound to conditions to determine whether m_p and m_q must switch or not. The relationship $R_{HRS} >> R_G >> R_{LRS}$ is assumed and therefore a high R_{OFF}/R_{ON} ratio is very desirable for the design to rely on the conclusions of this analysis, shown in detail for every input combination, as follows:

- Case 1: $p = q = 0$, resulting in $q' = 0$. Both m_p and m_q keep their state:

$$V_p < V_{SET} \Rightarrow \frac{(1/R_{HRS} + 1/R_G)V_{COND+} - (1/R_{HRS})V_{COND-}}{(1/R_{HRS}) + (1/R_{HRS}) + (1/R_G)} \approx V_{COND+} < V_{SET} \Rightarrow \Rightarrow V_{COND+} < V_{SET} \quad (27)$$

$$V_q < V_{SET} \Rightarrow \frac{-(1/R_{HRS})V_{COND+} + (1/R_{HRS} + 1/R_G)V_{COND-}}{(1/R_{HRS}) + (1/R_{HRS}) + (1/R_G)} \approx V_{COND-} < V_{SET} \Rightarrow \Rightarrow V_{COND-} < V_{SET} \quad (28)$$

- Case 2: $p = 0$ and $q = 1$, resulting in $q' = 1$. Both m_p and m_q keep their state:

$$V_p < V_{SET} \Rightarrow \frac{(1/R_{LRS} + 1/R_G)V_{COND+} - (1/R_{LRS})V_{COND-}}{(1/R_{HRS}) + (1/R_{LRS}) + (1/R_G)} \approx V_{COND+} - V_{COND-} < V_{SET} \Rightarrow \Rightarrow V_{COND+} - V_{COND-} < V_{SET} \quad (29)$$

$$V_q > V_{RESET} \Rightarrow \frac{-(1/R_{HRS})V_{COND+} + (1/R_{HRS} + 1/R_G)V_{COND-}}{(1/R_{HRS}) + (1/R_{LRS}) + (1/R_G)} \approx 0 > V_{RESET} \Rightarrow \Rightarrow 0 > V_{RESET} \quad (30)$$

- Case 3: $p = 1$ and $q = 0$, resulting in $q' = 0$. Both m_p and m_q keep their state:

$$V_p > V_{RESET} \Rightarrow \frac{(1/R_{HRS} + 1/R_G)V_{COND+} - (1/R_{HRS})V_{COND-}}{(1/R_{LRS}) + (1/R_{HRS}) + (1/R_G)} \approx 0 > V_{RESET} \Rightarrow \Rightarrow 0 > V_{RESET} \quad (31)$$

$$V_q < V_{SET} \Rightarrow \frac{-(1/R_{LRS})V_{COND+} + (1/R_{LRS} + 1/R_G)V_{COND-}}{(1/R_{LRS}) + (1/R_{HRS}) + (1/R_G)} \approx V_{COND-} - V_{COND+} < V_{SET} \Rightarrow \Rightarrow V_{COND-} - V_{COND+} < V_{SET} \quad (32)$$

- Case 4: $p = q = 1$, resulting in $q' = 0$. m_p keeps its state and m_q experiences a RESET process:

$$V_q > V_{RESET} \Rightarrow \frac{(1/R_{LRS} + 1/R_G)V_{COND+} - (1/R_{LRS})V_{COND-}}{(1/R_{LRS}) + (1/R_{LRS}) + (1/R_G)} \approx \frac{V_{COND+} - V_{COND-}}{2} > V_{RESET} \Rightarrow \Rightarrow V_{COND+} - V_{COND-} > 2V_{RESET} \quad (33)$$

$$V_q < V_{RESET} \Rightarrow \frac{-(1/R_{LRS})V_{COND+} + (1/R_{LRS} + 1/R_G)V_{COND-}}{(1/R_{LRS}) + (1/R_{LRS}) + (1/R_G)} \approx \frac{V_{COND-} - V_{COND+}}{2} < V_{RESET} \Rightarrow \Rightarrow V_{COND-} - V_{COND+} < 2V_{RESET} \quad (34)$$

Not all the conditions (27)-(34) are useful for the design of the CNIMP operation. Unlike shown previously for IMPLY, here some of the conditions are not giving explicit information and further

mathematical manipulation is required. From (27) and (28), it's clear that both V_{COND+} and V_{COND-} are lower than V_{SET}. But from (34) and (30), V_{COND+} and V_{COND-} relationship is more explicit:

$$V_{COND-} - V_{COND+} < 2V_{RESET} < 0 \Rightarrow V_{COND-} < V_{COND+} \quad (35)$$

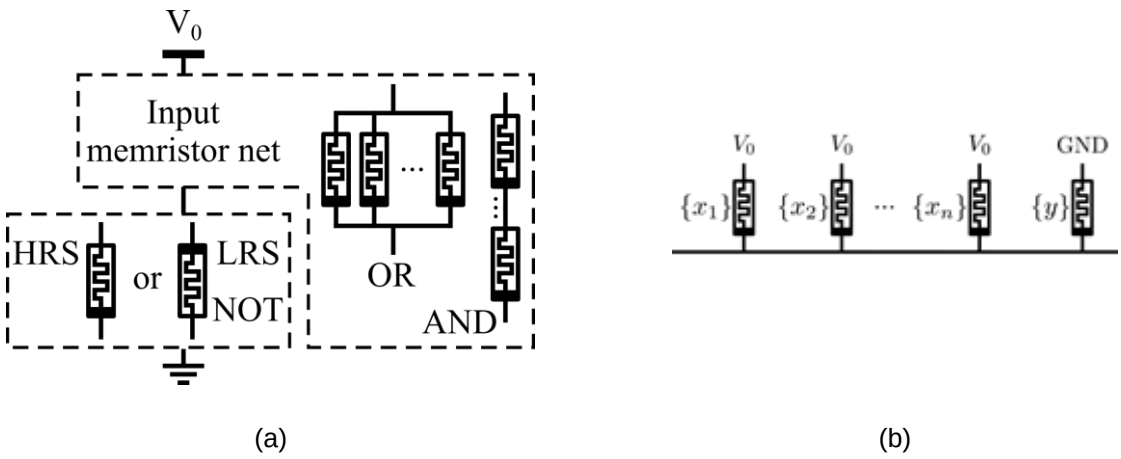
This new relationship makes (28), (32) and (33) redundant. Hence, the conditions (27), (29), (34) and (35) are enough to decide on the required V_{COND+} and V_{COND-}. Apart from determining V_{COND+} and V_{COND-}, conclusions about the memristor desired characteristics can be also derived combining conditions (29) and (34) as follows:

$$2|V_{RESET}| < V_{COND+} - V_{COND-} < V_{SET} \quad (36)$$

This last relationship is really important as it apparently relates both memristor voltage thresholds for SET and RESET. Indeed, it is required that the SET threshold V_{SET} is at least higher than twice the absolute value of the RESET threshold V_{RESET}. Therefore, the target memristors should comply with this asymmetry condition for the CNIMP operation to be correctly executed. The larger this asymmetry is, the larger the available distance between the V_{COND+} and V_{COND-} amplitudes to be selected.

5.1.3 Memristor-Aided Logic (MAGIC)

In this Section we describe the memristor-Aided Logic (MAGIC) [82], which is a stateful logic style that makes possible building different logic gates by just using memristors in series or parallel connection. The general structure of a logic gate comprises a set of input memristors connected to an output memristor which is grounded, as shown in **Fig. 5-6a**. The logic function is generally performed when a positive voltage pulse of amplitude V_0 is applied at the input terminal. The logic function that the logic gate performs depends on the input memristor set and the output memristor polarity. More specifically, to compute a logic AND of two or more inputs, the input memristors are put in series, whereas for a logic OR of two or more inputs, the input memristors are put in parallel. The output memristor polarity/orientation determines if the logic function output is inverted or not. When it is not inverted, the output memristor is initialized at HRS before applying the input voltage pulse, otherwise it is initialized at LRS. For instance, the NAND logic gate has all memristors in series and the output memristor is initially programmed to LRS (representing logic "1") and it is reversely polarized. When a positive voltage pulse V_0 is applied, the input memristors should not switch their state, so that the input data are maintained, but the output memristor should switch depending on the data stored in input memristors (i.e. the logic operation consists in a conditional switch).



(a) (b)

Fig. 5-6. Memristor-Aided Logic style. (a) General structure of a MAGIC gate. The top block may consist of an array of paralle memristor to compute an OR, or memristors in series to compute an AND. The bottom block consists on a single memristor that stores the output and depending on the orientation, it may compute the complementary (NOT) version of the chosen function OR or AND. (b) MAGIC NORn circuit implementation with *n* input memristors in parallel storing input data x_1 … x_n, and the output memristor that stores the output *y*.

Unlike in IMPLY, the MAGIC scheme is able to compute in only one step logic functions such as OR, NOR, AND, or NAND. Despite this improvement in throughput, there are two important considerations. First, only NOR logic gates are implementable in crossbar structures. NOR is a universal logic operator, thus any logic function can be implemented by sequential NOR operations. Hence, the MAGIC logic style can be used to perform CIM in crossbar arrays by means of only NOR operations. Second, the design of the V_0 amplitude and duration depends strongly on V_{SET}, V_{RESET}, R_{OFF} and R_{ON}, and memristors used in this logic style must fulfill certain parameters requirements [82].

For instance, the MAGIC NOR*n* logic gate (see circuit schematic in **Fig. 5-6b**), i.e. a MAGIC implementation of a *n*-input NOR gate, is analyzed next. The voltage across the output memristor V_y and across the input memristors V_x *at the beginning of the logic evaluation* is obtained by applying a voltage divider formula and taking into account that R_y is initially at LRS:

$$V_y = \frac{R_y}{(R_{x1}||\dots||R_{xn}) + R_y} V_0 = \frac{R_{LRS}}{(R_{x1}||\dots||R_{xn}) + R_{LRS}} V_0 \tag{37}$$

$$V_x = \frac{R_{x1}||\dots||R_{xn}}{R_{x1}||\dots||R_{xn} + R_{LRS}} V_0 \tag{38}$$

The logic outcome of NOR*n* operation is ‘1’ (LRS) only if all inputs are ‘0’ (HRS) and ‘0’ if at least one input is ‘1’.

- Case 1: $x_1 = \dots = x_n = 0$ and $y = 1$, all memristors should keep their state:

$$V_x = \frac{R_{HRS}\,/\,n}{R_{HRS}\,/\,n\; + R_{LRS}} V_0 < V_{SET} \Rightarrow V_0 < \left(1 + \frac{nR_{LRS}}{R_{HRS}}\right) V_{SET} \tag{39}$$

$$V_y = \frac{R_{LRS}}{R_{HRS}\,/\,n\; + R_{LRS}} V_0 < |V_{RESET}| \Rightarrow V_0 < \left(1 + \frac{R_{HRS}}{nR_{LRS}}\right) |V_{RESET}| \tag{40}$$

- Case 2: at least one input memristor in LRS and $y = 1$, input memristors should keep their state and m_y experiences a RESET process. The worst input combination is when only one input memristor is at LRS, because it makes V_x to be higher and V_y to be lower:

$$V_x \approx \frac{(R_{LRS}||R_{HRS}/(n-1))}{(R_{LRS}||R_{HRS}/(n-1)) + R_{LRS}} V_0 < V_{SET} \Rightarrow V_0 < \left(1 + \frac{R_{LRS}}{(R_{LRS}||R_{HRS}/(n-1))}\right) V_{SET} \quad (41)$$

$$V_y \approx \frac{R_{LRS}}{(R_{LRS}||R_{HRS}/(n-1)) + R_{LRS}} V_0 > |V_{RESET}| \Rightarrow V_0 > \left(1 + \frac{R_{LRS}||R_{HRS}/(n-1)}{R_{LRS}}\right) |V_{RESET}| \quad (42)$$

Combining conditions (39) to (42) and noticing that condition (39) is stricter than (41):

$$\left(1 + \frac{R_{LRS}||R_{HRS}/(n-1)}{R_{LRS}}\right) |V_{RESET}| < V_0 < min\left(\left(1 + \frac{nR_{LRS}}{R_{HRS}}\right) V_{SET}, \left(1 + \frac{R_{HRS}}{nR_{LRS}}\right) |V_{RESET}|\right) \quad (43)$$

For $n = 2$ and $R_{HRS} >> R_{LRS}$, the boundaries of V_0 is:

$$2|V_{RESET}| < V_0 < V_{SET} \quad (44)$$

Thus, interestingly this analysis shows that the V_0 amplitude has the same boundaries as V_{COND+} - V_{COND-} observed previously in (36). Furthermore, the target memristors implementing MAGIC NOR logic operations must comply with exactly the same requirements, as in the CNIMP case, meaning that both CNIMP and MAGIC could be implemented with the same memristor device technology.

5.1.4 Memristor-Ratioed Logic (MRL)

Memristor-Ratioed Logic (MRL) was proposed in [86] as an approach for building logic gates out of memristors and CMOS devices, in a non-stateful logic style. It may be interpreted as a memristor-based counterpart of the pass-transistor logic or the ratioed logic schemes. **Fig. 5-7** presents the different possible logic gates implemented based on this scheme. In this alternative, input signals are delivered in the form of voltage, assuming V_{DD} for logic '1' and 0 V for logic '0'. The output is obtained at V_{OUT} in voltage form, using the same logic levels as the inputs. The initial state of memristors is irrelevant in this scheme, but their orientation determines the logic function that the circuit performs. If all input voltages are equal, the output is exactly equal to the input voltage. Otherwise, memristors switch their state in a determined way which depends on the combination of input signals and the device polarity, since the memristors act as a gate pass element, i.e. memristors in LRS let to "pass" the voltage in their input terminal, whereas memristors in HRS block the voltage in their input terminal. For instance, in the AND (OR) gate, all memristors directly connected to a '1' input will switch to HRS (LRS) whereas those connected to '0' inputs will switch towards LRS (HRS).

The output voltage level of the logic gate is neither V_H nor V_L, since it depends on the R_{OFF}/R_{ON} ratio. Furthermore, chaining computations may provoke logic failures because every output is gradually

accumulating deviations from their expected logic levels. Hence, CMOS inverters are included in order to regenerate the output signal to the V_H or V_L levels. Also, the inverters provide the complement operation (NAND and NOR), which is necessary in the MRL scheme to enable computing any Boolean function in NAND/NOR logic.

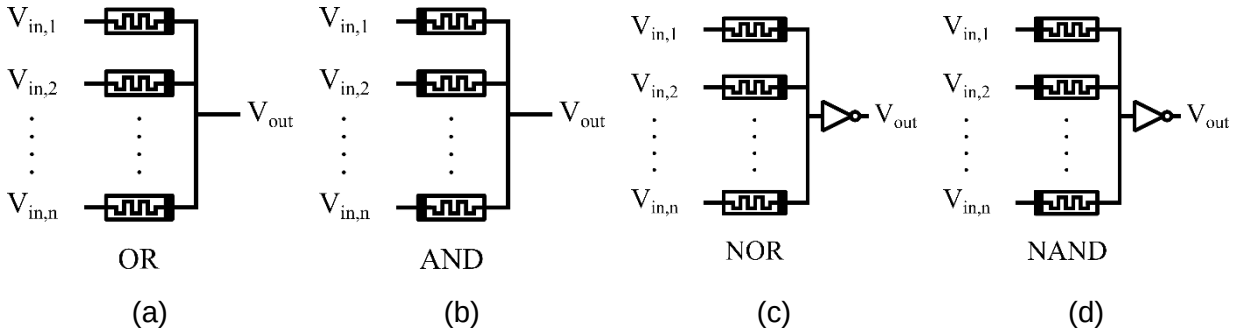


Fig. 5-7. Circuit schematics for four different MRL logic style gates: (a) OR, (b) AND, (c) NOR and (c) NAND. A CMOS inverter stage is put after the MRL logic gate to implement complementary logic functions.

5.1.5 Logic with Memristors As Drivers (MAD)

According to the "Memristors As Drivers (MAD)" logic scheme, proposed in [84], logic gates are implemented in two separate sensing and programming stages. **Fig. 5-8** shows some logic gate examples. More specifically, at the left part of each circuit the sensing stage holds memristors storing input data (where HRS stands for '0' and LRS stands for '1'), whereas at the right part the programming stage is composed of a memristor that stores the output of the computation and one or more voltage-controlled switches. A low amplitude V_{COND} voltage -insufficient to switch any of the memristors in the circuit- is applied to the input memristors in the sensing stage; one or more intermediate nodes are sensed and are the input to one or more voltage-controlled switches found in the programming stage. Consequently, if the voltages surpass some threshold, the switches close and the output memristor sees a voltage drop high enough to switch its state from a previously programmed one. This way, a logic gate evaluation is performed in one step. Logic gates that perform operations such as AND, COPY, NOT and XOR may be built using MAD logic style (see **Fig. 5-8a**, **Fig. 5-8b**, **Fig. 5-8c** and **Fig. 5-8d**, respectively). The OR can be computed using the AND gate but modifying the threshold voltage for the voltage-controlled switch. Moreover, the MAD logic gates can be chained to evaluate complex logic functions, such as the full-adder circuit in [84], which is able to compute the function in two steps.

Everything considered, MAD is a stateful logic with two main advantages: (i) the required voltage pulses are easy to design and also the logic gates, including the XOR and XNOR which were not present directly in the other logic style alternatives, are here computed in just one step (without considering the initialization of the output memristor before the logic gate computations). However, MAD scheme lacks a regularity in the derived circuitry, which in turn impedes the implementation in a crossbar array.

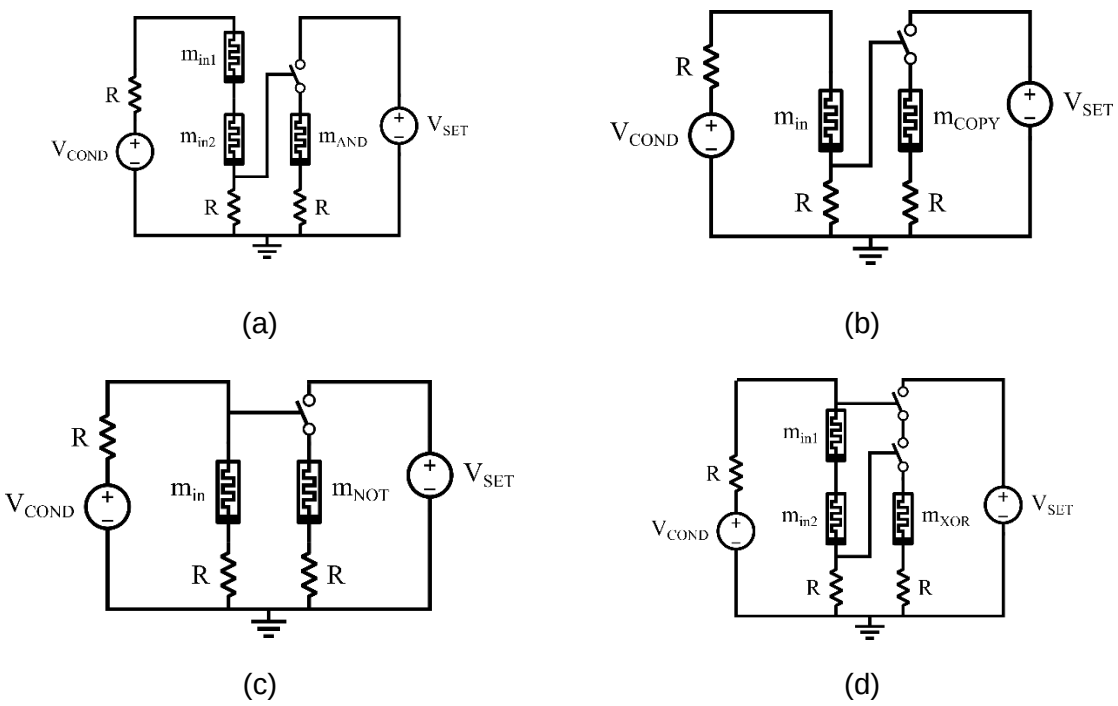


Fig. 5-8. Examples of gates implemented using Memristors As Drivers logic style. (a) AND, (b) COPY, (c) NOT and (d) XOR.

5.1.6 CMOS-like Memristor-based Logic

The CMOS-like logic style proposed in [85] is a logic design style which allows to build memristor-based logic gates inspired on the conventional CMOS logic design. As it is well known, the CMOS logic circuits are composed of two networks of NMOS and PMOS transistors, controlled by signals encoding the input data. NMOS and PMOS networks are complementary, e.g. if NMOS devices are connected in series, the corresponding PMOS devices are put in parallel. The latter is based on the fact that NMOS and PMOS behave in a complementary way when they receive the same input signal. Likewise, the CMOS-like memristive logic gates consist in two complementary networks of memristors, connected between reference and a low sensing voltage V_{DD}, as shown in **Fig. 5-9**. Inputs are not fed directly using external signals, but are instead stored previously in the state of the input memristors. In the bottom net, inputs are stored according to the usual convention using LRS for a '1' and HRS for a '0'. However, the top net stores inputs using the opposite convention, i.e. HRS for a '1' and LRS for a '0'. Hence, when the V_{DD} voltage is applied, the output voltage should get close to V_{DD} or 0 if the result of the logic gate is '1' or '0', respectively, without altering the state of the input memristors. Different examples of CMOS-like logic gates are shown in **Fig. 5-9**, where the topology is inherited from the conventional CMOS gates while substituting transistors for memristors.

The CMOS-like design provides a different way for computing for the stateful logic alternative schemes. Instead of managing sensing and conditional programming in the same circuit, such as with the IMPLY (or CNIMP) and MAGIC, the computation here is just a sensing operation. The output however is delivered as a voltage; thus it must be stored in a memristor if it is supposed to be used as input to another logic gate. The separation of sensing and programming tasks may add delay because there are two steps

instead of one. However, the design of this logic style gets simplified compared to other alternatives, offering more feasibility in the design regarding V_{DD} and memristor characteristics, and reliability in controlling the entire operation, which is the reason why such scheme served as inspiration for others developed later. Finally, it is worth noting that CMOS-like is not compatible with crossbar structures, given the large number of memristors assembled in a complex CMOS-like logic gate, and it also requires considerable additional circuitry to program the inputs.

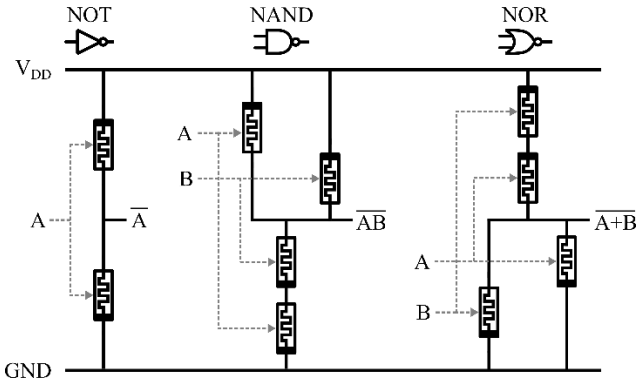


Fig. 5-9. Examples of logic gates built using CMOS-like logic style approach. From left to right, NOT, NAND2 and NOR2 gates. Logic gates schematics are arranged to be related to their CMOS counterparts. The grey lines indicate where the inputs A and B are stored; in the top reverse polarized memristors, the complemented input is stored. The node at the right side of each logic gate indicate the logic function output (delivered in voltage form). The schematics don't include the circuit to program the input memristors (adapted from [85]).

5.1.7 Scouting Logic

The Scouting Logic [83] is another stateful logic style in which the computing task is performed by just sensing the state of the input memristors. The basic principle is shown in **Fig. 5-10**. Memristors are distributed in a topology compatible with a crossbar structure. Each one has a dedicated driver and holds some input data. Drivers supply a sensing voltage V_s and the sensing amplifier (SA) connected to all the memristors senses the input current I_{in}, while comparing it with different reference currents I_{ref}. Depending on the input data, different current levels are obtained, as shown in the diagrams found on the right side of **Fig. 5-10**. Most importantly, by using different I_{ref} values, the SA is able to evaluate different logic functions. Both current and voltage-based SA designs are provided, able to read the state of memristors and to compute AND, OR or XOR logic functions. The work [83] also analyses the variability impact in this logic, proving its reliability features. Likewise, in the CMOS-like case, the output is delivered in voltage form, whereas inputs are in the form of resistance. Therefore, in order to use the output result in future operations, it has to be previously written back to an available memristor.

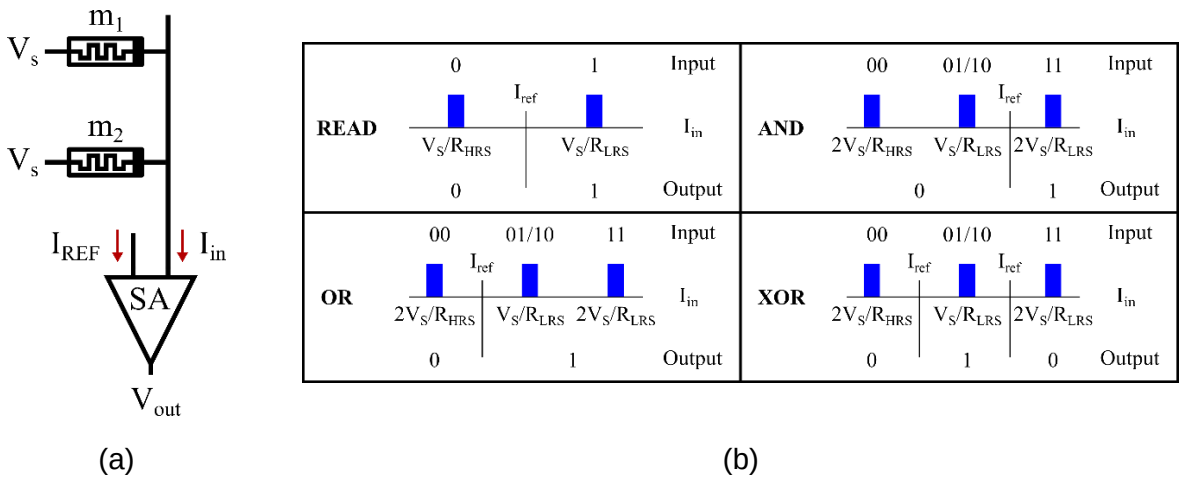


Fig. 5-10. Scouting Logic logic style. (a) Scouting logic scheme in a row or column of a crossbar; V_s is applied to each memristor involved in the operation and a sense amplifier (SA) compares the sensing current I_{in} with a reference current I_{ref}. (b) Sensing and logic functions implemented by using different reference currents I_{ref}.

5.2 Impact of Variability in Memristive Logic inside Crossbar Memory Arrays

Most of the aforementioned memristor-based logic design styles were conceived, and also the design guidelines were devised, while assuming that memristor characteristics are totally deterministic. However, in Chapter 3 it was shown that there are several challenges for memristor reliability that are not negligible. Therefore, this Section focuses on two logic styles, well-received so far by the relevant scientific community, which are crossbar-compatible; namely MAGIC and IMPLY (concretely its CNIMP variant). It further explores the logic design space in scenarios where both C2C and D2D variability exist in the memristor devices. The memristor model adopted for simulations was the Stanford/PKU model, whereas the MOS transistor models were from a realistic 0.35 μm technology. All simulations were performed in the Cadence Virtuoso suite. The major contributions of this Thesis can be found in relevant publications in [77], [78] and [79].

5.2.1 Memristor Model Requirements for CNIMP and MAGIC

The majority of the memristor model parameters were kept at the default values listed in **Table 2-2** in Chapter 2, except those directly affecting the switching thresholds, i.e. the average active energy of oxygen vacancies (E_a), the hopping barrier of O_2- (E_h) and the energy barrier between the electrode and the oxide (E_i). Tuning these parameters is recommended to adjust the overall device behavior according to the target application requirements. For instance, in order to comply with the threshold voltage requirements for MAGIC and CNIMP logic operations, previously mentioned in Subsections 5.1.3 and 5.1.4 (i.e. that is should be $V_{SET} > 2\times|V_{RESET}|$), these parameters were tuned as follows: $E_a = 0.9$ eV, $E_h = 0.9$ eV and $E_i = 0.7$ eV.

Fig. 5-11a demonstrates the *i-v* curves corresponding to 20 cycles under a triangular voltage application to the memristor terminals. The compliance current (cc) is adjusted by tuning the gate voltage of a series NMOS transistor. Since the memristor model does not include voltage thresholds directly as parameters, we define as SET threshold V_{SET} the voltage at which the current reaches to 90% of the cc.

Likewise, we define as RESET threshold V_{RESET} the voltage when the current first experiences a sudden decrease. These statistics give us the following mean values: V_{SET} = 2 V and V_{RESET} = -0.5 V, which allow establishing the appropriate amplitude of the programming/reading voltage pulses. Moreover, the conduction of the model is purely ohmic for the conductive filament body (LRS) but non-linear for the tunneling gap (HRS). This is observed in **Fig. 5-11b** which shows how the effective R_{HRS}/R_{LRS} ratio can change depending on the applied voltage. The device is first SET to R_{ON}, the lowest resistance value, and then a positive voltage is applied without modifying its state. As expected, R_{LRS} shows the ohmic conduction of the CF. Nevertheless, when the device is RESET to R_{OFF}, the highest resistive value, and a large negative voltage is applied, again without disturbing its state, interestingly a highly nonlinear behavior is observed in the effective R_{HRS} owing to the hopping current through the tunneling gap. Therefore, such dependency of the effective R_{HRS} state on the voltage across the device marks a significant difference when compared to other device models commonly used in memristive logic design, such as [48] or [53] This fact could have a significant impact on the efficiency of memristive applications, thus it is an important consideration also for logic circuit design, which is the context of this Chapter.

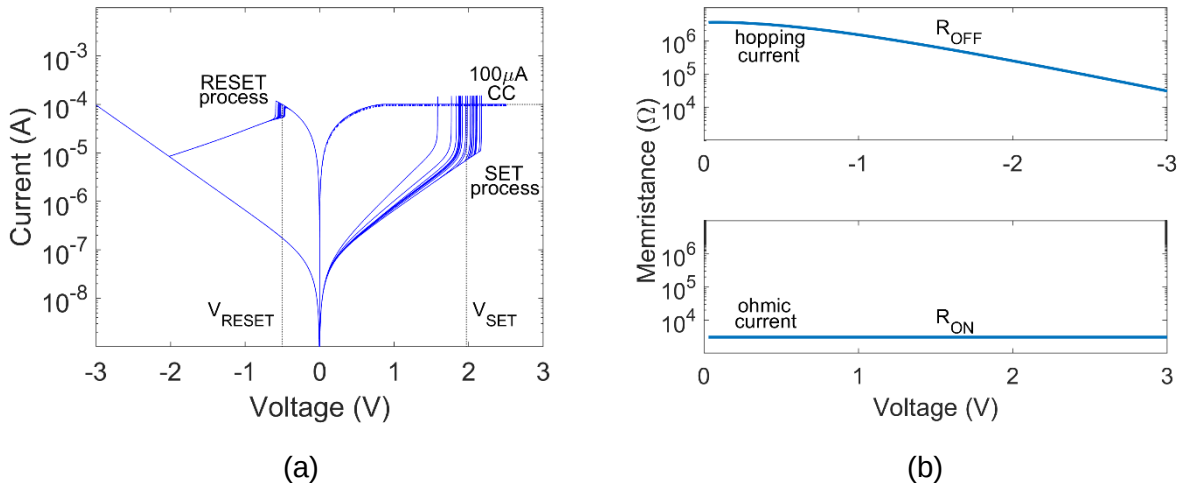


Fig. 5-11. Simulation results to highlight memristor features. (a) *i-v* loops for 20 cycles with the default variability applied, using a 400 µs- period triangular voltage pulse from -3 V to 2.5 V and a 100 µA compliance current for the SET process. (b) Boundary states R_{OFF} and R_{ON} as a function of the applied voltage across the memristor (without variability), illustrating the conduction modes in HRS and LRS states.

5.2.2 Reading and Programming Operations in Presence of Device Variability

Binary encoding of memristance is necessary for our target logic design styles. Given the used parameter values, the memristor model exhibits a memristance range from 5 kΩ (R_{ON}) to 3 MΩ (R_{OFF}). Taking into consideration the results in **Fig. 5-11b**, we defined values above 1 MΩ as HRS and values below 100 kΩ as LRS, whereas all values within the intermediate guard band are treated as undefined states (see memristance map in inset of **Fig. 5-12a**). Reading of the memristor state is performed in voltage mode with a 100 kΩ series resistor by applying a small voltage pulse (V_{read} = 0.5 V, 20 ns- wide) which does not affect the device state. For the purposes of our simulations, we implemented the state decoder shown in **Fig. 5-12a** (adapted from [91].) The voltage read through the memristive voltage divider is compared with two reference values that represent the upper bound of LRS, i.e. $R_{LRS,h}$ = 100 kΩ, and the lower bound of HRS,

i.e. $R_{HRS,l}$ = 1MΩ. So, depending on the result of this comparison, the output of the circuit is either logic '1' for LRS, logic '0' for HRS, or 'X' for any undefined intermediate resistive state.

Regarding the programming operation, variability adds uncertainty to the switching process, and thus to the achieved target state. As introduced in Chapter 3, C2C variability is applied to the memristor parameters in simulation and in our analysis we explore the reliability while considering different amounts of variability; the latter is adjusted using factor *k*. For example, in **Fig. 5-11a** it is illustrated how the current readings as well as the switching thresholds are modified while using k = 1. The circuit in charge of programming and reading a memristors must guarantee that these operations are not affected by the previous history of the device. There is an additional consideration for C2C variability when working with the Stanford/PKU model. If the memristors are initialized to R_{ON} or R_{OFF} by setting the corresponding parameter value in the netlist (for their initial LRS and HRS states, respectively), then C2C variability modelling is not effective. This is because the R_{ON} and R_{OFF} are the state min/max boundaries in which the programmed states is not further modified with variability injection.

As a solution, for our simulation purposes, a two-step initialization process was used instead, as shown in **Fig. 5-12b**. More specifically, when programming the device to the LRS (HRS), this is done by first performing a hard RESET (SET) and then a soft SET (RESET). Hard SET (RESET) completely forms (destroys) the CF to thus eliminate the previous history of the memristor. On the contrary, the HRS or LRS programming taking place right next (called soft programming) initializes the memristor to a value within the LRS or HRS range, as desired, thus including the variability effect in the target memristance during the programming phase. The voltage pulses applied for the HRS initialization concern: V_{ON} amplitude of 3 V, 200 ns width and 500 μA cc for hard SET; $V_{OFF,soft}$ amplitude of – 2 V, and 100 ns width for soft RESET. On the other hand, for the LRS initialization it is: $V_{OFF,hard}$ amplitude of -2.5 V, 200 ns width for hard RESET, and V_{ON} amplitude, 100 ns width and 50 μA cc for soft SET. **Fig. 5-12c** shows simulation results for the memristance distribution of R_{LRS} and R_{HRS} as it was read after such initialization process. It can be observed that variability-aware simulations for HRS/LRS programming show normal distributions for the device state whose std. dev. increases linearly with increasing values of factor *k*.

5.2.3 1T1R Crossbar Array and Peripheral Circuitry to Enable Memory and Logic Operations

The one transistor one resistor/memristor (1T1R) crossbar is the target memory topology in this work and all considered logic design styles are crossbar-compatible. The crossbar memory array as well as the entire peripheral circuitry were designed to make possible the proper programming and reading of the devices (multi-level memory operations), and the realization of the logic operations of interest, according to the selected logic styles described previously.

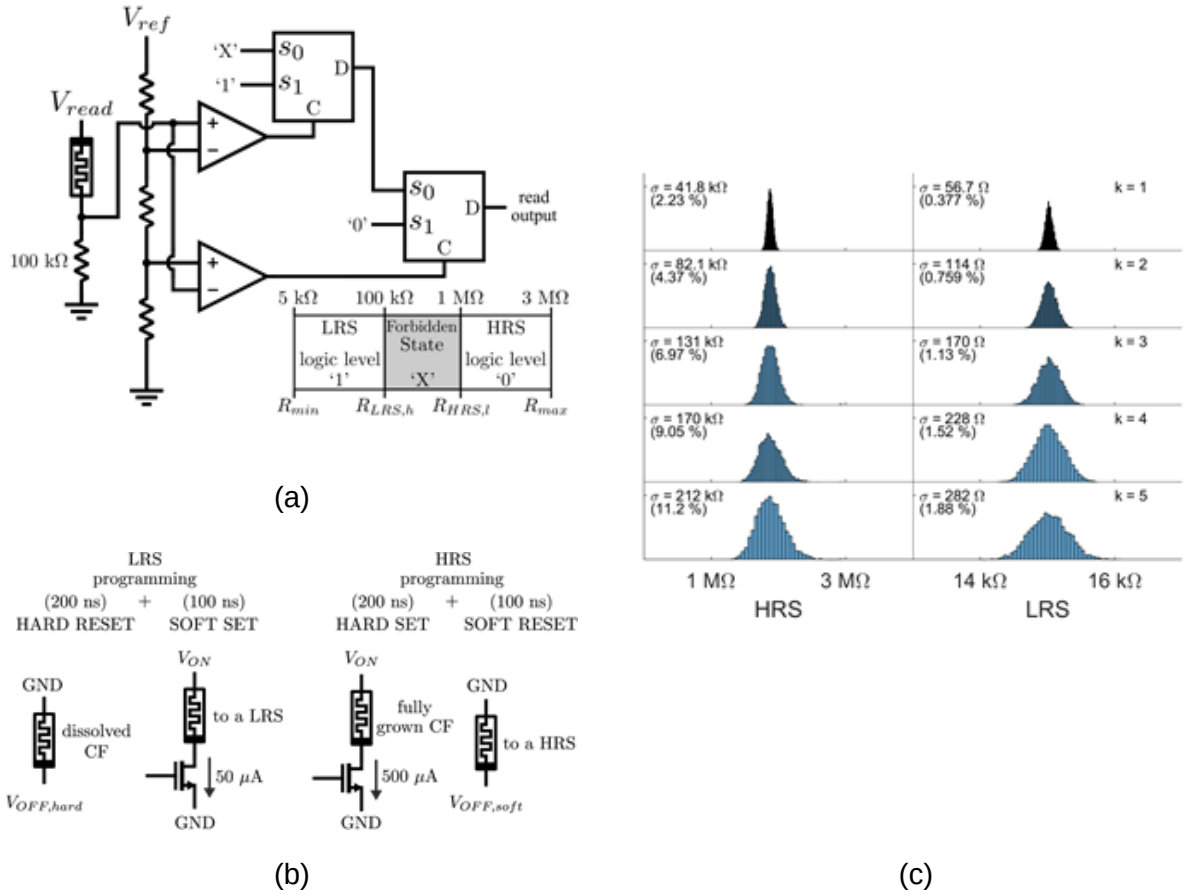


Fig. 5-12. (a) Memristance binary coding and the read-out circuit that senses the state in voltage mode and compares with two different reference voltages to decode the memristance as logic '0', '1' or undefined (badly written) 'X'. (b) Two-step programming details for RESET and SET. (c) 5000 total HRS (left column) and LRS (right column) reading distributions for different *k* values.

The main unit of the simulated system is the crossbar array shown in **Fig. 5-13a**, where each cross-point cell consists of a memristor and a series select transistor (1T1R). 1T1R arrays are beneficial since the selector-enhanced cross-points mitigate the sneak-path currents and also isolate the memristors involved in memory/logic operations from the rest of the devices. However, compared with passive arrays that use highly nonlinear memristors for the same purposes, the only drawback of the 1T1R structure is a penalty on the cross-point area that can be nearly 2.5 times the passive cell area, according to [92]. However, such area penalty may be compensated by the improvements in terms of reliability of memory operations. In this context, it is assumed that the memristors are stacked directly on top of group-accessed select transistors, which are placed at the bottom layer (fabricated as front-end transistors) to limit the influence of the parasitic capacitance and resistance and also to minimize the area penalty (such as in the case of [57] with the nano-pillar gate-all-around (GAA) transistors). The transistor in each row *i* are selected by setting the signal sel_i.

Moreover, there are multiplexer (MUX) and demultiplexer (DEMUX) units to drive the selected columns and rows of the array. The MUX, whose detailed implementation is shown in **Fig. 5-13b**, provides different voltage levels to the rows or columns. It is primarily composed by a decoder whose outputs are connected to the gate terminals of NMOS transistors, so as to selectively connect the output to any of the input voltages according to the selection signals *smr* (for row multiplexers) and *smc* (for column

multiplexers). Alternatively, it may disconnect completely the block from the crossbar (high impedance node - HZ) using the enable signal *emr* (for row multiplexers) and *emc* (for column multiplexers). The input voltage signals are divided in two groups; those concerning memory operations and those used in logic operations with either of the three logic styles explored next. On the other hand, the DEMUX shown in **Fig. 5-13c** provides connection to extra circuitry whenever necessary. Such circuitry concerns the reading circuit described in **Fig. 5-12a**, the transistor that limits the current in the LRS or HRS programming, and the R_G used in the CNIMP logic style. As with the multiplexers, there are selection signals *sdr* and *sdc* (for row and column demultiplexers, respectively) and enable signals *edr* and *edc* (for row and column demultiplexers, respectively)

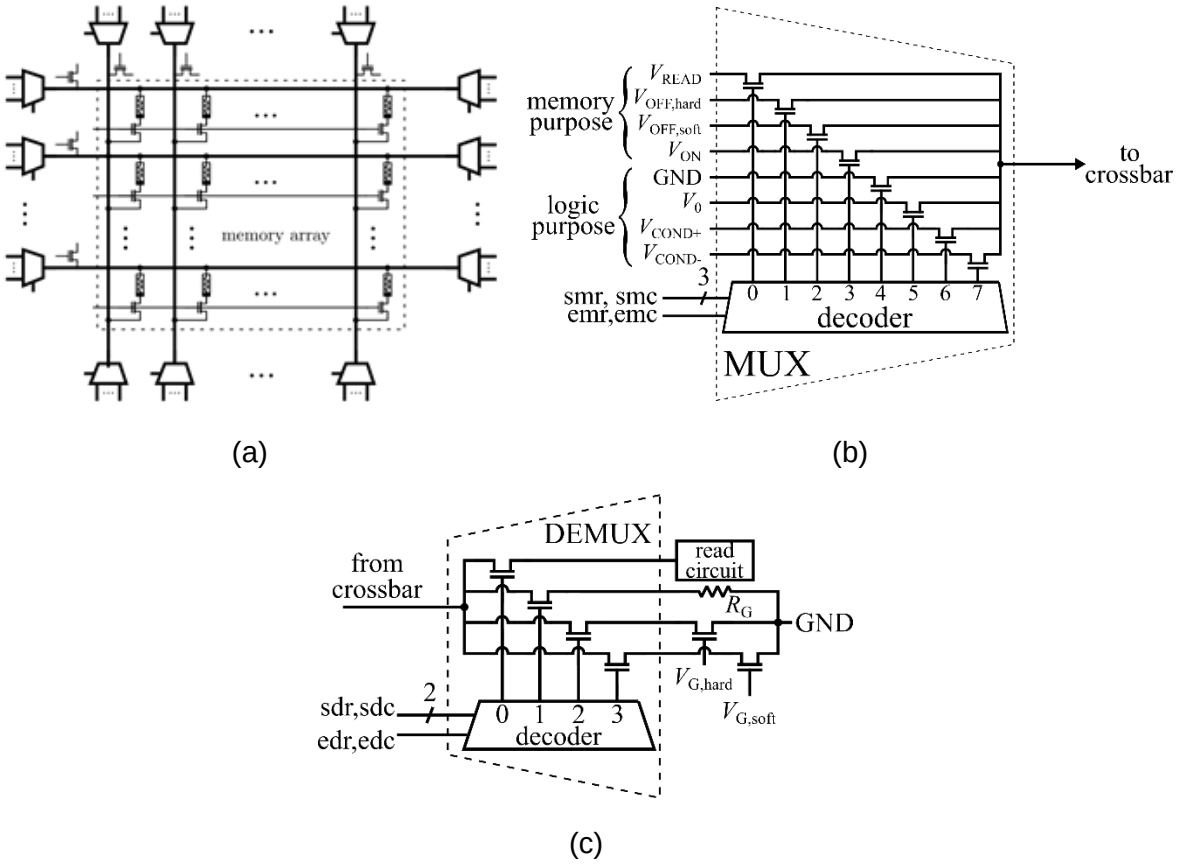


Fig. 5-13. Simulated system overview. (a) The crossbar memory array and its peripheral circuitry for the requirements of this study. (b) Detailed schematic of the voltage multiplexer unit. (c) Detailed schematic of the demultiplexer unit, providing all the necessary connections needed both for memory and logic operations.

The system was designed particularly to support all features necessary in MAGIC and CNIMP logic styles (apart from the typical memory operations.) So, the area overhead of the peripheral circuitry is not an issue here since it will be severely reduced if just one logic style is selected. Even though the system is capable of performing different memory and logic operations, not all peripheral blocks and cells are used in each operation. Therefore, input signals must be set in a predetermined configuration depending on the desired operation to be performed. **Table 5-3** shows the setting for all operations considered in the system. For memory operations, i.e. READ ROW, PROGRAM HRS ROW and PROGRAM LRS ROW, the

configuration follows the same pattern explained in Chapter 4, but it was adapted for this system (instead of inverter drivers, there are multiplexers). In the case of logic operations, the different possibilities to compute CNIMP and MAGIC NOR in the crossbar were first explored. It is certainly more versatile to be able to compute having the input data aligned both in rows and columns.

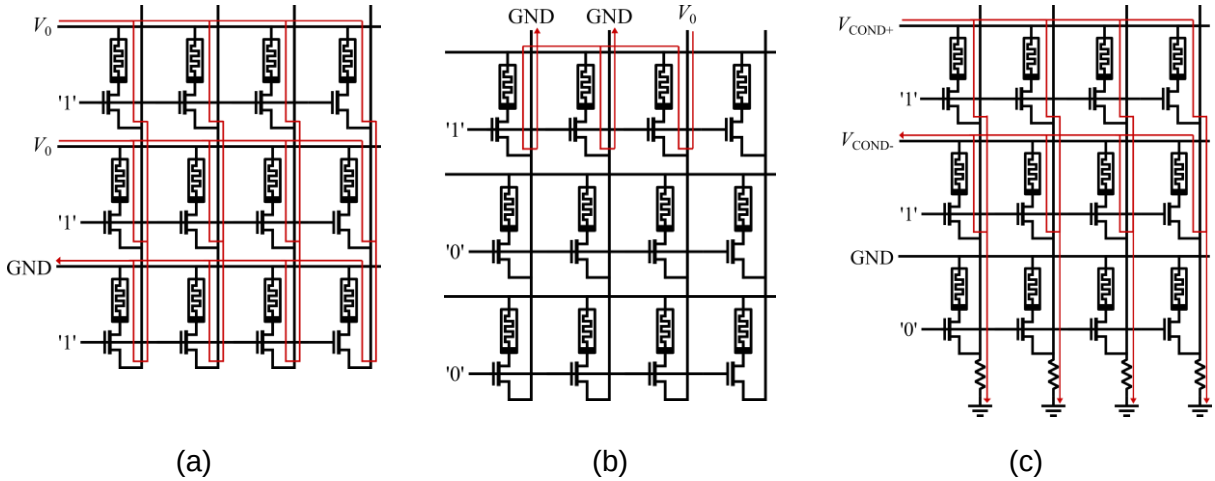


Fig. 5-14. Examples of MAGIC NOR implementation depending on how the input data is spatially stored: (a) in row orientation (one row each time) or (b) column orientation (all columns in parallel). The voltages required for the computation are required and the current paths indicating the memristors involved in the operation are marked in red. CNIMP only admits (c) column orientation (all columns in parallel) with the available voltages in the multiplexers.

In case of MAGIC NOR, there are two possible ways of computation. On one hand, all column-wise operations can be realized in parallel, where two or more selected rows act as input data and an additional selected row stores all outputs (see **Fig. 5-14a**). On the other hand, a single NOR operation is realized using input data found in the same row and having the output stored in the same row as well (see **Fig. 5-14b**). Note that, in order to do so, voltage polarity is reversed since the bottom terminals of the memristors are connected to the multiplexers outputs; input memristors are connected to ground while the output memristor is connected to V_0. Unlike MAGIC gates, there is only one option for CNIMP; parallel computing for all columns, by selecting a row for the p memristors and another for the q memristors (see **Fig. 5-14c**). Although the row operation could be also implemented, different voltage levels, unavailable in the presented multiplexer block, are required. In any case, the parallel computation is the most attractive solution, as it saves a considerable amount of operation steps. In addition, the system supports programming and reading from cells in a row-wise manner. Hence, in such a system a row acts as a memory word, and parallel column-wise computation performs operations between rows (words).

5.2.4 Piecewise Linear File Compiler Tool

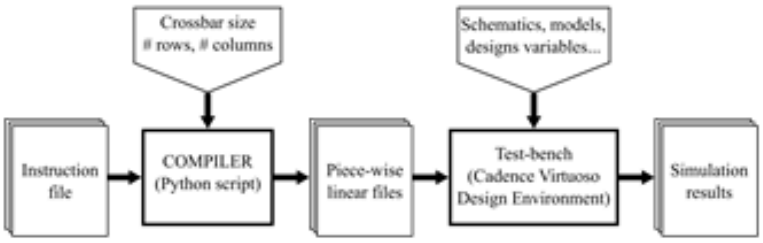
In order to accommodate a large number of logic operations involving several memristors, and to simplify the configuration of every simulation, a Python script called “PWL Compiler” was developed which works similarly to a compiler program. More specifically, it takes as inputs the crossbar size and an instruction file that includes the list of operations to be performed in the system during the simulation, and generates all the required input signals (e.g., enable and select signals for MUX/DEMUX blocks, enable

signals applied to the gate terminal of the row select transistors, the PMOS transistors, etc.) in the form of piecewise descriptions w.r.t. time, as required by the Cadence simulation environment. Once the piecewise linear files are generated by the Python script, the Cadence tool incorporates them along with the schematics, device models, variable values, and then the simulation is executed for the desired scenario. Finally, the simulation results are exported, containing the state of all the memristors of interest, both before and after the execution of the logic operation(s). It is worth mentioning that such "compiler" is versatile enough to permit being extended in order to support more logic styles (by updating the peripheral circuitry blocks), as well as to be optimized to support less logic styles, while incorporating different models of memristors and transistors. The complete simulation flow is shown in **Fig. 5-15a**.

Table 5-3. Signal setting for all memory and logic operations

Operation	***sel***	***emc***	***smc***	***emr***	***smr***	***edc***	***sdc***	***edr***	***sdr***
READ ROW *i*	i-th = 1 others = 0	all 0	X	all 1	011	all 1	00	all 0	X
PROGRAM OFF $b_{n-1}...b_0$ **ROW** *i*	i-th = 1 others = 0	$b_{n-1}...b_0$	001	all 1	100	all 0	X	all 0	X
PROGRAM HRS $b_{n-1}...b_0$ **ROW** *i*	i-th = 1 others = 0	$b_{n-1}...b_0$	010	all 1	100	all 0	X	all 0	X
PROGRAM ON $b_{n-1}...b_0$ **ROW** *i*	i-th = 1 others = 0	all 0	X	all 1	000	$b_{n-1}...b_0$	10	all 0	X
PROGRAM LRS $b_{n-1}...b_0$ **ROW** *i*	i-th = 1 others = 0	all 0	X	all 1	000	$b_{n-1}...b_0$	11	all 0	X
MAGIC n **NOR r** i $j_1 \dots j_n$ j_{out}	i-th = 1 others = 0	$j_1 \dots j_n = 1$ $j_{out} = 1$ others = 0	$j_1 \dots j_n = 100$ $i_{out} = 101$ others = X	all 0	X	all 0	X	all 0	X
MAGIC n **NOR c** $i_1 \dots i_n$ i_{out}	$i_1 ... i_n = 1$ $i_{out}= 1$ others = 0	all 0	X	$i_1 ... i_n = 1$ $i_{out}= 1$ others = 0	$i_1 ... i_n = 101$ $i_{out}= 100$ others = X	all 0	X	all 0	X
CNIMP c i_p i_q	i_p , $i_q = 1$ others = 0	all 0	X	i_p , $i_q = 1$ others = 0	$i_p = 110$ $i_q = 111$ others = X	all 0	X	all 1	01
X : don't care, i: row number, j: column number									

The instruction file is a sequential list of all the different possible operations supported (i.e. HRS and LRS programming, READ, MAGIC NOR, CNIMP and RATIOED NOR, which is described next in this Chapter) along with several arguments related to the position in the array of the memristors that are involved in every operation, or other specific features such as the number of inputs of a logic operation, etc. The set of instructions admitted by the PWL Compiler are listed in **Table 5-4**, related with the ones previously indicated in **Table 5-3**. An example of a simulation file for a MAGIC NOR2 with previous inputs programmed at "01" and memristors read before and after the simulation, is given in **Fig. 5-15b**.




```
SET 0 0          //LRS programming of device in row 0 and column 0
RESET 0 1        //HRS programming of device in row 0 and column 1
SET 0 2          //LRS programming of device in row 0 and column 2
READ r 0         //Read content of row 0
MNOR 2 r 0 0 1 2 //Perform MAGIC NOR2 in row 0, with inputs in
                 columns 0 and 1, output in column 2
READ r 0         //Read content of row 0
```

Fig. 5-15. (a) Description of the simulation flow. (b) Example of an instruction file for a MAGIC NOR2 gate with previously programmed inputs in "01", where the sequence of operations is shown on the left. Some comments about the details of each operation are given on the far right side, starting with //.

Table 5-4. Instruction set supported by the PWL Compiler script

Instruction	Description
SET i j	Program cell (i,j) to LRS. Sequence of operations: **PROGRAM OFF** 0…$(1)_j$...0 **ROW** i **PROGRAM LRS** 0…$(1)_j$...0 **ROW** i
RESET i j	Program cell (i,j) to HRS. Sequence of operations: **PROGRAM ON** 0…$(1)_i$...0 **ROW** j **PROGRAM HRS** 0…$(1)_i$...0 **ROW** j
READ r i	Return content of cells in i-th row. Operation: **READ ROW** i
MNOR n **r** i j_1 … j_n j_{out}	Compute n-input **MAGIC NOR** in i-th row for inputs in columns j_1 j_2 … j_n and output in column j_{out}. Operation: **MAGIC** n **NOR r** i j_1 … j_n j_{out}
MNOR n **c** i_1 … i_n i_{out}	Compute n-input **MAGIC NOR** in j-th column for inputs in rows i_1 i_2 … i_n and output in row i_{out}. Operation: **MAGIC** n **NOR c** i_1 … i_n i_{out}
CNIMP c i_p i_q	Compute **CNIMP** in i-th for inputs in columns ji j2 j and output in column j. Operation: **CNIMP c** i_p i_q

5.2.5 MAGIC Logic Gates

Regarding MAGIC, the proper design space for NOR2 and NOR4 was first explored and the impact of variability was then evaluated for both gates, as well as for the AOI22 logic function. Note that the AOI22 does not have to be designed as a separate logic operation, but it is designed instead using sequential NOR2 operations. In the proposed design process, the V_0 input voltage value was swept while monitoring the state of all the memristors after the logic operation. The applied V_0 voltage pulse width was 200 ns, time

long enough to perform the switching of the output memristor. If any input memristor did not hold its initial logic level, or if the logic level of the output memristor was not the expected one, then the specific V_0 value was not included in the design space. This process took place for every distinct input combination of the states of the input memristors ("00", "01/10" and "11").

Fig. 5-16a presents the results for NOR2, i.e. the resistance of the input and output memristors after the logic operation. In the x-axis the valid V_0 range is shown, extending from 1.89 V to 2.19 V. The initial HRS ($R_{HRS,ini}$) and initial LRS ($R_{LRS,ini}$) is marked by a black dashed line. The higher bound of the V_0 range is found in the "00" input case as the state of the input memristors enter the undefined region when $V_0 >$ 2.19 V. Likewise, the lower bound is found in the "11" input case, when the state of the output memristor enters the undefined region when $V_0 <$ 1.89 V. Note that in the "00" input case, the input memristors do not switch their state from HRS to LRS; however, we notice a state drift (see **Fig. 5-11b**) towards the undefined region, due to the considerable voltage drop on the input memristors, which is insufficient to cause a complete switching event but it is high enough to slightly modify their state while the V_0 pulse is applied. State drift is undesirable as it can cause wrong evaluations in subsequent computations that may use these devices as input memristors.

Similarly, the design of the NOR4 gate is performed in the same way but involves checking more input cases. Here V_0 is bounded from 1.89 V (for any combination having two inputs at '1', such as "0011") up to 2.47 V (for the "0000" input case). Interestingly, **Fig. 5-16b** shows how the design space of NORn is modified with increasing fan-in. The range of valid V_0 values is different for different numbers of inputs, first getting wider up to a particular point, and then apparently shrinking significantly.

Once the design space for the NORn gate was defined, variability was injected to the involved memristors. The error rate of NOR2 and NOR4 for variability factor $k = 5$ is displayed in **Fig. 5-16c** and **Fig. 5-16d**, respectively. The error rate is shown separately for each input case (equivalent input cases are grouped in the same category; e.g. "10" and "01", or "0001" and "0100", since what is important is just the number of logic '1' in the input combination rather than the specific bit sequence). Apparently, the most affected case is the "all-zero" input as V_0 increases. We observed that, the higher the applied voltage, the larger the state drift observed in the input memristors. Although all V_0 values within the design process should in principle be valid, it turns out that the error rate in some cases is unacceptable. The average error rate is minimized for V_0 = 1.95 V in NOR2 and for V_0 = 2 V in NOR4, respectively, highlighting that memristor device variability must be taken into account in the design process! Note that, in NOR4 the average error rate seems more flat than in NOR2. This, owing to the fact that most input cases exhibit similar low error rates, whereas the critical case is just the "all-zero" case, so it weighs less in the computation of the average value.

Next, the error rate of AOI22 was computed. Since AOI22 was implemented via sequential NOR2 operations, the design space of the NOR2 gate was considered and only values around V_0 = 1.95 V were tested, i.e. around the value that minimized the error rate for NOR2. **Fig. 5-16e** shows the results for variability factor $k = 5$. The error rate is detailed for each grouped input case. Equivalent input cases are

grouped in pairs, each corresponding to the inputs of one of the two first-level NOR2 gates. For instance, the "00|01" also includes "00|10", "01|00", and "10|00", which are equivalent. We observed that the most error-prone cases are the ones with a "00" input pair. This was somehow expected since "00" is the input combination with the highest error rate in NOR2 (**Fig. 5-16b**). Thus, in AOI22 the first-level NOR2 gates produce quite unreliable outputs that are used next as inputs in the second level, to eventually produce an even less reliable final output. Moreover, the average error rate, here minimized for V_0 = 1.9 V, does not change significantly with V_0. We also observe that the error rate curve, even for the most critical input case, is not as steep as in previous results. Generally, in agreement with NOR2 and NOR4 functions, it seems that using higher voltage amplitudes results in an increasing error rate. However, the error rate for AOI22 shows higher values than for the NOR2 or NOR4 operations, reflecting the undesired consequence of chained logic operations. A logic operation with a minimum error rate of 15.6% ("00|00" input for V_0 = 1.9 V) is clearly not feasible. It means that, after every NOR2 operation, the state of the output memristor should be properly rewritten to minimize the propagated error rate.

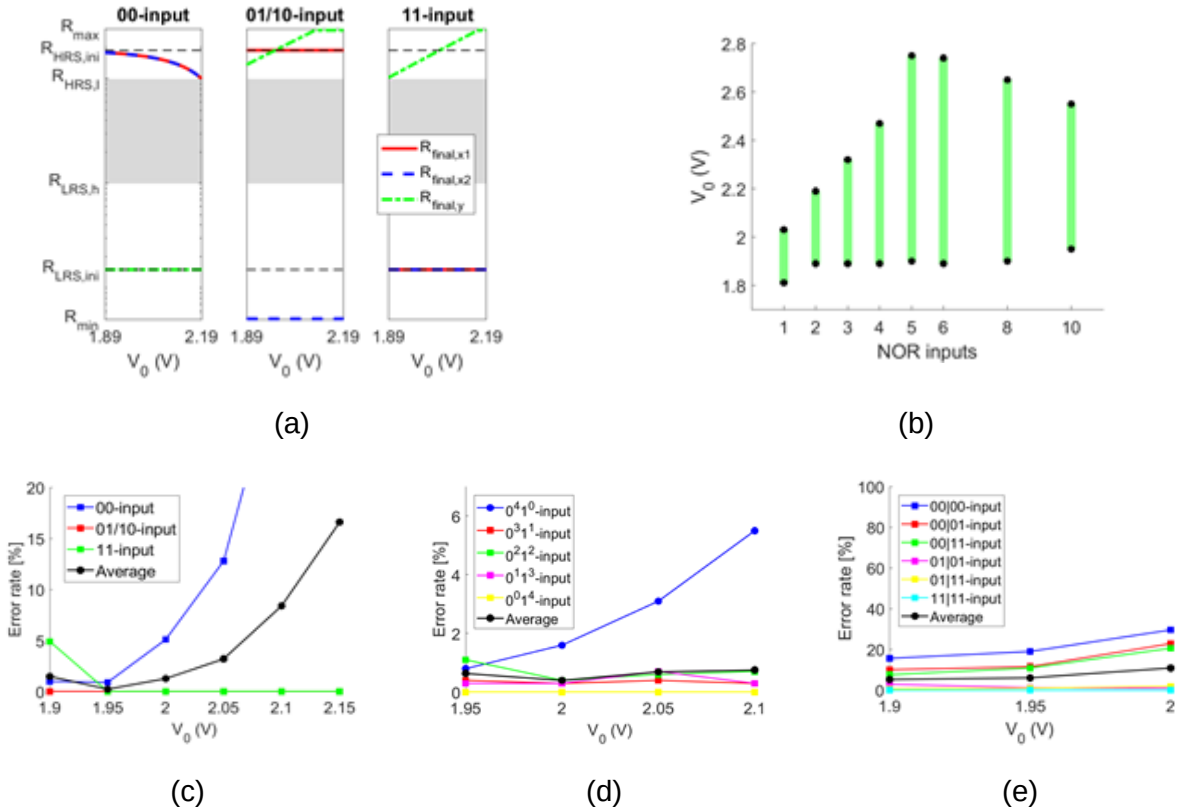


Fig. 5-16. Simulation results for MAGIC logic style. (a) Design space of NOR2 concerning all possible input combinations. (b) Design space for NORn for different numbers of inputs. (c) Error rate of NOR2 after the evaluation of 1000 operations for each possible input combination and V_0 = {1.9, 1.95, 2, 2.05, 2.1, 2.15} V. (d) Error rate of NOR4 for each possible combination shown grouped in the form 0x1y; i.e. combinations that involve x zeros and y ones in their binary representation, and V_0 = {1.95, 2, 2.05, 2.1} V. (e) Error rate of AOI22 evaluated over 1000 operations for every possible input combination grouped as two-input pairs of NOR2 input combinations of interest, these being 00, 01/10 and 11, for V_0 = {1.9, 1.95, 2} V.

5.2.6 IMPLY Logic Gates

For CNIMP, the only design necessary is that of the basic CNIMP operation which is used to perform any other logic function. Once this is completed, the NOR2, NOR4 and AOI22 are simulated while injecting the same amount of variability to finally compute the error rate. Contrary to the MAGIC case, in

the CNIMP design there are three design parameters: V_{COND+}, V_{COND-} and R_G. So, in order to keep the design process as simple and short as possible, we reduced the test set of V_{COND+} and R_G values and rather swept only V_{COND-}. The applied input voltage pulse width was again 200 ns as in the case of MAGIC logic operations.

The corresponding results are shown in **Fig. 5-17a**. Different designs were obtained for different R_G values. Then, NOR2, NOR4 and AOI22 operations were tested to obtain the error rate, as shown in **Fig. 5-17b**, **Fig. 5-17c** and **Fig. 5-17d**, respectively. The R_G = 10 kΩ and V_{COND+} = 1 V were fixed, whereas V_{COND-} was tested for -1.1 V, -1.2 V, and -1.3 V. Likewise in the MAGIC case, once again the results prove that the error rate is very much dependent on the design parameters. Also, we observe that the CNIMP NOR2 error rate is found lower than the MAGIC NOR2 but, as the number of inputs increases, the error rate of the CNIMP gate results higher than the corresponding MAGIC gate (see for NOR4). We even note a set of parameter values that leads to 100% error rate for the "0000" input combination. This is an undesirable effect of state drift, since NOR4 is computed using a chain of four CNIMP operations. Furthermore, the AOI22 results confirm the fact that, although the function is implemented with sequential NOR2 operations, the error rate is not only higher than NOR2 but actually prohibitive. Generally, in CNIMP-based implementations the chained operations are even more critical than in MAGIC as every NORn involves a chain of basic CNIMP operations. Likewise, in NOR4, we confirmed that V_{COND-} = -1.3 V produces 100% faulty results for certain input combinations. Unfortunately, it is not easy to establish any correlations between NOR2 and AOI22 faulty input cases as it was for the MAGIC AOI22. However, it is noticeable that input combinations with a high number of logic '0' and also those including a few logic '1' (for instance, "1000", "0100" or "1100"), evaluated in early stages of the computation, are the ones with the highest error rate. Finally, the minimum average error rate was displaced here (w.r.t. that in CNIMP NOR2) to V_{COND-} = -1.1 V.

All in all, MAGIC and CNIMP were proved quite sensitive to variability in memristor states even after being properly designed, and while considering quite a large resistance window. What is more, since large voltages are generally required, the state drift issue appears. So, chained operations may worsen the situation by increasing the amount of faulty operations to prohibitive levels and thus requiring for extra intermediate steps to restore the state of memristors in-between logic operations.

Furthermore, in **Fig. 5-18**we show simulation results for some of the scenarios shown previously, while considering CMOS nonidealities and having all blocks in the peripheral circuit designed using 0.35 μm standard CMOS models. Monte Carlo simulations were carried out including the best possible levels of process variability to highlight the potential impact of CMOS device mismatch on the memristive logic operations. Generally, the same trends are observed and the error rate now increases in both MAGIC and CNIMP gates. So, the CMOS components could eventually worsen the performance of logic operations and an optimized design of the driving circuit is required, along with a potential adjustment of other parameters related to memristors (V_0, V_{SET}, V_{RESET}, V_{COND+}, V_{COND-}, ...) so as to minimize the error rate. Therefore, it is expected that a logic scheme which does not imply conditional switching of the memristor

states, would most likely constitute a more viable and robust approach to follow for crossbar-based in-memory logic computations.

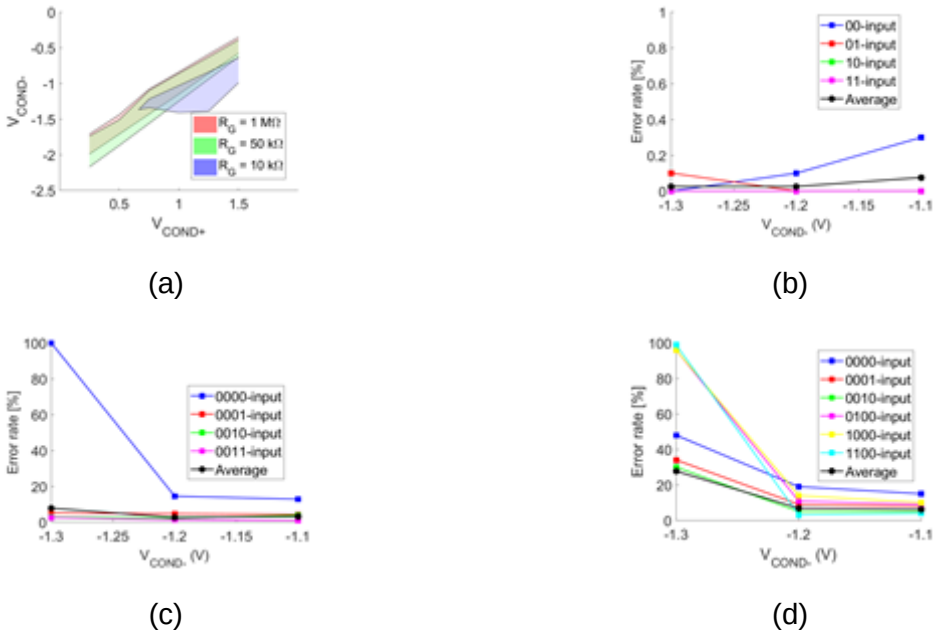


Fig. 5-17. Simulation results for the CNIMP logic style. (a) Design space of CNIMP operation using a set of R_G values of {10, 50, 100} kΩ. (b) Error rate of NOR2 for R_G = 10 kΩ and V_{COND+} = 1 V, each input combination evaluated 1000 times for each V_{COND-}. (c) Error rate of NOR4 for R_G = 10 kΩ and V_{COND+} = 1 V, each input combination evaluated 1000 times for each V_{COND-} (showing only the input cases with higher error rates). (d) Error rate of AOI22 for R_G = 10 kΩ and V_{COND+} = 1 V, showing only the input cases with higher error rates.

Fig. 5-18. Simulation results for error rate while considering CMOS mismatch for (a) MAGIC NOR2 and (b) CNIMP NOR2. 500 operations were realized in each case, with circuit parameters as described in **Fig. 5-16c** and **Fig. 5-17b**, respectively.

5.3 Variability-Aware Logic Design Style for Higher Reliability

The former Section highlights the sensitivity of MAGIC and CNIMP styles to variability and the extra difficulty the latter imposes to the design parameters. Although the regeneration of the correct resistive states after an amount of sequential logic operations might be an option, it adds undesirable delay in the computation. In this direction, this Section presents a novel logic design style that is variability tolerant and compatible with crossbar architectures, being also combined with a straightforward design procedure. The improvements in operation yield, as shown through simulation results, prove its superiority against the rest of logic design styles explored in this Chapter.

5.3.1 Memristor-Based Ratioed Logic Style and Design Constraints

Memristor-based Ratioed Logic is a stateful logic style for building memristor-based logic gates, inspired on the pseudo-NMOS logic design, whose logic gates are compatible with crossbar memory arrays. It works similar to the CMOS-like logic [85] described previously in this Chapter, but it involves less circuit complexity. Two different possible implementations of logic gates are shown in **Fig. 5-19a**, which concern a resistor R_L connected to a pull-down (pull-up) memristor network (the version of the PMOS transistor and the pull-down memristors is assumed in the rest of this Chapter). In fact, the memristors substitute the transistors in the NMOS pull-down (or PMOS pull-up) block of a conventional ratioed logic gate. Moreover, the memristors hold the input data; i.e. a memristor in LRS substitutes a turned-ON transistor, whereas a memristor in HRS substitutes a turned-OFF transistor. The logic operation is performed by applying a voltage V_{READ} across the entire circuit, while making sure that the voltage drop on the memristor network is appropriate for sensing the input states without disturbing them. Contrary to MAGIC and CNIMP, both of which have a resistive output, here the logic output corresponds to a voltage level V_{out} whose value falls between V_{READ} and a reference voltage (which is ground by default). V_{out} is then compared to a threshold value V_{COMP}, properly selected so that V_{out} outcomes are classified in '0' and '1' logic levels. V_{COMP} and R_L values depend on the built logic gate.

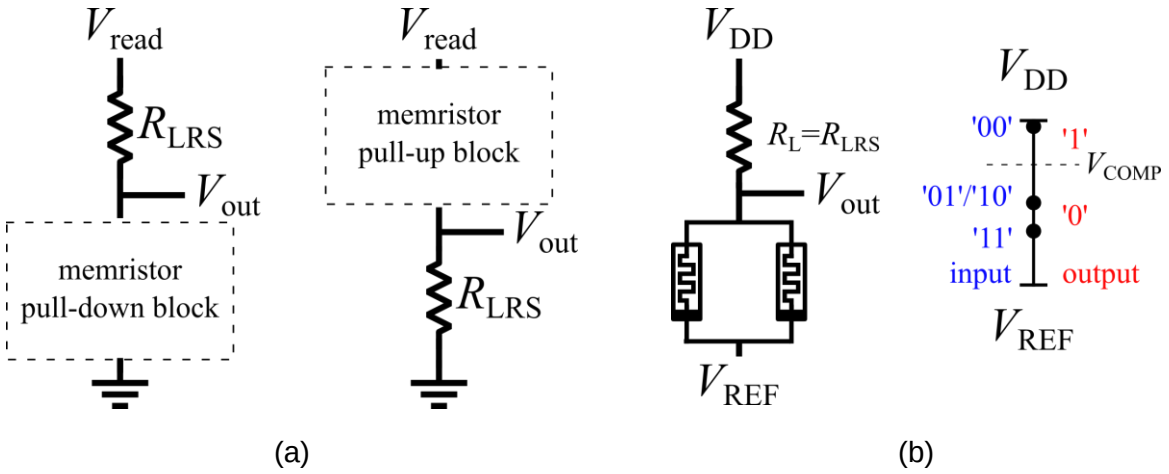


Fig. 5-19. Memristor-based Ratioed Logic style. (a) General scheme of logic gates; (b) Example of a NOR2 gate using the memristor pull-down version and the expected voltage levels regarding all input combinations.

A particular example of a NOR2 gate is shown in **Fig. 5-19b**. The overall circuit works as a voltage divider with an output voltage as described in (45), where R_{eq} is the whole memristor network equivalent resistance and R_L is R_{LRS}. V_{COMP} is set between $(V_{read})/2$ and V_{read}; output is '1' if $V_{out} > V_{COMP}$ and '0' otherwise. When input is "00" the input memristors are both in R_{HRS}, so V_{out} is roughly V_{read}, which is interpreted as logic '1'. On the other hand, if at least one input is '1' (R_{LRS}) then $V_{out} < (V_{read})/2$ falls in the range corresponding to logic '0'. Avoiding the conditional switching of memristors during logic computations is beneficial in many different aspects, and the advantages of the proposed ratioed logic scheme are further demonstrated in the rest of this Section. Note that the result of computation is not simultaneously stored in the memory array; so, it is reasonable to talk about near-memory computing in

this case. However, the result can be stored back to a memory element right afterwards through an additional programming step.

$$V_{out} = \frac{R_{RLS}}{R_{eq} + R_{RLS}} \tag{45}$$

5.3.2 1T1R Crossbar Modifications

The 1T1R crossbar system shown in Section 5.2.3 can be adapted to perform successfully a ratioed NORn logic gate. For instance, **Fig. 5-20** shows a scheme concerning an implementation in a single 1T1R crossbar row. The whole logic operation has been simplified using a control signal called "write/op". This signal controls most of the peripheral blocks. The logic operation is performed when "write/op" is '0'. A pull-up PMOS transistor, in substitution of the resistor R_L, is turned ON and tuned to exhibit a channel resistance R_{LRS} since ratioed circuits depend on proper pull-up/down resistance for correct operation. The use of a PMOS transistor allows to disconnect this element when performing other tasks in the row, such as typical memory operations. However, in order for this logic scheme to work properly, a higher voltage is used in the PMOS source, namely V_{DD}. For that reason, the reference voltage of the logic gate must shift from 0 V to $V_{REF} = V_{DD} - V_{read}$ in order to preserve the V_{read} voltage drop in the entire circuit (logic gate). In the same way, V_{COMP} is shifted by adding $V_{DD} - V_{read}$ to the previously selected voltage.

Hence, with *write/op* = '0', V_{REF} and V_{DD} pulses are applied in the selected crossbar row (enable signals "en_1" … "en_n" define which memristors are involved) and two different cases are observed depending on the states of the input memristors: $V_{out} < V_{COMP}$ or $V_{out} > V_{COMP}$. After V_{out} is compared with V_{COMP}, the output logic level is stored temporarily in a flip-flop. Afterwards, the operation finalizes by storing back the output into a memristor by setting *write/op* = '1' and choosing the storing cell with the enable "*en*" signal. The transmission gate also turns ON a PMOS transistor to limit the current when programming the state of the memristor. Note that the implementation of **Fig. 5-20** constitutes a simplification of the modifications for the system described in Section 5.2.3; actually, in our real implementation the voltage V_{REF} is supplied by means of the column multiplexer as another voltage available, while the comparator and the flip-flop is another output of the row demultiplexer. Additionally, a digital circuit that enables and selects the V_{REF} voltage for the selected memristor to store the output is required. The flip-flop and the comparator certainly add some area and power overhead, whereas the rest of the peripheral circuitry is in fact also required in the case of MAGIC and CNIMP gates.

After covering crossbar implementation, it is worth mentioning here some other dimensions in which logic design could be benefitted via the proposed ratioed scheme. MAGIC and CNIMP allow for row-wise and column-wise operations in the crossbar array, which could be executed in parallel if a 1R passive array is considered. However, for the ratioed logic a 1T1R crossbar is preferred, because the logic operation is equivalent to sensing a voltage level and this could be affected by the rest of the crossbar if no select transistors are used in the cross-points. Moreover, if the selector transistors are connected in rows,

then NOR operations can be performed in a row-wise or column-wise manner, but only column-wise parallel operations are possible, as shown in **Fig. 5-21**.

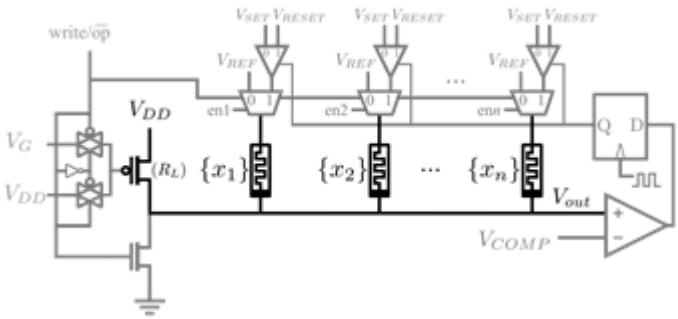
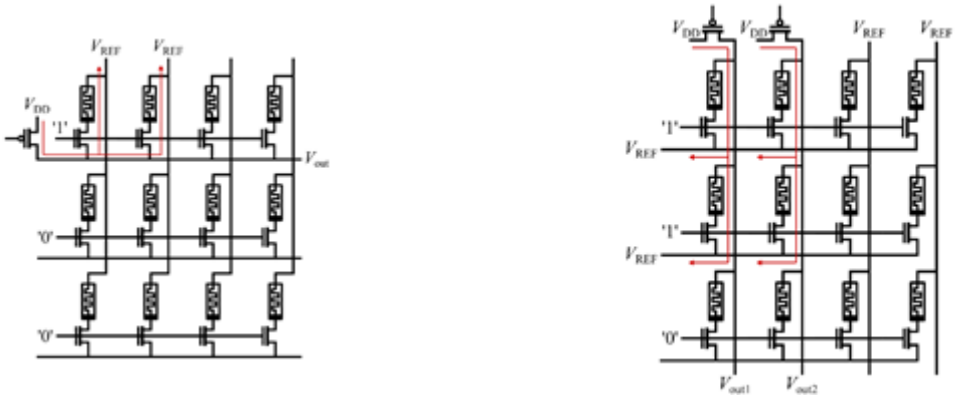


Fig. 5-20. Implementation of a Ratioed NORn gate in the CIM system described in Section 5.2.3.

Furthermore, the ratioed logic scheme accepts several modifications to enable other useful logic functions in a very compact way. For instance, ORn gates are possible just by including an inverter stage after the comparator (or by using instead a parallel memristor pull-up block). With the ORn gate directly available, a particular case for $n = 1$ is the COPY gate, allowing to copy the content of a particular cell in another one in just one step, unlike MAGIC and CNIMP which would need two steps (two NOT operations) and conditional switching of memristors. Therefore, assuming that NOR, OR, NOT and COPY are readily available, **Table 5-5** shows the cross-point area needed to compute a NOR, OR, AND or NAND gates for each one of the three logic styles compared here, aiming to provide a perspective of the necessary resources in terms of area. {IN, AUX, OUT} columns correspond to memristors holding the input data (always 2 for 2-input gates), the number of auxiliary devices to hold intermediate results in chained operations, and eventually a single memristors to hold the output value in all cases. Both aligned and not-aligned input memristors are considered, depending on whether they share the same row or column. With aligned input devices, the number of required AUX devices is less in most cases and thus it makes the difference in this metric. Generally, it can be observed that the ratioed logic scheme outperforms the rest, hence it is proved to be the most compact solution among the three.



(a) (b)

Fig. 5-21. Possible settings to perform a ratioed logic NORn in a 1T1R crossbar. (a) One row-wise NOR2, (b) An example of two column-wise NOR2 done in parallel, giving V_{out1} and V_{out2} outputs. The voltage pulse levels, the involved PMOS devices, and the selected rows, are indicated in the schematic.

Table 5-5. Number of cross-points required for each logic style

Logic Style	Input Position	NOR			OR			AND			NAND		
		IN	AUX	OUT	IN	AUX	OUT	IN	AUX	OUT	IN	AUX	OUT
MAGIC	Not aligned	2	2	1	2	3	1	2	2	1	2	3	1
	Aligned	2	0	1	2	1	1	2	2	1	2	3	1
CNIMP	Not aligned	2	2	1	2	3	1	2	2	1	2	3	1
	Aligned	2	0	1	2	1	1	2	2	1	2	3	1
Ratioed Logic	Not Aligned	2	1	1	2	1	1	2	2	1	2	2	1
	Aligned	2	0	1	2	0	1	2	2	1	2	2	1

Finally, regarding scalability, the CNIMP (or IMPLY) and MAGIC mapping has been studied in some recent works [93], [94]. Mapping relies on the capability of parallel row- and column-wise operations with NOR and NOT gates, so optimized algorithms for faster and less area-/power-consuming logic realizations have been proposed. Likewise, given the structural similarities in the circuits representing the logic gates, the memristive ratioed logic style also has the same potential to allow efficient mapping, while using slightly different strategies than for MAGIC or CNIMP.

5.3.3 Simulation results

The simulation results of the alternative ratioed-logic scheme, concerning the design exploration and yield against memristor variability, are presented here and discussed/compared with those of MAGIC and CNIMP. Regarding the specific design used in simulations, the voltage difference $V_{DD} - V_{REF}$ must be selected low enough to not disturb the state of the input memristors. Using $V_{READ} = 0.5$ V as a non-disturb voltage, we applied $V_{DD} = 3.3$ V to operate the 0.35 μm PMOS load transistor and set $V_{REF} = 2.8$ V. Moreover, the PMOS channel resistance value was achieved very similar to the LRS resistance of memristors, through proper sizing and by using a gate voltage of 1.6 V. Moreover, since no switching of any memristor is required, the voltage pulse width was selected 20 ns, which is much shorter compared to the pulses applied in the case of MAGIC and CNIMP gates.

Regarding the evaluation of the variability impact, it should be noted that the error rate of NOR2 is typically zero, since there is no conditional switching of any device here and the logic result is written back to a memory device in a controllable way. So, it was meaningless to use the error rate as a metric of evaluation in this case. Alternatively, the distributions of the V_{out} values for each different input combination, was considered more illustrative for this logic style. The simulation results are shown in **Fig. 5-22a**. Each distribution is colored depending on the variability factor k used in the simulation, i.e. blue for $k = 1$, red for $k = 5$ and black for $k = 10$, whereas the insets show separately the details of each distribution. As expected. in the "00" input case V_{out} falls near V_{DD} and has a very narrow distribution. On the other hand, "01/10" and "11" input cases have broader distributions that are centered in particular values. The results

confirm that the three cases are clearly distinguishable and a comparison voltage V_{COMP} can be easily defined to interpret correctly the values corresponding to logic '1' or logic '0'. The NOR4 results in **Fig. 5-22b** show similar performance. Indeed, as there are more input memristors in parallel, the V_{out} distributions are displaced further from V_{DD}. So, V_{COMP} clearly separates the input cases that give a logic '1' output from those that give a logic '0' output.

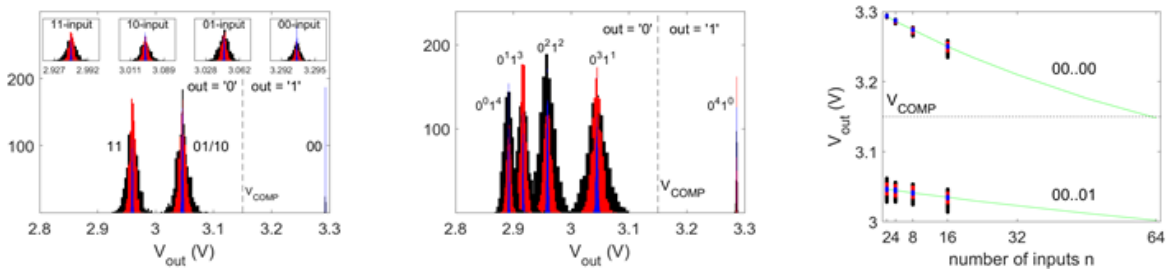


Fig. 5-22. Simulation results for the ratioed logic style. (a) Histograms of V_{out} for NOR2 after the evaluation of 1000 operations for each possible input combination when k = 1 (blue), k = 5 (red) and k = 10 (black). (b) Histograms of V_{out} for NOR4 after evaluating 1000 operations for each possible input combination for k = 1, 5 and 10. (c) Maximum fan in exploration for the NOR operation, using k = 1, 5 and 10 (1000 evaluations for each distribution). The results concern the "all-zeros" input and those combinations having only one input at logic '1'. The green line shows V_{out} when k = 0.

Next, **Fig. 5-22c** aims to underline the really high tolerance of the proposed scheme to device variability. The graph shows the V_{out} range evolution with increasing number of inputs of the NOR gate. The output is shown only for the all-zero input case ("00…00") and for the combinations with only one input storing a logic '1' (indicated as "00…01"), since their results are closer to V_{COMP}, as seen in **Fig. 5-22a,b**. The green line shows an extrapolation following the results without any variability for input number n = 2, 4, 8 and 16, whereas the simulation results shown were taken after the injection of variability. It can be noticed that the two cases remain separated enough so as to be properly distinguished by the comparator. Eventually, the "00..00" case theoretically intersects the V_{COMP} line approx. when n = 64! However, such limitation could be erased by justifying accordingly the V_{COMP} level to the number of inputs. Hence, only when the V_{out} values for these two extreme input combination cases are too close with each other, then the error rate will be non-zero for the ratioed logic scheme.

We did not include any simulation results for AOI22 since the performance of the basic operation, i.e. NORn, was found a priori superior than in the other logic schemes of interest and practically "error-free". On the other hand, since output here is expressed in voltage instead of memristance, conducting chained operations in this scheme assumes adding an intermediate step to store the logic output to a memristor state which will later act as input variable. Therefore, in principle the delay of computations in the ratioed scheme would be higher. However, as commented before, unless an intermediate "regenerating" step is included both in MAGIC and CNIMP gates, chained operations in either of these styles is unacceptably error-prone. So, after all, requiring an extra step to store intermediate results in rationed logic operations is not that much a disadvantage, considering the high gains in reliability and also the much

shorter computation time which involves only sensing the memristor states. Finally, no simulation results are shown here for the impact of CMOS non idealities to the ratioed logic scheme, since we found they have no significant effect on the output voltage levels, thus the error rate was kept zero.

Relying on logic styles that assume frequent device switching, even for the implementation of simple logic functions, actually poses a major concern for the endurance-limited memristor technology. This is why the focus has been gradually moved to alternative schemes that implement logic using just read operations, such as the Scouting Logic [83] seen earlier in this Chapter. The latter is also crossbar compatible and is based on a modified sense amplifier for the implementation of different logic gates. It shares some of the advantages with the proposed ratioed logic scheme, even though the design process is iterative, similar to the one presented for MAGIC and CNIMP. On the other hand, the ratioed logic was proved here robust and with a straightforward and noniterative design process, whereas proper functionality was verified even when scaling the number of logic gate inputs.

5.4 Concluding Remarks

This Chapter analyzed the impact of the memristor variability of different logic styles in a CIM system. Prior to the study, the Chapter reviewed memristor-based digital computing applications. Most of the review material specially focused on CIM systems and logic styles which constitute a new paradigm of computing, taking advantage of the favorable properties of non-volatile memory systems.

Regarding the study of the yield of logic operations, the results showed how two different logic styles that rely on conditional switching/programming of the state of memristors holding the logic output, namely CNIMP and MAGIC, are sensitive to memristor variability. Furthermore, a good control of the design parameters could benefit the yield in the logic operations. Nevertheless, even with an improved yield, problems still arise when chaining logic operations are used. The solution presented is a different approach that avoids the conditional programming and it simply consists of a reliable reading operation, followed by a controllable programming operation, if needed. Results showed how our proposal reaches a notable yield for a smaller number of inputs in a NOR operation, while erroneous output calculations are expected only for a very large number of inputs. Although this logic style alternative requires two steps in order to store the output back to the crossbar, our design requires short pulses to perform the read operation compared with the ones used in the programming operations. Hence, the delay overhead is not significant.

Chapter 6 Analog Applications. Fast memristor-based adaptive chaotic synchronization through fault memristor events

So far, previous chapters mostly focused on digital applications where memristors are treated either as binary elements, or as multi-state elements with quantized memristance. However, the resistive state of ideal memristors varies in an analog manner between the HRS and LRS boundaries, and some real devices also demonstrate an analog switching behavior similar to such ideal switching behavior. This feature has been appreciated in many analog-oriented applications, ranging from signal processing circuits to unconventional forms of computing applied to the solution of complex problems in efficient ways, to name a few. Reliability of memristors may impose additional challenges that can be surpassed in some cases by using tolerant design strategies. There are other cases though where reliability may not be an issue, but instead an asset.

This Chapter first presents a brief review of the most relevant memristor contributions in the field of analog applications. Next, it focuses on the field of chaotic synchronization where contribution of memristors is analyzed in terms of reliability, and a novel solution to accelerate the synchronization process of an arbitrary number of interconnected chaotic circuits is presented. In the following paragraphs, this Chapter introduces the principles of chaotic systems presenting the example of the "Chua's circuit", and lays the foundations of the adaptive synchronization using memristors as coupling elements, information necessary to study further Sections 6.3 and 6.4, previously presented in [95], [96] and [97].

6.1 Review of analog applications

This Section has been divided into four groups, covering very diverse topics which are: programmable signal processing circuits, programmable oscillators, unconventional analog computing and security. In each application group, memristors play different roles depending on the particular characteristics that each application takes advantage of.

There are many signal processing circuit modules -for instance, comparators, amplifiers, filters, to name a few- that use passive elements to tune their performance characteristics. It has been demonstrated that memristors can be introduced in such circuits, mainly with the role of programmable resistors, so as to upgrade functionality on these analog blocks [98]. Signal amplifying stages usually employ resistors to set a closed-loop gain. A simple inverting amplifier can use a memristor to substitute one of the resistors used in this configuration, as shown in **Fig. 6-1a**. Additionally, a programming circuit is also connected, to supply positive or negative short pulses to smoothly decrease or increase the resistance of memristor m, and effectively increase or decrease the close-loop gain. Likewise, comparators also can get benefited by adding memristors. In such a case, using a memristor adds a programmable voltage threshold feature by simply using a voltage divider in which a memristor takes the place of a resistor, as depicted in **Fig. 6-1b**. By changing the memristor state the voltage threshold supplied to the comparator input is also modified. Furthermore, a memristor can substitute a resistor in a Schmitt trigger comparator to change the hysteresis thresholds, as shown in **Fig. 6-1c**. In all the aforementioned cases, a common rule must apply: the memristors

should be exposed to low voltages in the circuits under normal operation, except when the programming pulses have to be applied.

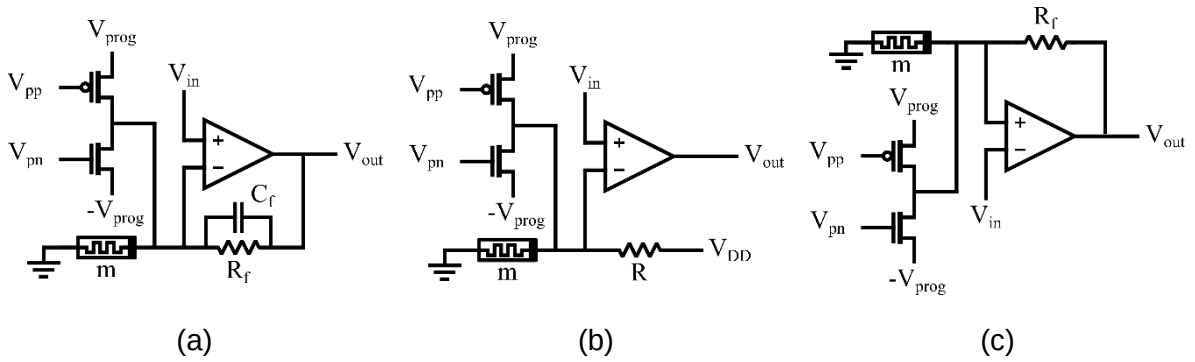


Fig. 6-1. Different programmable analog circuit modules. (a) Inverting amplifier with a capacitor for noise suppression, (b) comparator with programmable threshold voltage, and (c) Schmitt trigger comparator with programmable hysteresis.

Oscillators are another type of circuits that can be realized using memristors. **Fig. 6-2a** shows an example of a relaxation oscillator [98], where again a resistor was substituted by a memristor and a programming circuit is in charge of tuning the memristor state which affects the frequency. In the same direction, thinking of a more compact circuit design, a memristive Sisyphus concept was suggested in [99] as explained in **Fig. 6-2b**. More specifically, by applying voltages at each memristive branch during the different phases of operation, each memristor increases or decreases its resistance, with theSET process being usually faster. Based on this scheme, a clock signal generator [99] was built as shown in **Fig. 6-2c**. The OR gate output defines the operation of the circuit: when output is '0', the resistance is rising as in phase 1 of the memristive Sisyphus, and when the output is '1', the resistance decreases as in phase 2.

For the time being, one of the topics where a great deal of research is done around the world in the field of memristors, has to do with unconventional analog computing paradigms. Although there are different approaches proposed so far in this context, the central idea is to mimic in some degree the behavior of biological systems such as the human brain or living organisms [100], [101], in order to solve complex problems in more power-/time-efficient ways. In this direction, neuromorphic computing takes advantage of the cerebral system, where a massively parallel architecture of neurons connected through a large number of synapses is able to solve different problems avoiding currently known limitations in conventional computing systems such as the Von Neumann bottleneck [102]. Memristor in fact resembles a biological synapse at circuit level: it is a two-terminal component able to connect two elements (in this case neurons), it is plastic, and its connection weight (resistance) can be regulated and it actually depends on the recent activity of signals applied to its terminals.

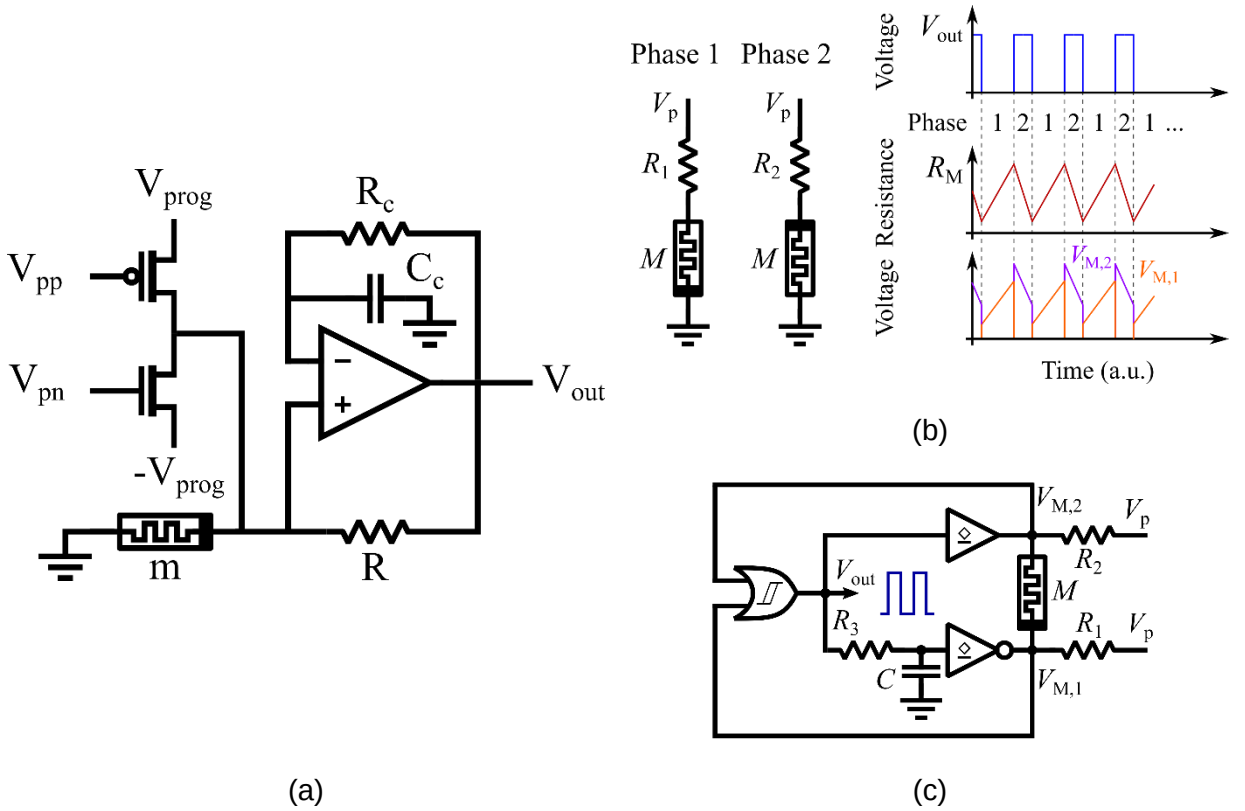


Fig. 6-2. Memristor-based oscillators schemes. (a) Relaxation oscillator, (b) memristive Sisyphus in which a (c) particular clock signal circuit is based.

Moreover, the concept of "Reservoir Computing" is an extension of neural networks in which only the output layer is trained after mapping the input into a fixed dynamical system called reservoir. In this context, it has been shown that memristors could be used as structural elements of the reservoir due to the dynamical behavior and richness they provide. The relevant works in the literature show some examples of reservoirs and applications, such as using a random memristor-based reservoir to recognise signal patterns or reproduce the so-called Pavlov conditioning experiment [103]. Other works have shown reservoirs made of memristors with short-memory properties that recognize basic images and predict specific time-series [104], or used random memristor networks formed by an unorganized fabric mesh for detecting isolated digital speech signals [105]. As far as device variability is concerned, in [106] the memristor variability impact in memristive reservoirs was tested, with more favorable results obtained using a regular mesh rather than random meshes.

Nevertheless, although memristor variability is still a major concern for this device technology, some applications have managed to take full advantage of this otherwise negative issue. Concretely, this happened in hardware security applications for true random number generators (TRNGs) or physical unclonable functions (PUFs). These applications usually rely on different variability mechanisms: weak PUFs based on D2D variability [107], strong PUFs built on variations in the memristor internal state and nonlinear conduction [108], TRNGs enabled by RTN [109], stochasticity [110] or C2C variability [111].

In general, memristor PUFs alternatives have shown great potential compared to their CMOS counterparts, while the same is still not true for TRNG circuits [112].

Last, the development of chaotic oscillators constitutes an active research area in which memristors are also involved. In the implementation of chaotic circuits, owing to the nonlinearity in its switching behavior, the memristor has attracted considerable attention to be used to implement the active nonlinear element in Chua's circuits [113] [114]. In this context, chaotic synchronization is a promising technique with applications in secure communications. Few recent works demonstrated how networks of several Chua's circuits [11] [115] could be synchronized using memristors to provide adaptive coupling properties while being a low-complexity interconnect medium. Therefore, in the following Sections the application of memristor-based chaotic synchronization is explored in detail. Although the chaotic circuit of interest is the Chua's circuit, the analysis exposed in this Chapter could be applied to other existing chaotic circuits.

6.2 Memristor-coupling Two-way Chaotic Synchronization

6.2.1 Chaotic Systems and Circuits

A chaotic system is a dynamical system described with differential equations and characterized by a strong sensitivity to the initial conditions of certain variables. Although it is a deterministic system, its state apparently evolves in an unpredicted way. In fact, their predictability is compromised by factors such as the impossibility of infinite precision in measuring a system or computing a forecast, as shown in [116].

The general form of a chaotic system is shown in (46), where *s* contains the internal state variables of the system and *f*(*s*) describes the dynamics. The phase space is the space representation where possible states are found and the trajectory of the system is the temporal evolution of the system´s state in the phase space.

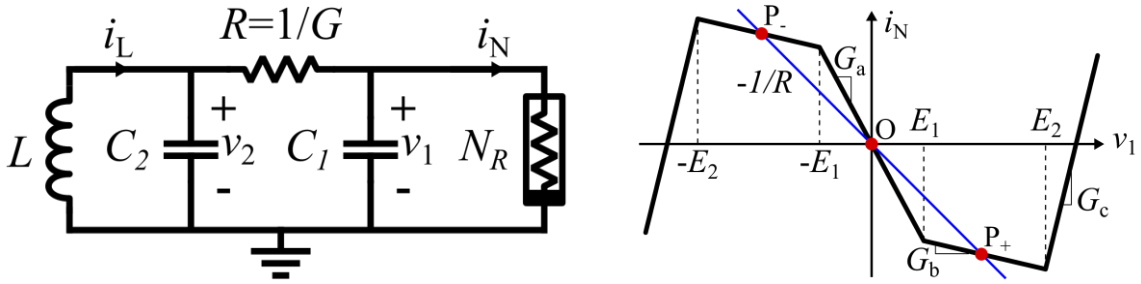
$$\frac{ds}{dt} = f(s) \tag{46}$$

Electrical circuits derived from chaotic systems began to appear in 1983 with the so-called Chua's circuit, which is probably one of the most widely-studied circuits in this field. Later, many other chaotic circuits were proposed, such as the Lorenz or Rossler circuits, to name just a few of them [117]. Chaotic circuits could find application in pseudo random number generators and in secure communications performed via synchronization of chaotic circuits found in the sender and receiver modules. Nevertheless, to the best of the author's knowledge, such applications are still not found in commercial solutions.

6.2.2 Chua's circuit

The Chua's circuit is considered the simplest electronic circuit that implements a chaotic system. As shown in **Fig. 6-3a**, the circuit only uses four passive elements, i.e. a resistor *R* (with conductance *G*), two capacitors C_1 and C_2, an inductor *L*, and a nonlinear active element labelled as N_R which is commonly known as "Chua's diode". The Chua's diode is an essential part of the Chua's circuit which makes possible

a chaotic response. The Chua's diode *v-i* characteristic is defined by eq. (47) and it is depicted in **Fig. 6-3b** with a black thick line. As it can be observed, it can be described as a piecewise linear negative resistance divided in three regions. When v_1 is in the [$-E_1$, $+E_1$] interval, the Chua's diode shows a conductance of G_a, otherwise the conductance is G_b, where $|G_b| < |G_a|$.



(a) (b)

Fig. 6-3. (a) Schematic of the Chua's circuit, and (b) the Chua's diode N_R *i-v* characteristics shown in black thick line. The blue line represents the load *R* whereas P_-, O and P_+ are the equilibrium points of the system.

$$i_N(v_N) = \begin{cases} G_b v_N + (G_b - G_a)E_1 & if\ v_N \le -E_1 \\ G_a v_N & if\ |v_N| < E_1 \\ G_b v_N + (G_a - G_b)E_1 & if\ v_N \ge E_1 \end{cases} \quad (47)$$

The dynamics of this system are described in the state equations of the circuit shown in (48484848), that can be found in a straightforward way by applying Kirchhoff's laws. Note that these equations already have the form seen in (46), where v_1, v_2 and i_L are the state variables of the system.

$$\begin{aligned} \frac{dv_1}{dt} &= \frac{1}{C_1}[G(v_2 - v_1) - g(v_1)] \\ \frac{dv_2}{dt} &= \frac{1}{C_2}[G(v_1 - v_2) + i_L] \\ \frac{di_L}{dt} &= -\frac{1}{L}v_2 \end{aligned} \quad (48)$$

A DC analysis gives more information about the equilibrium points of this dynamic system. In DC, all derivatives of the state equations are 0, obtaining the equations in (49). This analysis is electrically equivalent to substitute capacitors by open circuits and short-circuiting the inductor.

$$\begin{aligned} Gv_1 + g(v_1) &= 0 \\ v_2 &= 0 \\ i_L &= -Gv_1 \end{aligned} \quad (49)$$

Equilibrium points must be evaluated in the three regions of Chua's Diode: i.e. where $v_1 < -E_1$, $-E_1 < v_1 < E_1$, and $E_1 < v_1$. In each region, an equilibrium point is found, P_-, 0 and P_+, as depicted in **Fig. 6-3b**. The equilibrium points coordinates are described in **Table 6-1**.

Table 6-1. Coordinates of the equilibrium points in Chua's circuit system

Point P- ($v_1 < -E_1$ region)	Point 0 ($-E_1 < v_1 < E_1$ region)	Point P+ ($E_1 < v_1$ region)
$V_{1,P-} = -\frac{G_b - G_a}{G + G_b} E_1$ $V_{2,P-} = 0$ $I_{L,P-} = \frac{G_b - G_a}{G + G_b} G E_1$	$V_{1,O} = 0$ $V_{2,O} = 0$ $I_{L,O} = 0$	$V_{1,P+} = \frac{G_b - G_a}{G + G_b} E_1$ $V_{2,P+} = 0$ $I_{L,P+} = -\frac{G_b - G_a}{G + G_b} G E_1$

The dynamics of this system are analyzed using the differential equations shown in (48) which describe the temporal evolution in the 3-dimensional state space (v_1, v_2, i_L). By choosing carefully all the circuit parameters, the circuit will demonstrate a chaotic behavior known as a double-scroll attractor. This behavior is described as spiral trajectories are centered in P_+ or P_- (the attractors) and are growing inwards. When the trajectory is close to P_+ and P_-, it is pushed towards the $|v_N| \leq E_1$ region and subsequently is pushed either to P_+ or to P^-, generating again another spiral and repeating the same process. The results of a MATLAB-based evaluation of the Chua's circuit state equations, to illustrate its chaotic behavior, are shown in **Fig. 6-4**. The Chua's circuit parameters used (adapted from [118]) were R = 1.67 kΩ, C_1 = 10 nF, C_2 = 100 nF, L = 18 mH, G_a = -0.76 mS, G_b = -0.41 ms and E_1 = 1.174 V. **Fig. 6-4a** depicts the space of the state variables and the trajectory of the Chua's circuit dynamics forming the so-called double-scroll attractor shape and the spirals going towards P_+ and P_-. The temporal evolution of the circuit signals is shown in more detail in **Fig. 6-4b**. Although the dynamics are generated by a deterministic system, the number of oscillations around an attractor and succession of spirals centered in P_+ and P_- are seen as unpredictable and non-periodical. Additionally, grey dashed lines mark the values of P_+ and P_- expected coordinates in v_1 and i_L (v_2 is always 0).

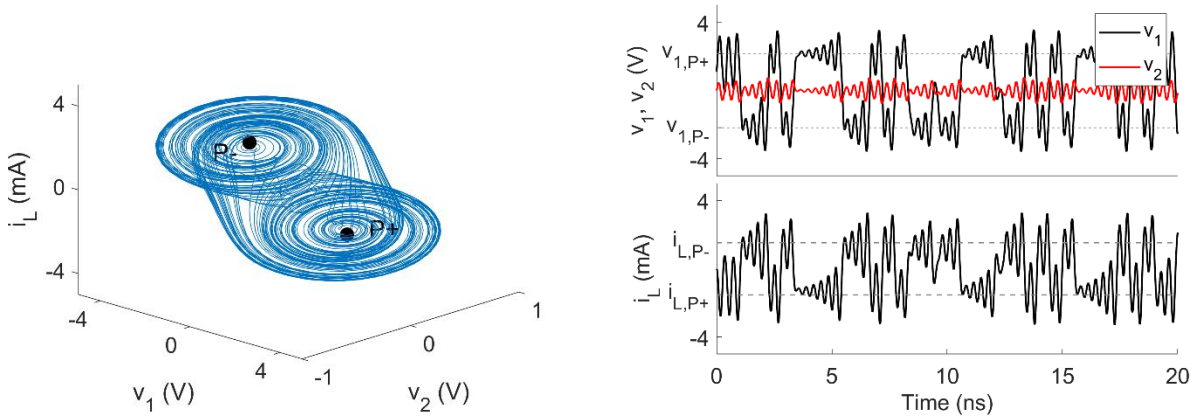


Fig. 6-4. (a) Simulation results for a double-scroll attractor´s behavior in the phase space. (b) Temporal evolution of v_1, v_2 and i_L.

It is usual to transform the state equations in (48) to a dimensionless version for mathematical manipulation purposes. It is possible by defining the transformations $x = v_1/E_1$, $y = v_2/E_1$, $z = i_L/(E_1 G)$, τ

$=tG/C_2$, $a = G_a/G$, $b = G_b/G$, $\alpha = C_2/C_1$ and $\beta = C_2/(LG^2)$. The dimensionless state equations are shown in (4), where $f(x)=bx+0.5(a-b)(|x+1|-|x-1|)$. From now on, the terms v_1 or x, v_2 or y and i_L or z, referring to state variables or signals, will be used interchangeably throughout this text.

$$\begin{aligned} x &= \alpha[y - x - g(x)] \\ y &= x - y + z \\ z &= -\beta y \end{aligned} \tag{4}$$

6.2.3 Implementation of a Chua's Circuit

Although the Chua's circuit was presented as a quite simple and compact circuit, its implementation is not straightforward. For instance, the Chua's diode is not a real device; it needs to be formed using other circuit elements. There are different options regarding the implementation of a Chua's diode. This work opts for one presented in [118], as it requires few elements and it has several options to tune the characteristics of the Chua's diode. **Fig. 6-5a** shows the circuit schematic of this alternative, using two operational amplifiers and six resistors, grouped in two blocks connected in parallel. Each block acts as a negative resistor, whose *i-v* characteristic is shown in **Fig. 6-5b** (top and middle); the resistors determine its shape. The current contribution of each block is summed up and the overall *i-v* characteristic is a five-segment piecewise linear graph shown in **Fig. 6-5b** (bottom) and the parameters depicted are tuned using the eqs. (50). Although there are two additional segments in this case than in the original graph of a Chua's diode, the three central ones match the characteristics. If v_1 never exceeds $\pm E_2$ the dynamic behavior of the Chua's circuit does not suffer any modification. Therefore, E_2 value must be designed to be high enough to not affect the circuit dynamics. The design we have adopted follows the same example of [118], where $R_1 = R_2 = 220$, $R_3 = 2.2$ kΩ, $R_4 = R_5 = 22$ kΩ, $R_6 = 3.3$ kΩ, which leads to $G_a = -0.76$ mS, $G_b = -0.41$ mS, $G_c = 4.59$ mS. Operational amplifiers used were TL082 supplied with ±9V, expecting a $\pm E_{sat}$ = i, thus $E_1 = 1.174$ V and $E_2 = 8.182$ V.

Apart from the implementation of the Chua's diode, the inductor may be another concern depending on the context of the circuit application. When it is not convenient to use an inductor, there is the option to build an inductorless Chua's circuit. In such a case, the inductor is replaced by the emulator circuit from [119] and shown in **Fig. 6-6**. Through eq. (51), the value of the emulated inductance may be tuned. Another advantage in using such a circuit is that state variable i_L can be easily determined indirectly by measuring voltage v_z and using the expression (52).

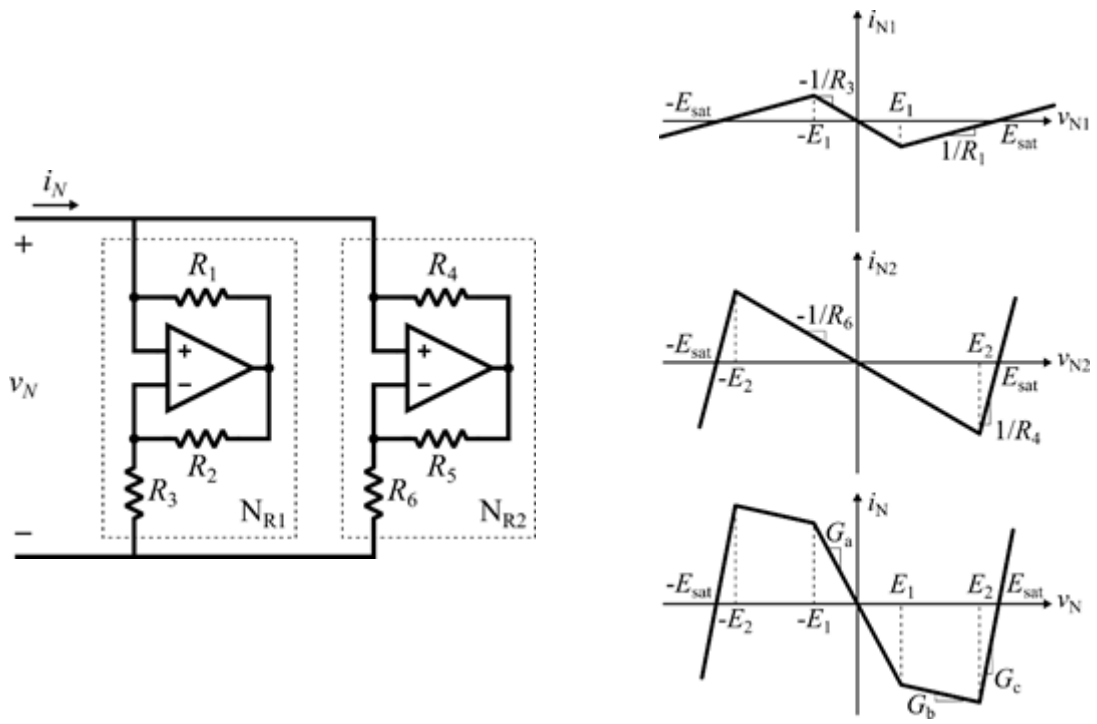


Fig. 6-5. Physical implementation of a Chua's diode adopted from [118]. (a) Circuit schematic of the implementation. (b) Separated i-v characteristics for sub-circuits N_{R1} and N_{R2} (top and middle, respectively), and *i-v* characteristics of the whole circuit (bottom).

$$
\begin{aligned}
G_a &= -\frac{1}{R_3} - \frac{1}{R_6} \\
G_b &= \frac{1}{R_1} - \frac{1}{R_6} \\
G_c &= \frac{1}{R_1} + \frac{1}{R_4} \\
E_1 &= \frac{R_3}{R_2 + R_3} E_{sat} \\
E_2 &= \frac{R_6}{R_4 + R_6} E_{sat}
\end{aligned}
\tag{50}
$$

$$
L = \frac{R_1 R_3 R_4 C}{R_2} \tag{51}
$$

$$
i_L = \frac{v_2 - v_z}{R_1} \tag{52}
$$

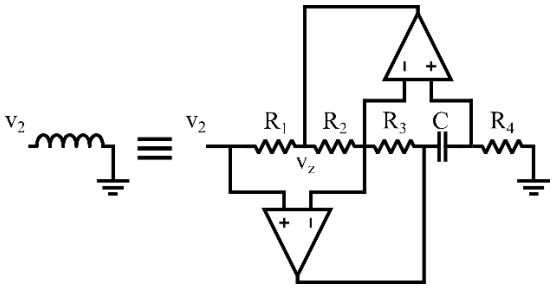
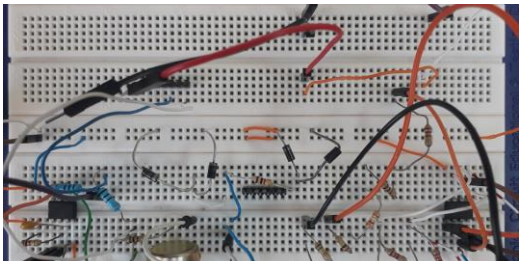
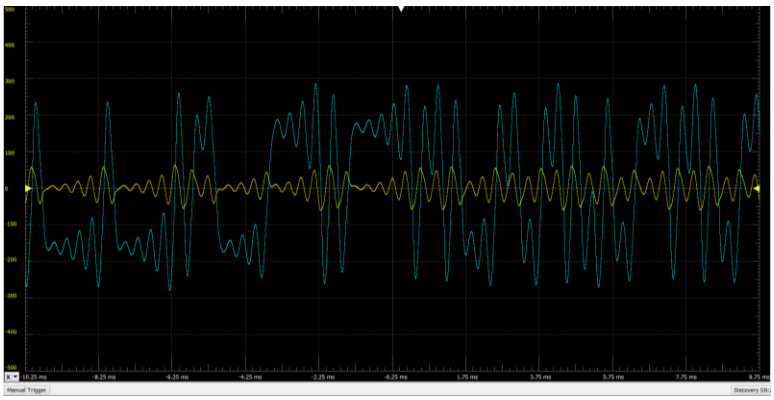


Fig. 6-6. Circuit that emulates an inductor with an inductance that depends on R_1, R_2, R_3, R_4 and C. Inductance current is indirectly measured by sensing voltage across R_1.

We physically built a Chua's circuit (**Fig. 6-7a**) following the above mentioned guidelines and the same parameters used in simulations. **Fig. 6-7b** shows the v_1 and v_2 signals that this circuit generates, which prove the double-scroll attractor behavior of the circuit. This implementation served as a verification for our simulations, since observations found in the built circuit were taken into account for fixing the parameters used in the simulated Chua's circuit.



(a)



(b)

Fig. 6-7. Physical implementation of a Chua's circuit. (a) Circuit in the protoboard. (b) capture of v_1 and v_2 signals.

6.2.4 Chaotic Synchronization

It is characteristic in such systems that the slightest variation of the initial conditions causes two identical chaotic systems to show trajectories that rapidly evolve independently. Therefore, chaotic synchronization techniques are mandatory when it is required for two or more chaotic circuits to operate in an identical manner. Chaotic synchronization is the process where each chaotic system adjusts some

property of their motion to a common behavior owing to a coupling force. There are different types of synchronization techniques, divided into two main alternatives: *one-way* or unidirectional, and *two-way* or bidirectional.

In a one-way synchronization, a system *1* receives as input the internal state of another system *2*. Two alternatives are usually mentioned for one-way coupling: those are *complete replacement* and *master-slave*. Complete replacement consists in the replacement of at least one state variable of system *1* using another from system *2*. On the other hand, in master-slave synchronization the feeding of the state variable is not that direct; instead, the difference between two internal state variables is fed into the slave system. However, in the two-way synchronization both coupled systems interact by mutually exchanging their internal states and achieving a common trajectory.

The details on the stability of the synchronization process are not mentioned as such analysis is out of the scope of this Chapter. However, the chaotic synchronization is evaluated qualitatively by means of the evolution of the system, but also quantitatively by using two metrics; the synchronization error and the synchronization time [120]. Synchronization error $e(t)$ is calculated using eq. (53), where N stands for the number of systems and s is a vector that contains the state variables of the i-th system (in the specific case of the Chua's circuit, $s = (x, y, i_L)^T$). Synchronization time t_s is defined as the instant such that $e(t) < e_{lim}$ for all $t > t_s$, being e_{lim} an upper bound for error close to zero. For all the simulations carried out in the context of this Chapter, e_{lim} was set at 0.01 V.

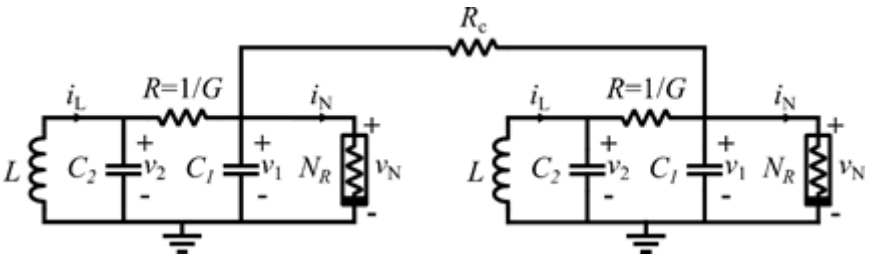
$$e(t) = \sqrt{\frac{1}{N(N-1)} \sum_{i,j} \left\| s_i - s_j \right\|^2} \tag{53}$$

6.2.5 Two-way synchronization of Chua's circuits

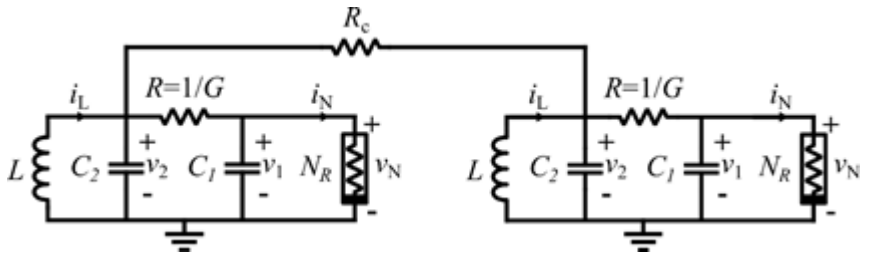
Synchronization of Chua's circuits can be done both in one-way and two-way. This Section is focused on the two-way synchronization, following the work in [11]. **Fig. 6-8** shows two alternatives of Chua's circuits bidirectional synchronization using a single coupling resistor R_c. While in **Fig. 6-8a** the coupling is done via v_1, in **Fig. 6-8b** circuits are coupled through v_2. From now on, each Chua's circuit will be called a "system" and will be numbered adequately, and all their internal state variables will include the system's number as a subscript.

The synchronization of coupling systems is analyzed in terms of the synchronization stability. The state equations of the coupled Chua's circuits, now modified by the added coupling part, is the starting point of the analysis. By applying Kirchhoff circuit laws, the state equations of each circuit are obtained and are shown in eqs. (54) and (55) for the schemes in **Fig. 6-8a** and **Fig. 6-8b**, respectively. By comparing them with the state equations of a Chua's circuit in (48), we observe that a summation term has appeared in the internal dynamics where $k_x = \alpha_R/R_c$ and $k_y = R/R_c$ are called *coupling strength*. Indeed, the coupling amplifies the error between the coupled state variables as a feedback information given to each system, attempting to synchronize by minimizing the error. **Fig. 6-9** synthesizes the whole synchronization process in a scheme that highlights the feedback loop. A higher coupling strength guarantees a successful

synchronization, whereas the opposite may end in erratic behaviors. Mathematical justification can be found using Master-Stability Function theory [121] and according to [122], synchronization occurs when $k_x > 6.274/2$ or $k_y > 1.184/2$ for two units. Either way, we will show these conclusions through electronic circuit simulations.



(a)



(b)

Fig. 6-8. Two-way synchronization schemes for Chua's circuits, were R_c coupling can be located in (a) node v_1 or (b) node v_2.

$$\begin{aligned} x_1 &= \alpha[y_1 - x_1 - f(x_1)] + k_x(x_2 - x_1) \\ y_1 &= x_1 - y_1 + z_1 \\ z_1 &= -\beta y_1 \\ x_2 &= \alpha[y_2 - x_2 - f(x_2)] + k_x(x_1 - x_2) \\ y_2 &= x_2 - y_2 + z_2 \\ z_2 &= -\beta y_2 \end{aligned} \tag{54}$$

$$\begin{aligned} x_1 &= \alpha[y_1 - x_1 - f(x_1)] \\ y_1 &= x_1 - y_1 + z_1 + k_y(y_2 - y_1) \\ z_1 &= -\beta y_1 \\ x_2 &= \alpha[y_2 - x_2 - f(x_2)] \\ y_2 &= x_2 - y_2 + z_2 + k_y(y_1 - y_2) \\ z_2 &= -\beta y_2 \end{aligned} \tag{55}$$

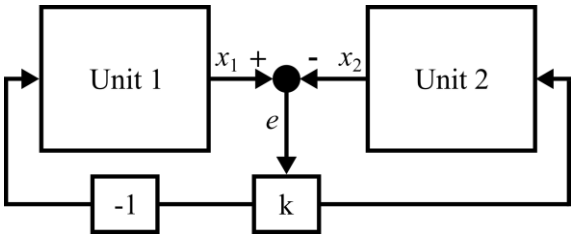


Fig. 6-9. Equivalent scheme for the bidirectional synchronization. Both systems output is the state variable x (or v_1). Then, the difference between these outputs e is amplified by factor k and fed back to each system to contribute to the internal dynamics.

Circuit simulations of the scheme shown in **Fig. 6-8a** were carried out using LTSpice software in order to illustrate the synchronization stability. Synchronization stability depends on the coupling strength k_x, which in turn depends on α, R and R_c. Parameters α and R can't be easily modified, as the circuit may easily lose the double-scroll attractor behavior. Hence, only R_c is modified to tune k_x. By doing so, three different behaviors are observed. More specifically, when R_c is high (100 kΩ in **Fig. 6-10a**), all Chua's circuits show independent trajectories due to the poor coupling strength. Thus, the chaotic systems are not synchronizing. The synchronization error also confirms this; there is no trend towards zero observed during simulations. A second kind of behavior is found while decreasing R_c (R_c = 25 kΩ in **Fig. 6-10b**). The temporal evolution towards null synchronization error confirms that both systems are finally synchronized. However, the chaotic behavior of the Chua's circuits is no longer exhibited. Instead, both systems start to oscillate periodically within the operational amplifiers output voltage swing, clearly limited by their saturation voltages. Certainly, it is an unstable response and an undesirable scenario that applications based on chaotic synchronization must avoid. Finally, for R_c selected sufficiently low (5 kΩ in **Fig. 6-10c**), the Chua's circuits trajectories are rapidly synchronized (**Fig. 6-10c**). The synchronization error is asymptotically reaching zero, which is a sign of a complete chaotic synchronization process.

After testing the circuit scheme for different R_c values, the boundaries of the behaviors previously found in terms of R_c value were explored. The results are shown in **Fig. 6-10d**: chaotic synchronization is guaranteed for R_c < 5.5 kΩ, whereas a periodic oscillation synchronization is experienced in the range R_c ⊂ [5.5, 100] kΩ and no synchronization at all is produced when R_c > 100 kΩ. In the chaotic synchronization region, the synchronization time is also shown. It can be observed that it decreases exponentially as R_c decreases (or as the coupling strength increases). In conclusion, a higher coupling strength guarantees a successful synchronization, whereas the opposite may end up in erroneous behavior or no synchronization at all.

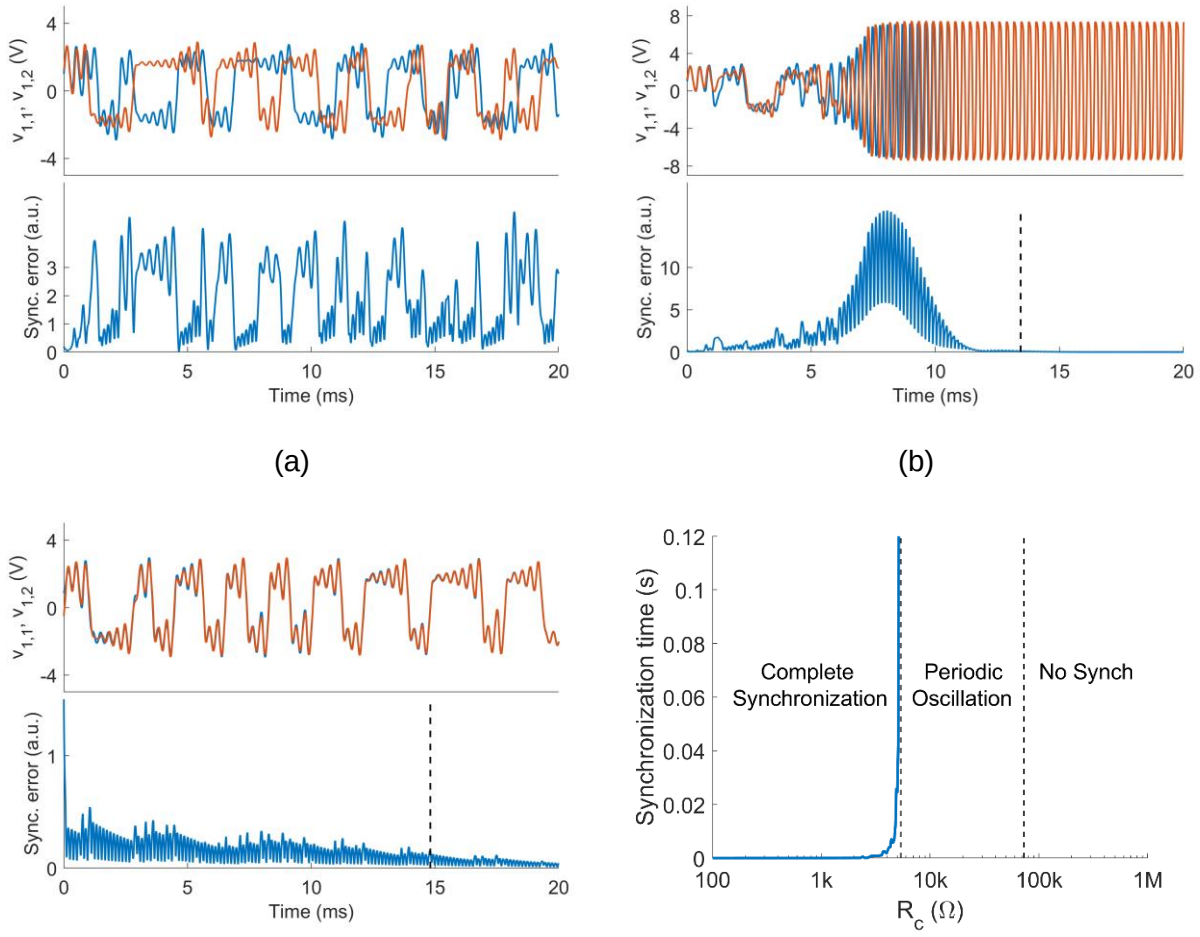


Fig. 6-10. Temporal evolution of voltages at v_1 and v_2 nodes and synchronization error in a bidirectional synchronization scheme of **Fig. 6-8a** for three different R_c values: (a) 100 k**Ω**, (b) 25 k**Ω** and (c) 5 k**Ω**. A vertical black dashed line marks the instant when both systems reach synchronization. In (d), different behaviors observed and synchronization times for the range $R_c \subset$ [100, 1 M] **Ω** are shown.

6.2.6 Adaptive Synchronization of Chaotic Circuits Using Memristors

Adaptive synchronization (concretely "edge-based" adaptive synchronization) provides coupling strength which is not fixed but varies with time [123]. This approach is more realistic, as many real-world networks are characterized by evolving couplings, like in the behavior observed in some biological networks [124]. The main difference with the bidirectional synchronization is that coupling elements react to the divergence of two trajectories, adapting their coupling strength and ultimately achieving a stable synchronization state. Formally, the network of identical systems is described in (56) [123], where $\sigma_{ij}(t)$ stands for the adaptive coupling strength between system *i* and system *j* (with initial value σ_{ij}^{0}), $h(s_i)$ stands for the output function of system *i*, ε_i is the set of coupling links between system *i* and other systems, i.e. $j \in \varepsilon_i$ and if there exists a coupling element between systems *i* and *j*, and *a* is a constant factor.

$$\frac{ds_i}{dt} = f(s_i) - \sum_{j \in \varepsilon_i} \sigma_{ij}(t)\left(h(s_j) - h(s_i)\right)$$
$$\frac{d\sigma_{ij}}{dt} = a\|s_i - s_j\|, \sigma_{ij}(t) = \sigma_{ij}^0 \quad (56)$$

Recent works have shown that memristors can be used successfully to build adaptive coupling elements, such as the one described here. Concretely, [11] proves the usefulness of the linear memristor model in the implementation of bidirectional, adaptive synchronization of Chua's circuits. To meet the requirements of the coupling element in (56), the R_c coupling resistor mentioned earlier was substituted by two memristors in antiparallel association. When two memristors connected in antiparallel are stimulated with enough voltage across them, depending on the voltage polarity, one of them will switch towards LRS whereas the other does the same towards HRS (and vice versa). In either case, one of the two memristors will be in LRS, thus the equivalent resistance of the two will be in the levels of LRS. **Fig. 6-11a** shows a network of two coupled Chua's circuits using such antiparallel association. As mentioned previously, the two Chua's circuits are called systems and labeled with a number, for representation purposes. Each system is only accessible via the node v_1, and all couplings are done through it. When it is necessary to distinguish circuit elements or magnitudes, the number of the system is included as a subscript. For instance, $v_{1,1}$ and $v_{1,2}$ refers to the v_1 magnitude of system 1 and 2, respectively. Because the coupling is being done through v_1, the output function at (56) is the v_1 node, i.e. $h(s_1) = x_1$.

The behavior of the adaptive synchronization in **Fig. 6-11a** is illustrated through simulations results. Because our interest is to use the Stanford/PKU model as a realistic memristor model in simulations, but [11] proved the memristor-based adaptive synchronization using linear memristor model [5], a simulation with each model was performed for comparison purposes. The simulation setup can be described as follows; both systems were initialized using different initial conditions: $v_{1,1}(0) = 1$ V, $v_{1,2}(0) = 1.01$ V, $v_{2,1}(0) = v_{2,2}(0) = 0$ V, $i_{L,1}(0) = i_{L,2}(0) = 0$ A. Stanford/PKU model keeps its default parameters listed in **Table 2-2** in Chapter 2, but because it works at high switching times, the Chua's circuit was adapted to this speed by setting $C_1 = 100$ fF and $C_2 = 1$ pF. At the same time, the linear memristor model was also adapted to work at such high frequency by changing its parameters as follows: $R_{ON} = 200\Omega$, $R_{OFF} = 20K\Omega$, $\mu_v = 200e\text{-}9$ m^2/sV, $D = 10$ nm, and $p = 2$. Memristors are both initialized at R_{OFF}; otherwise at LRS the coupling strength could be high enough to provoke a direct synchronization, and the adaptive evolution would not be observed.

First, the results concerning the linear memristor model are shown in **Fig. 11b**. At the top, the temporal evolution of $v_{1,1}$ and $v_{1,2}$ show that systems are effectively synchronizing. In the middle, the synchronization error also reveals the same information. But the most interesting part about the adaptive synchronization lies in observing the evolution of the internal states of the memristors. At the bottom, the integral of the currents passing through m12 and m21 are shown. We use the time integral of the current rather than simply the device resistance as it captures in a more illustrative way the overall contribution of every memristor during the synchronization process. The integrals start at low values (due to the HRS in

memristors). However, a high synchronization error means a higher voltage drop across the memristors, which triggers the SET process. Afterwards, the coupling strength increases and the error decreases until synchronization is complete. Both memristors contribute evenly as both integrals of currents increase. When using the Stanford/PKU model (see **Fig. 6-11c**), the outcome is also a complete synchronization between the two systems. However, the whole picture is not identical: in fact, only one memristor contributes to the synchronization. Furthermore, the error increase observed in **Fig. 6-11c** is higher than what shown in **Fig. 6-11b**, and the resistance switching is more abrupt. We attribute this difference in the fact that, while the switching speed of the devices when using the linear memristor model is proportional to the current flowing through the memristors, the Stanford/PKU model exhibits a fundamentally different behavior which depends on the applied voltage and also on the existence of switching thresholds.

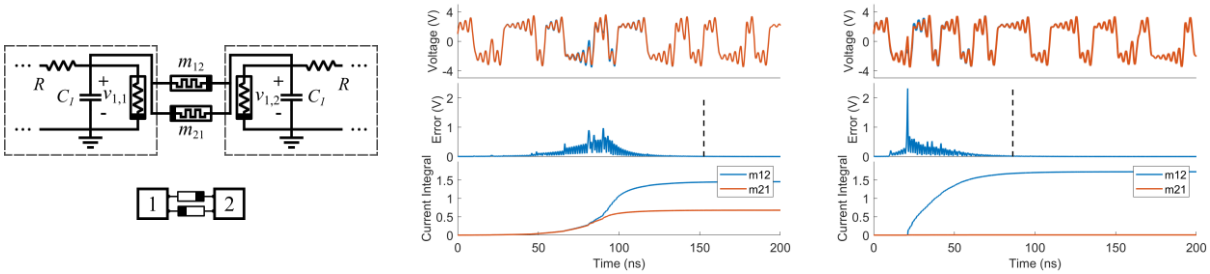
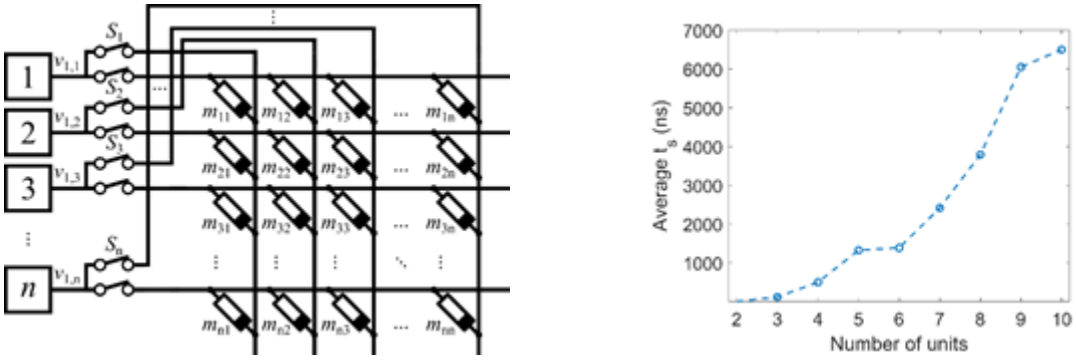


Fig. 6-11. Synchronization of two Chua's circuits using an antiparallel memristor association. (a) Circuit schematic and simplified scheme for Chua's circuit systems and memristors. Simulation results in (b) concern the linear memristor model with parameters R_{ON} = 200 Ω, R_{OFF} = 20 KΩ, μ_v = 200e-9 m^2/sV, D = 10 nm, and p = 2 in Biolek window function, and in (c) the Stanford/PKU model with the default parameter values. The plots show the v_1 evolution in every circuit, along with synchronization error (the synchronization time is marked with vertical dashed line), and the time integral of current through each memristor.

Finally, it is worth noting that crossbar structures can be used as a topology for coupling systems [115]. In such a case, N systems are connected to each other through antiparallel memristors by means of a N x N crossbar if the v_1 node of every circuit is tied to a row and a column of the array; each row and column must be used only once. **Fig. 6-12a** shows an example where the i-th system v_1 node is connected to the i-th row and i-th column. While the memristors placed in the diagonal are not functional (both of their terminals are short circuited), the other memristors are connected in antiparallel pairs. For instance, system 1 $v_{1,1}$ node is connected to the top terminals of memristors in row 1 and to the bottom terminals of memristors in column 1. The notation in memristors also helps to identify the antiparallel pairs: m_{12} is the memristor whose top and bottom terminals are attached to system 1 and 2, respectively, whereas the top and bottom terminals of m_{21} are connected to system 2 and 1, respectively. Switches S_1 to S_n allow or block some systems from being synchronized, if this is required in the context of the application of interest.

The scalability of networks is an interesting topic that is addressed in subsequent Sections, especially for the synchronization stability and the synchronization time. The synchronization time may depend strongly on the initial conditions of the circuit. Hence, for a detailed quantization of the synchronization time, instead of performing a single simulation using some fixed initial conditions, a Monte

Carlo method was applied by performing several simulations, each time assigning to each system $v_1(0)$ values selected arbitrarily between -1V and 1V, thus guaranteeing that the voltage drop on the functional memristor will have both polarities and varying amplitudes. Initial values $v_{2,i}(0)$ and $i_{L,i}(0)$ are kept at 0. Then, the synchronization times obtained by this method are averaged. **Fig. 6-12b** shows the average synchronization times achieved using a *n*x*n* crossbar topology for an increasing number of interconnected systems *n*. Indeed, synchronization is stable for all cases and the trend of the average synchronization time is to steadily increase with *n*.



(a) (b)

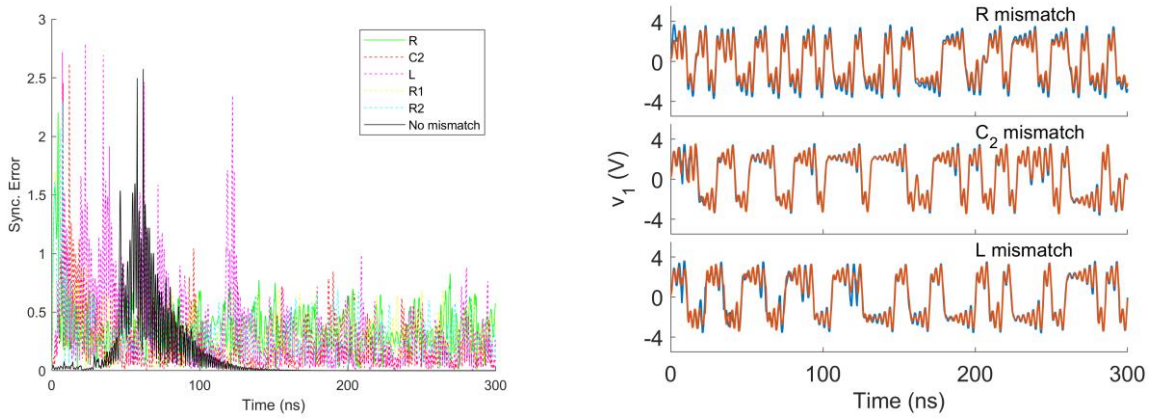
Fig. 6-12. Adaptive synchronization of systems using a crossbar array as interconnect medium. (a) Scheme of the coupling architecture. Each system is connected to a row and a column and switches S_1 … S_n may define which systems are to be synchronized. (b) Average synchronization time using a *n*x*n* crossbar topology and all *n* systems synchronizing, evaluated for 200 simulations using the Monte Carlo method when selecting initial conditions for $v_1(0)$ magnitudes randomly from -1V to 1V.

6.2.7 Impact of Mismatches Between Chua's Circuits

Up to this point, complete synchronization has been proved for Chua's circuits and memristors as coupling elements, always under the assumption of identical systems. Although it is not the main topic of this Thesis, a reasonable question may arise to the reader about the networks discussed so far: how do these results change when the systems are not matching between them exactly? Here, we evaluate the consequences of mismatches between the different components in the Chua's circuits, while the impact of memristor faults is covered in the next Section. As an answer to the aforementioned question, the scheme of **Fig. 6-11a** is simulated again while forcing mismatches between the passive components of both systems, one component at a time to gauge the impact of each fault separately. Concretely, we assumed that in each mismatched pair one element has 10% higher value than the nominal value whereas the other has 10% lower value. For instance, instead of using the element C_2 with nominal value of 1pF, system 1 has $C_{1,1}$ = 0.9 pF and system 2 $C_{1,2}$ = 1 pF, keeping the other circuit elements at their nominal values.

Fig. 6-13a shows the temporal evolution of voltage v_1 in both systems for the mismatch scenarios for R, C_2, L, R_1 and R_2, which are the ones where we observed the highest synchronization error. Additionally, the synchronization error of the case without mismatches has been added for comparison purposes, where we observed a rising error followed by a sudden steady decrease towards zero asymptotically. Unlike that, when mismatches are present, the synchronization error experiences one or more rising peaks and a decreasing trend after that, but without reaching to zero error. The higher peaks

mean that switching events take place in the memristors. In all cases, the synchronization error is almost never below the threshold defined previously. Nevertheless, the temporal evolution of v1 nodes for both systems, shown in **Fig. 6-13b**, clearly demonstrates a good degree of synchronization, even if their magnitudes are not as close as expected. Hence, in spite of not being in a complete synchronization state, all temporal simulations lead to a satisfying synchronization regime, meaning that in the unlucky event of having extreme device mismatches, as the ones assumed here, synchronization will still be possible.



(a) (b)

Fig. 6-13. (a) Temporal evolution of the synchronization error for the most relevant mismatch cases we observed in a network of two Chua´s circuits, along with the case of no mismatch for comparison purposes. (b) Temporal evolution of $v_{1,1}$ and $v_{1,2}$ nodes in *R*, C_2 or *L* mismatch scenarios.

6.3 Study of Fault Tolerant Properties Exploration of the Crossbar Coupling Medium

This section presents a collection of different worst-case scenarios in which it is assumed that memristor faults are present in the coupling medium. They are mainly studied in terms of synchronization stability and synchronization time. Ultimately, the results are compared with a non-faulty ideal crossbar to explore the fault tolerant properties of the crossbar topology being used as a coupling interconnect medium. The two types of memristor faults considered in this Section -faults and transient faults- were already covered in Chapter 3.

6.3.1 Presence of Permanent Memristor SAOFF Faults

In the first scenario of interest, a coupling crossbar topology with permanent memristor faults is considered. Since SAON faults correspond to the LRS state, in the context of this study, any coupling memristor found in LRS would guarantee synchronization of the chaotic systems it connects. The same applies for bridge defects that short-circuit two systems. Therefore, the SAOFF faults are far more interesting in this context because the devices are permanently in HRS, so they will not contribute to synchronization. Therefore, synchronization will depend on the contribution of the rest of existing memristors (redundancy) in the interconnect medium.

Note that we are only considering SAOFF faults caused by memristor malfunction, and not by other defects. Therefore, we evaluate the synchronization performance in the presence of SAOFF faults while modeling the faulty memristors in Spectre software as 1-MΩ resistors, i.e. a device with a high resistance equivalent to the memristor HRS. Even though this replacement is a rough simplification of the memristor model, we confirmed that replacing a faulty memristor with a simple resistor rather than modifying the state-change mechanism in the Stanford/PKU memristor model, practically led to identical simulation results. First, we focus on a network of just two systems, as shown in **Fig. 6-8a**, where there are only two memristors in an antiparallel configuration. An SAOFF fault is applied to one of the memristors, e.g. m_{21} (this is equivalent to assuming an SAOFF fault in m_{12}). We then evaluate the performance by means of the synchronization time. We carried out Monte Carlo simulations, in order to select random initial values as explained earlier, for 200 samples collecting results for the synchronization time. This time the average is not calculated, but all synchronization times obtained are displayed in **Fig. 6-14** in a histogram. In the same figure we show the results when considering two functional memristors. We observe that synchronization is achieved in both cases, so the antiparallel memristor association is not a strict requirement for coupling. However, when only one memristor is functional, synchronization could take much longer to be achieved.

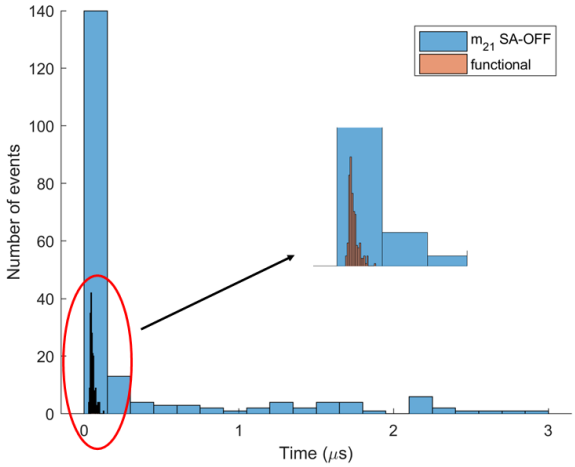
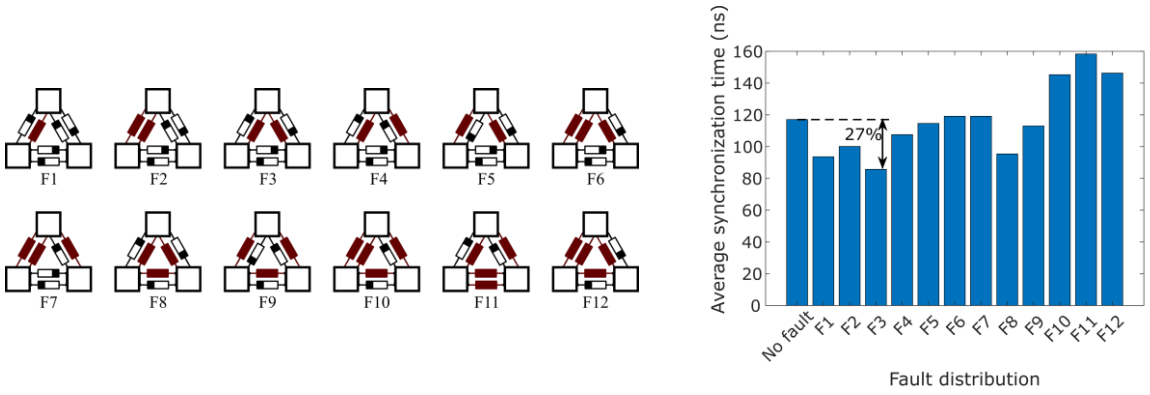


Fig. 6-14. Monte Carlo simulation results for the synchronization time of two network, considering 200 samples with random v_1 initial conditions. The red/dense histogram corresponds to having 2 functional memristors, whereas the blue/sparse histogram corresponds to having m_{21} in SAOFF fault.

Next, we extend this study to a network of three Chua´s circuits which we interconnect by means of a 3×3 crossbar array, following the general scheme proposed in [115] and shown in **Fig. 6-12** for an arbitrary number of circuits (a $n \times n$ array is required to connect up to n chaotic systems). We perform an exhaustive analysis of all possible SAOFF fault distributions among the nine memristors of the crossbar, while always assuming at least two remaining working devices. There are 46 interconnection schemes that keep all chaotic systems interconnected via at least one functional memristor, which can be grouped into 12 families labelled as F1, F2 … until F12 in **Fig. 6-15a**. For instance, the family F1 has only one SAOFF memristor, whereas F2 to F5 have two, F6 to F9 have three, and F10 to F12 have four SAOFF memristors,

respectively. Five or more faulty memristor distributions were not considered as we need at least two functional memristive links to synchronize 3 systems.

200 Monte Carlo simulations were performed for every such family, as well as for the fault-free crossbar case, while using the same configuration and initial conditions that were again normally distributed for the three chaotic systems. **Fig. 6-15b** depicts the obtained average synchronization time where we generally observe that some interconnect families perform better than the rest. Most importantly, we notice that families with one, two or three SAOFF faulty memristors behave similarly or even better than the fault-free crossbar case, with the best results corresponding to family F3. The worst results were observed for families that have just two functional memristors, where the synchronization process lasted much longer. The latter highlights the requirement for a minimum number of functional memristors in the coupling interconnect medium.



(a) (b)

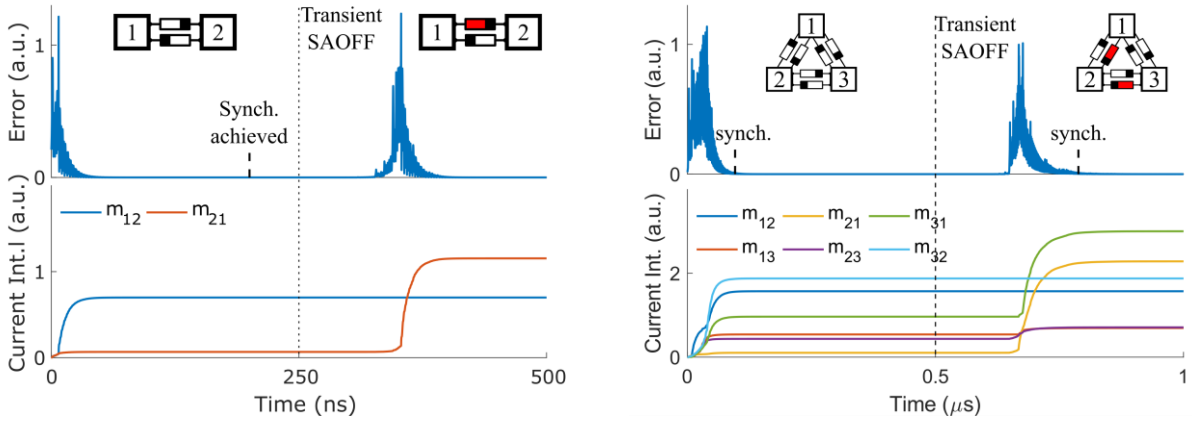
Fig. 6-15. (a) All possible distributions of interest for the SA-OFF faults in antiparallel memristors interconnecting three Chua's circuits, represented by white blocks. Faulty memristors are marked in red color. (b) Average synchronization time considering 200 samples of Monte Carlo simulations for each one of the fault distributions in (a).

6.3.2 Presence of Transient Memristor SAOFF Faults

This scenario explores the evolution of a network after the systems are synchronized if the memristors involved in the synchronization process are suddenly affected by a transient fault. We say a memristor was involved in a synchronization process when it switched significantly towards LRS, while the other memristors that were not involved were kept at HRS or their state was only slightly modified. The interest is to check if the network is able to cope with transient faults and maintain its coupling characteristics or, conversely, synchronization is lost; thus it is a qualitative test of the synchronization stability.

The test has been performed in networks of 2 and 3 systems interconnected using a crossbar topology, as shown in **Fig. 6-16a** and **Fig. 6-16b**, respectively. The simulations follow a common procedure. First, synchronization process occurs as usual, using functional memristors. Some memristors experience a larger decrease in their resistive state. In **Fig. 6-16a**, we referring to m_{12}, whereas in **Fig. 6-16b** the memristors contributing the most are m_{12} and m_{32}. After waiting for enough time so as to guarantee a complete synchronization, those memristors suffer a transient SAOFF failure (marked at 200 ns and 500 ns).

Although the coupling was strong enough to force a common trajectory for each system, the fault provokes an insufficient coupling strength and the systems eventually lose their initial synchronization. We observe that after this moment, the error starts growing again until other memristors finally switch (in the 2-system network it is m_{21}, while in the 3-system networks are mainly m_{31} and m_{21}), thus achieving the synchronization again. Such results confirm that synchronization can be lost due to faults occurring to the coupling elements on-line. Nevertheless, owing to the interconnect medium being fault-tolerant, the rest of the functional memristors contribute to overcome this situation and synchronize again.

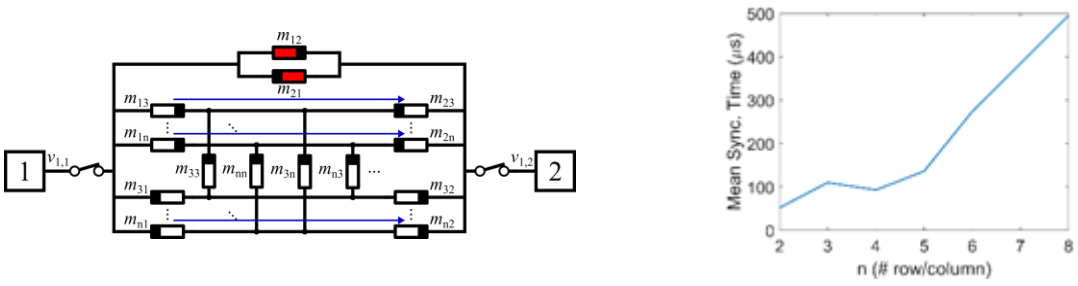


(a) (b)

Fig. 6-16. Temporal evolution of the synchronization error and current-time integral in a scenario of memristor transient SAOFF faults after a synchronization process, in a network coupled via a crossbar topology. The faults occur in memristors that have contributed to the synchronization process. Example for a (a) 2-system network and (b) 3-system network.

6.3.3 Evaluation of Coupling when Connecting Fewer Systems to a Crossbar

We explored further the redundancy of the crossbar topology simulating scenarios where $n \times n$ array was used to interconnect a number of chaotic systems smaller than n, leaving the rest of columns and rows as floating. In fact, we focused on just two chaotic systems connected to a $n \times n$ crossbar array with $n > 2$, and initialized the two antiparallel memristors that directly connect the two systems at SAOFF. This way we aimed to explore the role of every redundant cross-point device in the synchronization process. **Fig. 6-17a** shows the equivalent circuit of the crossbar array highlighting the two SAOFF memristors and the rest of main conducting paths generally observed in the simulation results. Most importantly, the synchronization was successful in all Monte Carlo simulations for crossbar size ranging from 3×3 to 8×8. It can be concluded that the current flowing through the rest of the cross-point devices eventually contributes so that the two systems synchronize. However, by observing **Fig. 6-17b** we note that the mean synchronization time tends to increase with the size of the crossbar array; similar to what was observed previously in **Fig. 6-12b** for the mean synchronization time with a fully-functional $n \times n$ crossbar connecting n systems. So far we concluded that the passive memristive crossbar array is a redundant and fault-tolerant interconnect medium that guarantees synchronization, but synchronization time increases significantly with the size of the array. However, the synchronization time was significantly improved while emulating SAOFF faults in the crossbar array. Generally speaking, assuming that a memristor is stuck at HRS is equivalent to assuming it is not present in the interconnect medium since in either case it does not contribute to the synchronization.



(a) (b)

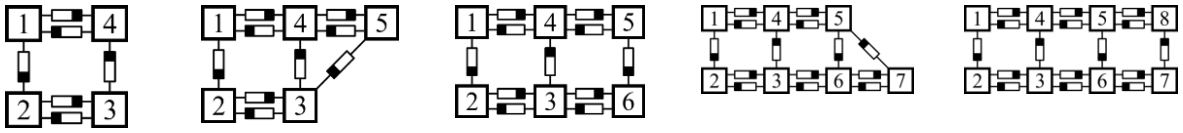
Fig. 6-17. (a) Equivalent circuit schematic for an $n \times n$ crossbar array where two systems are connected, showing the faulty memristors and highlighting the main alternative current paths observed in simulations. (b) Simulation results for the evolution of the average synchronization time of two chaotic systems connected to a $n \times n$ crossbar, while varying n.

6.4 Exploration of Optimized Interconnect Topologies for Synchronization

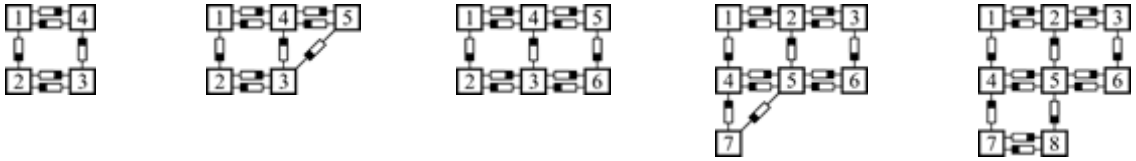
In Section 6.3.1, the simulation results for a 3-system network in **Fig. 6-15b** show evidence that some interconnection patterns that use less memristors seem to perform better than the full crossbar array in terms of synchronization time. Therefore, in this Section we build upon such results towards finding an optimized and scalable memristor-based interconnect medium for the fastest possible adaptive synchronization of an arbitrary number of chaotic circuits. Several interconnect patterns could be considered for this purpose and the memristors that were assumed SA-OFF in **Fig. 6-15a** are now simply removed from the network.

The proposed interconnect approach is described in **Fig. 6-18**. We show five different scaling examples for a network of an increasing number of chaotic systems. The topologies 3, 4 and 5 are practically inspired by the original form of the aforementioned F3 family. Every additional chaotic system is connected to the rest in the network in such a way that the main properties of connectivity observed in F3 are locally maintained. On the other hand, the topologies 1 and 2 are alternatives in which the connected systems form a square-like grid. We explored both 1-D and 2-D scaling strategies, resulting in a more "stretched" or a more "compact" grid, respectively. Finally, the topology 5 is an extended version of topology 3 with better symmetry in the interconnection of the chaotic systems, located at the two extreme positions of the chain.

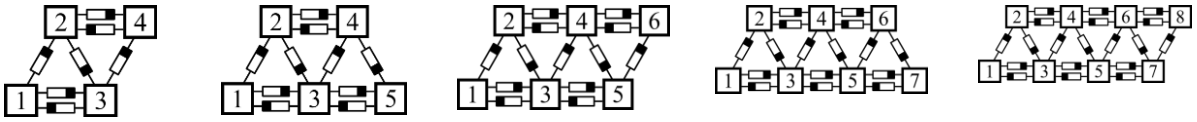
Topology 1: 1-D Square



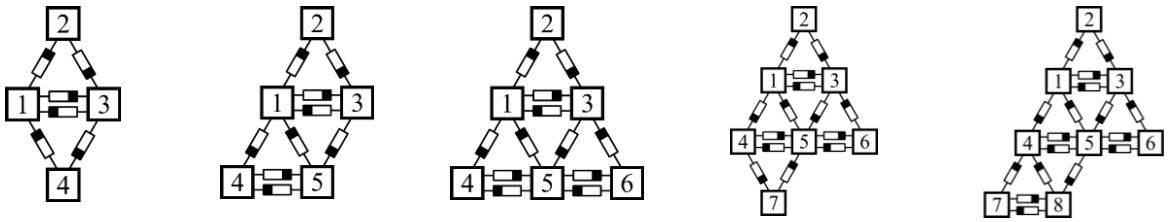
Topology 2: 2-D Square



Topology 3: 1-D triangular



Topology 4: 2-D triangular



Topology 5: 1-D triangular extended

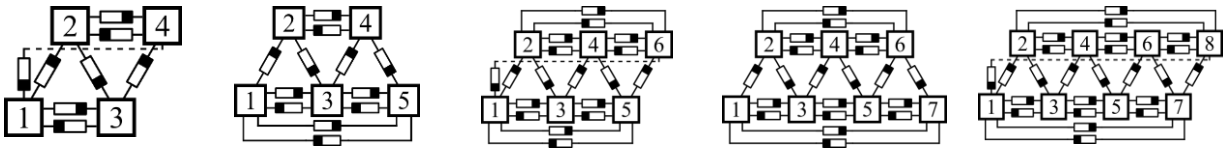
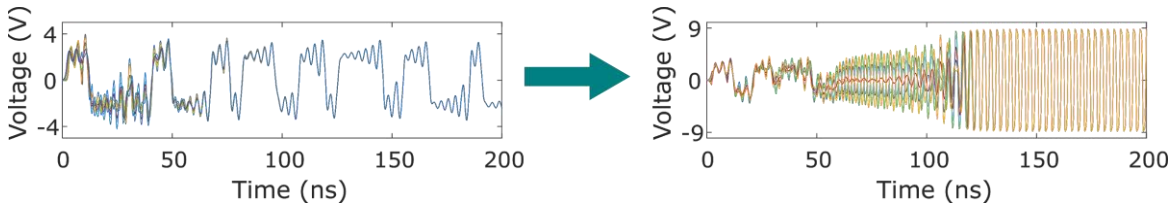


Fig. 6-18. Alternative interconnect topologies and scaling examples for an increasing number of chaotic systems (represented as numbered squares).

We compared the five topologies in transient simulations for an increasing number of interconnected systems. Overall, while scaling the grids to connect up to 10 systems we noticed that from some point the chaotic behavior was lost and an undesirable periodic oscillation was observed, as shown in **Fig. 6-19a**. We resumed our findings on the observed behavior in **Fig. 6-19b**. The topology 5 outperformed the rest, supporting up to 13 connected systems before the chaotic behavior was lost. On the other hand, we did not observe any such problem with the fully-functional crossbar topology, which could be scaled up to support more systems than the rest of the examined topologies. In fact, we did not observe any upper bound for that in all our circuit simulations for up to 20 systems. Therefore, the maximum number of interconnected systems is a restriction to consider while using the proposed topologies.

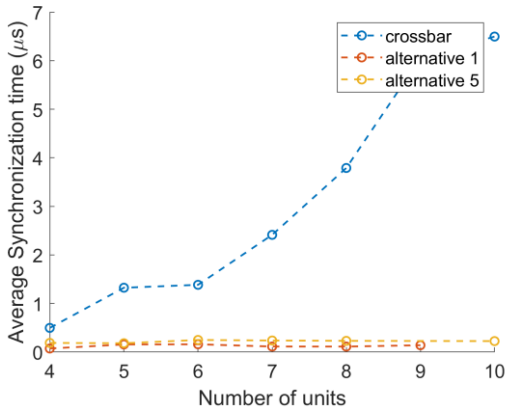


(a)

	Crossbar Topology	Alternative Topologies				
		1	2	3	4	5
4	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✓
10	✓	✗	✗	✗	✗	✓
11	✓	✗	✗	✗	✗	✓
12	✓	✗	✗	✗	✗	✓
13	✓	✗	✗	✗	✗	✓
14	✓	✗	✗	✗	✗	✗
...	✓	✗	✗	✗	✗	✗

✓	Chaotic Synchronization
✗	Periodic Oscillation

(b)



(c)

Fig. 6-19. (a) An example of the temporal evolution of v_1 in every circuit showing chaotic synchronization (left) and a periodic oscillation (right) once an upper scaling limit is exceeded. (b) Summary of findings in temporal simulations using different topologies and a varying number of interconnected chaotic systems. (c) Simulation results for the average synchronization time while using alternative topologies 1 and 5, for a varying number of systems *n*. The plot includes the curve corresponding to a full crossbar shown in **Fig. 6-12b**.

Moreover, from 200 Monte Carlo simulations we computed the average synchronization time for an increasing number of interconnected systems, and the simulation results are shown in **Fig. 6-19c**, where we also plot again the curve corresponding to the full crossbar array, shown previously in **Fig. 6-12b**. Some of the five suggested topologies performed better (e.g. in a 9-system network the topology 1 had 140 ns and the topology 5 had 225 ns average synchronization time), but the differences among them are negligible compared to the performance of the full-crossbar (6.05 μs). We show simulation results only for the topologies 1 and 5 to keep the plot more readable. As a first conclusion, we distinguish the significant improvement in the synchronization time compared to what was achieved using the crossbar architecture. Additionally, the required synchronization time seems to be independent of the number of interconnected systems, so time savings are higher for higher numbers of interconnected chaotic systems. After observing the resistance evolution of all memristors in the alternative topologies, we attribute such increase in

performance to the existence of what seems as “switching avalanches”; i.e. to the fact that certain SET events provoke other subsequent SET events in a very short time, a behavior not observed in the fully functional crossbar. Moreover, the five alternative topologies generally require a smaller number of memristors compared to the full-crossbar array, thus savings in circuit area are important. Nevertheless, for the same reason, the lower the number of memristors, the lower the redundancy and the fault-tolerance of the system. For instance, **Fig. 6-20a**, **Fig. 6-20b** and **Fig. 6-20c** shows the average contribution of all the memristors in the topologies crossbar, 1 and 5, respectively, for seven interconnected systems over 200 synchronization processes. We notice that topology 1 has just a few leading memristors that contribute approx. 2×compared to some of the rest. On the other hand, topology 5 consists of more memristors and we observe a more uniform contribution of all the devices, similar to what we observe when a full crossbar is used. Therefore, better fault tolerance is expected in this topology, but it is still inferior to the fault tolerance offered by a full crossbar.

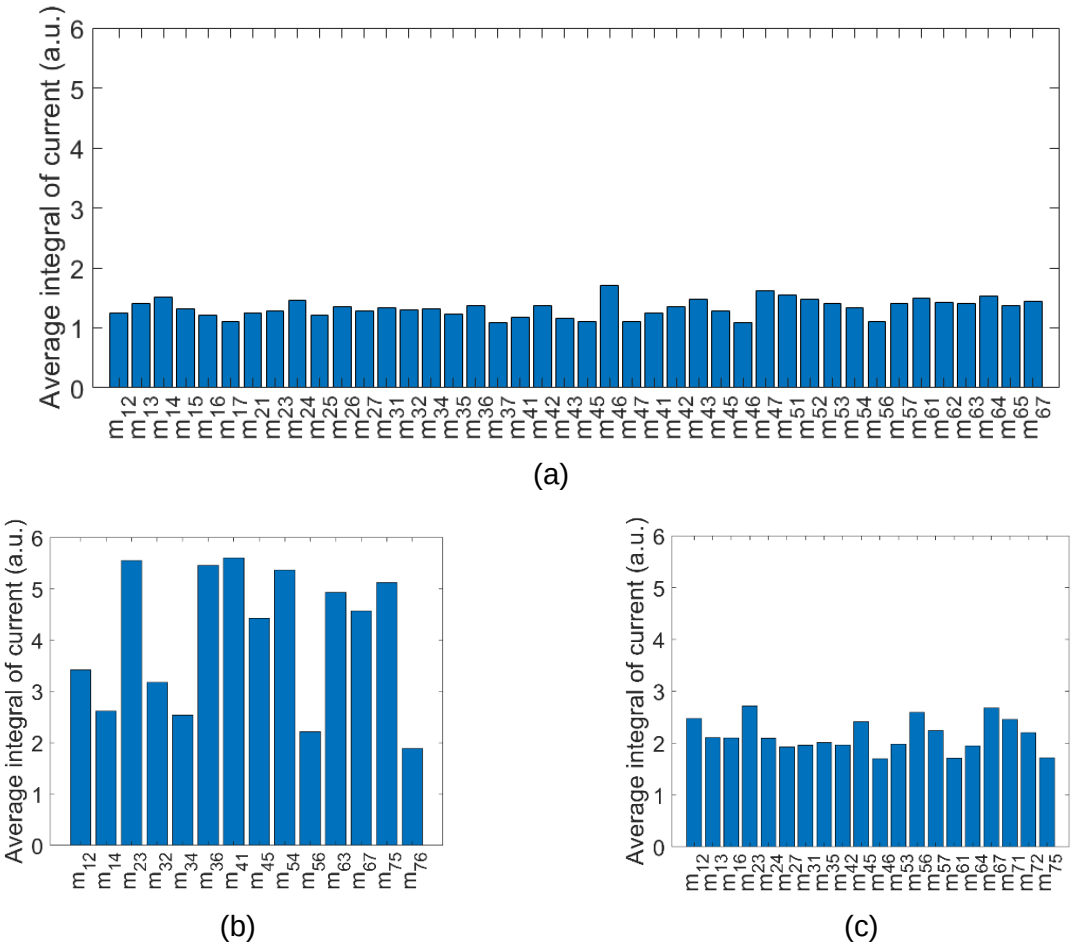


Fig. 6-20. Average of the integral of currents of each memristor in three different network topologies of 7 coupled systems obtained in the Monte Carlo simulations. Memristors contributions in (a) crossbar, (b) alternative 1 and (c) alternative 5 topologies.

6.5 Concluding Remarks

This Chapter moves from the range of applications where memristor states are discretized (covered in Chapters 4 and 5) to the range of analog applications, where the whole resistance range is interpreted as an analog internal state of memristors. In some cases, the memristor was used as a programmable element

for tuning performance of analog circuits (such as amplifiers, filters, oscillators) whereas in other cases its nonlinear switching characteristics were mostly appreciated, such as in the case of neuromorphic computing circuits or chaotic circuits.

Among all the applications reviewed, this Chapter focused on the chaotic synchronization evaluating the performance of the interconnection medium based on memristors. More specifically, the impact of SAOFF memristor faults was studied using realistic, filament-type models instead of simplistic models used in other relevant works of the recent literature. Different scenarios showed that the interconnect medium composed of memristors is tolerant to permanent and transient faults. Furthermore, it was discovered that in some cases the synchronization time could be decreased owing to such types of faults. Because of this finding, different topologies were discovered and we were able to observe important improvements in the required synchronization time. The downside for these novel interconnect topologies was found when scaling them to include more systems for synchronization; In fact, as the topologies get bigger to accommodate more systems, eventually the chaotic behavior of the circuits can be lost.

Chapter 7 Conclusions & Future Work

This Chapter pretends to summarize the main ideas developed in the Thesis and the conclusions reached based on the obtained results. Furthermore, some guidelines about future work addressing some of the topics covered in the Thesis, are discussed.

7.1 Main Conclusions & Contributions

The results presented throughout this Thesis illustrate that memristor variability and defects are very meaningful in memristor-based circuits and systems, and thus they have to be considered. The development of fault/variability-tolerant circuit design, as well as of testing/detecting methods to avoid malfunction, are really valuable. Indeed, another (and even more desirable) scenario concerns the eventual reduction of such undesired issues through the research of better materials and device fabrication. However, this is undeniably costly. As a consequence, in the meantime, smart approaches at device/circuit structure and/or architecture level, are more affordable, even if this comes at expense of increasing circuit complexity or lowering the device integration density. In this Thesis, the results and conclusions are limited to the experience in simulations and the faithful recreation of memristor features in the models used. Experimental verification would provide a further step to prove the validity of all the solutions proposed.

Chapter 1 introduces the Thesis topic contextualizing the reasons that led the industry to put efforts in looking for new technologies like memristors. Memristor promising characteristics, potential applications and their technology challenges are exposed, motivating the fulfillment of this dissertation. The chapter ends with the goals of the Thesis.

Before entering into the main motivation of the Thesis, a clear understanding of the memristor is required. **Chapter 2** does not only provide this explanation, but instead a review on memristors from the point of view of nonlinear circuits theory and the so far known experimental devices. From the circuit theory point of view, the memristor definition has been evolving to accommodate different systems that have similar features, although there is still an on-going discussion about whether the real devices claimed as memristors can actually be called using this term. On the other hand, several resistive switching devices have been identified as memristors as they share multiple common features with the mathematical definitions, although their behavior is attributed to different physical mechanisms. These devices present different specifications; the requirements of an application points towards specific devices that are more adequate.

At this stage of development of memristor devices and related circuits and systems, it is crucial that realistic physics-based models are available for simulation purposes. In this Chapter a reasonable representation of the available memristor models is given. It is concretely interesting the fact that the Stanford/PKU model, which is a physic-based model of memristors verified based on experimental data, served as a solid basis for the conducted simulations throughout this Thesis. Finally, the Chapter concludes with the presentation of crossbar arrays, which are circuit structures that further highlight the benefits of using memristors, enabling the highest possible device integration density.

Chapter 3 delves into the main motivation of this Thesis: the reliability issues in memristor devices. Three main sources of such issues are identified: random behavior (which comes in different forms, as RTN, stochasticity, or state variability), degradation, and faults. The causes and consequences for each of these issues are explained. In this direction, some guidelines about how to include variability into memristor models and model faults are proposed. Combined with memristor model(s), these are altogether the principle tools that are used in the following Chapters to study search for solutions to cope with applications where unreliable memristors are used. In this context, measurements of a real device to illustrate state variability were published in:

- M. Escudero-Lopez, E. Amat, A. Rubio, and P. Pouyan, "An experience with chalcogenide memristors, and implications on memory and computer applications," in *2016 Conference on Design of Circuits and Integrated Systems, DCIS 2016 - Proceedings*, 2017, pp. 249–254.

More specifically, the first block of applications concerns memory systems covered in **Chapter 4**. Memristor is recognized as a low-power (being non-volatile) and high density (owing to the crossbar structures) option for such application. The data density can be further increased with multi-bit codification in the memristor states while using crossbar as the target geometry. However, these such options are limited by the existence of sneak-path currents and variability. Therefore, an objective of the research within this Thesis was to focus on detecting faults in 1T1R-type crossbars. Assuming a set of possible faults, an on-line procedure that tests the entire array is able to successfully detect such faults. The simulation results presented in this Chapter are used to verify the design of the cell, the peripheral circuitry, and specific cases in which a specific fault can interfere with other faults present simultaneously in the same array. The main contributions of this Thesis in the context of this Chapter were published in:

- M. Escudero-Lopez, F. Moll, A. Rubio, and I. Vourkas, "An On-line Test Strategy and Analysis for a 1T1R Crossbar Memory," accepted in 2017 *IEEE Int. Symp. on On-Line Testing and Robust System Design* (*IOLTS*), Thessaloniki, Greece, July 03-05

The second type of applications reviewed in this Thesis concerns digital computing systems, presented in **Chapter 5**. Many different logic design styles have appeared in the recent literature, mainly classified regarding two criteria: crossbar compatibility and whether the logic is stateful or not. When a specific style complies with these two features, it can be implemented in a CIM system, introducing a new paradigm for digital computing architectures, which reduces the latency between computation and memory blocks by allocating both of them in the same physical structures.

In this direction, this Chapter studies the impact of memristor variability in a CIM system for the logic styles CNIMP and MAGIC. Simulation results showed that the yield of logic operations is very sensitive to the amount of variability injected to the involved memristors, even though the yield can be optimized by applying a proper design process. In any case, chaining multiple logic operations reduces the

yield to unacceptable values. Two alternatives can fix the yield: to regenerate the memristors holding the output of a logic operation after it has been produced with a programming step, or use an alternative, variability-tolerant logic style, as a solution for CIM systems. The latter was adopted in this Thesis and a novel, simple, yet robust logic design style was proposed. The simulation results reveal that when this new style is used, the variability cannot affect the yield unless the logic operation holds an extremely large amount of inputs. The main contributions of this Thesis in the context of this Chapter were published in:

- Vourkas, G. Papandroulidakis, M. Escudero-López, G. Ch. Sirakoulis, and A. Rubio, "Variability Challenges in Emerging Memristor-based Logic Circuits," workshop on Memristor Technology, Design, Automation and Computing (*MemTDAC*) affiliated with the *HiPEAC* 2017 conference, Stockholm, Sweden, January 23, 2017
- M. Escudero-Lopez, I. Vourkas, A. Rubio, and F. Moll "Variability-Tolerant Memristor-based Ratioed Logic in Crossbar Array," *IEEE/ACM Int. Symp. on Nanoscale Architectures (NANOARCH 2018)*, Athens, Greece, July 18-19
- M. Escudero-Lopez, I. Vourkas, F. Moll, and A. Rubio, "On the Variability-aware Design of Memristor-based Logic Circuits," *2018 IEEE Int. Conf. Nanotechnology* (*NANO* 2018), Cork, Ireland, July 23-26
- M. Escudero, I. Vourkas, A. Rubio, and F. Moll, "Memristive Logic in Crossbar Memory Arrays: Variability-Aware Design for Higher Reliability", *IEEE Trans. Nanotechnol*., vol. 18, pp. 635-646, 2019

Lastly, **Chapter 6** covers analog applications, where memristor resistance is not discretized in logic levels but the state of the memristor is understood as a tunable parameter. Possible applications range from amplifiers, oscillators and filters that use memristors as a programmable element to tune their properties, to nonlinear elements required in chaotic circuits or for adaptive purposes in neuromorphic computing. In this context, a review of the potential impact of memristors in several aspects of mixed-signal on-chip application tasks, was published in:

- Vourkas, M. Escudero, G. Ch. Sirakoulis, and A. Rubio, "Ubiquitous memristors in multi-level memory, in-memory computing, data converters, clock generation and signal transmission," in: P. Dimistrakis, I. Valov (Eds.) "Metal oxides for non-volatile memory, materials, technology and applications," ***in press***, Elsevier, 2019

From all such possibilities, an adaptive chaotic synchronization application where memristors form an interconnect medium (inside an 1R passive crossbar) between all chaotic circuits to be synchronized, was chosen to further study the impact of SAOFF faults. Through simulation results, the crossbar was found tolerant to faults, both permanent or transient. Moreover, it was found that the synchronization time was

shorter in crossbar arrays with faults rather than in fully functional arrays. Such results led to the further exploration of alternative circuit topologies than the crossbar that improved the synchronization time. The new topologies were indeed contributing to faster synchronization processes. However, while trying to accommodate a larger network of chaotic circuits, an unstable behavior from the chaotic circuits arose, unlike what noticed with the crossbar topology in use. The main contributions of this Thesis in the context of this Chapter were published in:

- M. Escudero, I. Vourkas, and A. Rubio, "Alternative Memristor-Based Topologies for Chaotic Circuit Synchronization," 2019 *Int. Conf. on Memristive Materials, Devices & Systems* (*MEMRISYS*), Dresden, Germany, Jul. 08-11.
- M. Escudero-Lopez, I. Vourkas, and A. Rubio, "Stuck-at-OFF Fault Analysis in Memristor-Based Architecture for Synchronization," 2019 *IEEE Int. Symp. on On-Line Testing and Robust System Design* (*IOLTS*), Rhodes Island, Greece, July 1-3.
- M. Escudero, I. Vourkas, and A. Rubio, "Alternative Memristor-based Interconnect Topologies for Fast Adaptive Synchronization of Chaotic Circuits," *Chaos, Solitons & Fractals*, under review (revise & resubmit), 2019.

Throughout all these cases, variability and faults have shown deviations from the normal behavior of a circuit; usually as an unwanted effect but sometimes enhancing somehow the circuit performance. In any case, there is no doubt that uncertainty in memristors must be taken into account to guarantee reliable functioning in emerging applications of memristors and to lead to the development of effective solutions when such effects provoke malfunction. Indeed, the device models must include faithful behavior of all such phenomena so as to contribute in this direction. The results provided in this Thesis highlight the importance of considering variability and faults in circuit simulations via proper device modeling, revealed possible malfunction in memristor-based circuits as it had never been presented in the relevant literature before, and proposed solutions for testing of memristor-based memory systems, for robust CIM, and for the proper exploitation of memristor variability in analog applications, thus paving the way for a more realistic approach to emerging memristor-based digital and analog applications.

7.2 Future Work

This Thesis stimulated the inclusion of different phenomena related to memristor uncertainty, especially in the very firsts steps of circuit design. Even though several interesting scenarios were studied throughout this Thesis, there are certainly many more that are still to be analyzed.

First, this Thesis covered studies related to variability or faults, but not related to degradation state mechanisms. It could be interesting to include such effects in similar studies to evaluate what level of degradation a circuit can manage before it turns into a real concern.

Chapter 3 presented other faults concerning the row and column nanowires in a crossbar array, as well as the peripheral circuitry. Such faults could be also considered to improve the online testing strategy in emerging memristor-based nonvolatile memory systems. Furthermore, multiple faults could be assumed to eventually study how the algorithm could deal with such cases.

Regarding the proposed ratioed logic design style with memristors, only the NOR logic operation was evaluated, since it was compatible with the crossbar array target topology. In this direction, the feasibility to implement other operations in the crossbar could be further explored. Additionally, in order to have the full picture about the suitability of this logic style for future CIM systems, the synthesis of complex logic functions and their mapping to the memory array could be further studied and compared with the results obtained from other logic style synthesis approaches proposed so far.

Finally, the results obtained from the analysis of the adaptive synchronization of chaotic circuits, were particularly encouraging but still more work could be done. For instance, the quantization of the synchronization time related to the faults found in the crossbar array used as interconnect medium, could help to gain a better understanding of what occurs in such dynamical systems.

References

[1] Gordon E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.

[2] “International Technology Roadmap for Semiconductors,” 2013. [Online]. Available: http://www.itrs2.net/. [Accessed: 24-Feb-2020].

[3] L. O. Chua, “Memristor—The Missing Circuit Element,” *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[4] L. Chua, “Resistance switching memories are memristors,” *Appl. Phys. A Mater. Sci. Process.*, vol. 102, no. 4, pp. 765–783, Mar. 2011.

[5] 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, May 2008.

[6] H. S. P. Wong *et al.*, “Metal-oxide RRAM,” in *Proceedings of the IEEE*, 2012, vol. 100, no. 6, pp. 1951–1970.

[7] H. Li, T. F. Wu, S. Mitra, and H. S. P. Wong, “Resistive RAM-Centric Computing: Design and Modeling Methodology,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 64, no. 9, pp. 2263–2273, Sep. 2017.

[8] 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, Feb. 2010.

[9] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, “Training and operation of an integrated neuromorphic network based on metal-oxide memristors,” *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015.

[10] A. Mazady, M. T. Rahman, D. Forte, and M. Anwar, “Memristor PUF-A security primitive: Theory and experiment,” *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 5, no. 2, pp. 222–229, Jun. 2015.

[11] L. V. Gambuzza, A. Buscarino, L. Fortuna, and M. Frasca, “Memristor-Based Adaptive Coupling for Consensus and Synchronization,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 62, no. 4, pp. 1175–1184, Apr. 2015.

[12] Y. Y. Chen *et al.*, “Understanding of the endurance failure in scaled HfO2-based 1T1R RRAM through vacancy mobility degradation,” in *Technical Digest - International Electron Devices Meeting, IEDM*, 2012.

[13] B. Chen *et al.*, “Physical mechanisms of endurance degradation in TMO-RRAM,” in *Technical Digest - International Electron Devices Meeting, IEDM*, 2011.

[14] C. Liu, M. Hu, J. P. Strachan, and H. H. Li, “Rescuing Memristor-based Neuromorphic Design with High Defects,” in *Proceedings - Design Automation Conference*, 2017, vol. Part 12828.

[15] M. B. Gonzalez, J. M. Rafi, O. Beldarrain, M. Zabala, and F. Campabadal, “Analysis of the switching variability in Ni/HfO2-based RRAM devices,” *IEEE Trans. Device Mater. Reliab.*, vol. 14, no. 2, pp. 769–771, 2014.

[16] T. W. Hickmott, “Low-frequency negative resistance in thin anodic oxide films,” *J. Appl. Phys.*, vol. 33, no. 9, pp. 2669–2682, Sep. 1962.

[17] F. Argall, “Switching phenomena in titanium oxide thin films,” *Solid State Electron.*, vol. 11, no. 5, pp. 535–541, May 1968.

[18] S. Menzel, U. Böttger, M. Wimmer, and M. Salinga, “Physics of the Switching Kinetics in Resistive Memories,” *Adv. Funct. Mater.*, vol. 25, no. 40, pp. 6306–6325, Oct. 2015.

[19] “Knowm.org – Neuro-memristive Artificial Intelligence.” [Online]. Available: https://knowm.org/. [Accessed: 25-Feb-2020].

[20] D. Panda, P. P. Sahu, and T. Y. Tseng, “A Collective Study on Modeling and Simulation of Resistive Random Access Memory,” *Nanoscale Research Letters*, vol. 13, no. 1. Springer New York LLC, pp. 1–48, 10-Jan-2018.

[21] L. Chua, “If it’s pinched it’s a memristor,” *Semicond. Sci. Technol.*, vol. 29, no. 10, p. 104001, Sep. 2014.

[22] L. O. Chua and S. M. Kang, “Memristive Devices and Systems,” *Proc. IEEE*, vol. 64, no. 2, pp. 209–223, Feb. 1976.

[23] B. Mouttet, “Memresistors and non-memristive zero-crossing hysteresis curves,” Jan. 2012.

[24] I. Valov *et al.*, “Nanobatteries in redox-based resistive switches require extension of memristor theory,” *Nat. Commun.*, vol. 4, no. 1, pp. 1–9, Apr. 2013.

[25] Y. V. Pershin and M. Di Ventra, “Memory effects in complex materials and nanoscale systems,” *Adv. Phys.*, vol. 60, no. 2, pp. 145–227, Nov. 2010.

[26] S. D. Ha and S. Ramanathan, “Adaptive oxide electronics: A review,” *J. Appl. Phys.*, vol. 110, no. 7, p. 071101, Oct. 2011.

[27] H. Zhang *et al.*, “Gd-doping effect on performance of HfO2 based resistive switching memory devices using implantation approach,” *Appl. Phys. Lett.*, vol. 98, no. 4, p. 042105, Jan. 2011.

[28] S. J. Song *et al.*, “Real-time identification of the evolution of conducting nano-filaments in TiO2 thin film ReRAM,” *Sci. Rep.*, vol. 3, no. 1, pp. 1–6, Dec. 2013.

[29] C. Xie *et al.*, “High-performance nonvolatile Al/AlOx/CdTe:Sb nanowire memory device,” *Nanotechnology*, vol. 24, no. 35, 2013.

[30] S. Menzel, P. Kaupmann, and R. Waser, “Understanding filamentary growth in electrochemical metallization memory cells using kinetic Monte Carlo simulations,” *Nanoscale*, vol. 7, no. 29, pp. 12673–12681, Aug. 2015.

[31] F. D'Acapito, E. Souchier, P. Noé, P. Blaise, M. Bernard, and V. Jousseaume, "Role of Sb dopant in Ag:GeSx-based conducting bridge random access memories," *Phys. Status Solidi Appl. Mater. Sci.*, vol. 213, no. 2, pp. 311–315, Feb. 2016.

[32] H.-D. Kim, H.-M. An, S. M. Hong, and T. G. Kim, "Unipolar resistive switching phenomena in fully transparent SiN-based memory cells," *Semicond. Sci. Technol.*, vol. 27, no. 12, p. 125020, Nov. 2012.

[33] C. H. Yang *et al.*, "Electric modulation of conduction in multiferroic Ca-doped BiFeO 3 films," *Nat. Mater.*, vol. 8, no. 6, pp. 485–493, Jun. 2009.

[34] C. L. He *et al.*, "Nonvolatile resistive switching in graphene oxide thin films," *Appl. Phys. Lett.*, vol. 95, no. 23, p. 232101, Dec. 2009.

[35] J. Yao, Z. Sun, L. Zhong, D. Natelson, and J. M. Tour, "Resistive switches and memories from silicon oxide," *Nano Lett.*, vol. 10, no. 10, pp. 4105–4110, Oct. 2010.

[36] R. Wang *et al.*, "Bipolar Analog Memristors as artificial synapses for neuromorphic computing," *Materials (Basel).*, vol. 11, no. 11, Oct. 2018.

[37] I. Messaris, A. Serb, S. Stathopoulos, A. Khiat, S. Nikolaidis, and T. Prodromakis, "A data-driven verilog-A ReRAM model," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3151–3162, Dec. 2018.

[38] W. Sun *et al.*, "Understanding memristive switching via in situ characterization and device modeling," *Nat. Commun.*, vol. 10, no. 1, pp. 1–13, Dec. 2019.

[39] Y. Yang *et al.*, "Electrochemical dynamics of nanoscale metallic inclusions in dielectrics," *Nat. Commun.*, vol. 5, no. 1, pp. 1–9, Jun. 2014.

[40] S. Bagdzevicius, K. Maas, M. Boudard, and M. Burriel, "Interface-type resistive switching in perovskite materials," *J. Electroceramics*, vol. 39, no. 1–4, pp. 157–184, Dec. 2017.

[41] G. Sassine, S. La Barbera, N. Najjari, M. Minvielle, C. Dubourdieu, and F. Alibart, "Interfacial versus filamentary resistive switching in TiO 2 and HfO 2 devices," *J. Vac. Sci. Technol. B, Nanotechnol. Microelectron. Mater. Process. Meas. Phenom.*, vol. 34, no. 1, p. 012202, Jan. 2016.

[42] "IEEE International Roadmap for Devices and Systems™." [Online]. Available: https://irds.ieee.org/. [Accessed: 28-Feb-2020].

[43] M. Lanza *et al.*, "Recommended Methods to Study Resistive Switching Devices," *Advanced Electronic Materials*, vol. 5, no. 1. Blackwell Publishing Ltd, p. 1800143, 01-Jan-2019.

[44] "Memryx: Reconfigurable in-memory computing chips with unparalleled energy efficiency and compute density." [Online]. Available: https://www.memryx.com/home. [Accessed: 25-Mar-2020].

[45] "MemComputing Inc.: Powerful Co-Processors with Speed like no other." [Online]. Available: https://www.memcpu.com/. [Accessed: 25-Mar-2020].

[46] Z. Biolek, D. Biolek, and V. Biolková, “SPICE model of memristor with nonlinear dopant drift,” *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.

[47] M. D. Pickett *et al.*, “Switching dynamics in titanium dioxide memristive devices,” *J. Appl. Phys.*, vol. 106, no. 7, p. 074508, Oct. 2009.

[48] Y. V. Pershin and M. Di Ventra, “SPICE model of memristive devices with threshold,” *Radioengineering*, vol. 22, no. 2, pp. 485–489, 2013.

[49] L. Chen, C. Li, T. Huang, H. G. Ahmad, and Y. Chen, “A phenomenological memristor model for short-term/long-term memory,” *Phys. Lett. Sect. A Gen. At. Solid State Phys.*, vol. 378, no. 40, pp. 2924–2930, Aug. 2014.

[50] V. A. Slipko and Y. V. Pershin, “Importance of the window function choice for the predictive modelling of memristors,” *IEEE Trans. Circuits Syst. II Express Briefs*, pp. 1–1, Nov. 2019.

[51] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, “Memristive switching mechanism for metal/oxide/metal nanodevices,” *Nat. Nanotechnol.*, vol. 3, no. 7, pp. 429–433, Jun. 2008.

[52] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, “Memristor SPICE modeling,” in *Advances in Neuromorphic Memristor Science and Applications*, Springer Netherlands, 2012, pp. 211–244.

[53] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, “TEAM: Threshold adaptive memristor model,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 60, no. 1, pp. 211–221, Jan. 2013.

[54] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, “VTEAM: A General Model for Voltage-Controlled Memristors,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.

[55] H. Li *et al.*, “Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model,” *Proc. -Design, Autom. Test Eur. DATE*, vol. 2015-April, pp. 1425–1430, 2015.

[56] “Stanford-PKU RRAM Model.” [Online]. Available: https://nano.stanford.edu/stanford-rram-model. [Accessed: 28-Feb-2020].

[57] B. Chen *et al.*, “Highly compact (4F2) and well behaved nano-pillar transistor controlled resistive switching cell for neuromorphic system application,” *Sci. Rep.*, vol. 4, no. 1, pp. 1–5, Oct. 2014.

[58] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, “Complementary resistive switches for passive nanocrossbar memories,” *Nat. Mater.*, vol. 9, no. 5, pp. 403–406, Apr. 2010.

[59] Y. Yang, J. Mathew, D. K. Pradhan, M. Ottavi, and S. Pontarelli, “Complementary resistive switch based stateful logic operations using material implication,” in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–4.

[60] *The Authoritative Dictionary of IEEE Standards Terms*. IEEE 100, 2000.

[61] Y. Pang, B. Gao, B. Lin, H. Qian, and H. Wu, “Memristors for Hardware Security Applications,” *Adv. Electron. Mater.*, vol. 5, no. 9, p. 1800872, Sep. 2019.

[62] Y. S. Chen *et al.*, “Highly scalable hafnium oxide memory with improvements of resistive distribution and read disturb immunity,” in *Technical Digest - International Electron Devices Meeting, IEDM*, 2009.

[63] A. Chen and M. R. Lin, “Reset switching probability of resistive switching devices,” *IEEE Electron Device Lett.*, vol. 32, no. 5, pp. 590–592, May 2011.

[64] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, “Stochastic memristive devices for computing and neuromorphic applications,” *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, Jul. 2013.

[65] R. Soni *et al.*, “Probing Cu doped Ge0.3Se0.7 based resistance switching memory devices with random telegraph noise,” *J. Appl. Phys.*, vol. 107, no. 2, p. 024517, Jan. 2010.

[66] D. Veksler *et al.*, “Random telegraph noise (RTN) in scaled RRAM devices,” in *IEEE International Reliability Physics Symposium Proceedings*, 2013.

[67] S. Hamdioui, M. Taouil, and N. Z. Haron, “Testing open defects in memristor-based memories,” *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 247–259, Jan. 2015.

[68] M. Escudero-Lopez, E. Amat, A. Rubio, and P. Pouyan, “An experience with chalcogenide memristors, and implications on memory and computer applications,” in *2016 Conference on Design of Circuits and Integrated Systems, DCIS 2016 - Proceedings*, 2017, pp. 249–254.

[69] K. A. Campbell, “Self-directed channel memristor for high temperature operation,” *Microelectronics J.*, vol. 59, pp. 10–14, Jan. 2017.

[70] K. Drake, T. Lu, M. K. H. Majumdar, and K. A. Campbell, “Comparison of the electrical response of Cu and Ag ion-conducting SDC memristors over the temperature range 6 K to 300 K,” *Micromachines*, vol. 10, no. 10, Oct. 2019.

[71] J. Gomez, I. Vourkas, and A. Abusleme, “Exploring Memristor Multi-Level Tuning Dependencies on the Applied Pulse Properties via a Low Cost Instrumentation Setup,” *IEEE Access*, vol. 7, pp. 59413–59421, 2019.

[72] M. Escudero-Lopez, F. Moll, A. Rubio, and I. Vourkas, “An on-line test strategy and analysis for a 1T1R crossbar memory,” in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design, IOLTS 2017*, 2017, pp. 120–125.

[73] S. Stathopoulos *et al.*, “Multibit memory operation of metal-oxide Bi-layer memristors,” *Sci. Rep.*, vol. 7, no. 1, pp. 1–7, Dec. 2017.

[74] J. Y. Seok *et al.*, “A review of three-dimensional resistive switching cross-bar array memories from the integration and materials property points of view,” *Adv. Funct. Mater.*, vol. 24, no. 34, pp. 5316–5339, Sep. 2014.

[75] “Predictive Technology Model (PTM).” [Online]. Available: http://ptm.asu.edu/.

[76] M. Nicolaidis, “Theory of transparent BIST for RAMs,” *IEEE Trans. Comput.*, vol. 45, no. 10, pp. 1141–1156, 1996.

[77] M. Escudero, I. Vourkas, A. Rubio, and F. Molll, “On the Variability-aware Design of Memristor-based Logic Circuits,” in *Proceedings of the IEEE Conference on Nanotechnology*, 2019, vol. 2018-July, pp. 1–4.

[78] M. Escudero, I. Vourkas, A. Rubio, and F. Moll, “Variability-tolerant memristor-based ratioed logic in crossbar array,” in *Proceedings of the 14th IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2018*, 2018, pp. 13–18.

[79] M. Escudero, I. Vourkas, A. Rubio, and F. Moll, “Memristive logic in crossbar memory arrays: Variability-aware design for higher reliability,” *IEEE Trans. Nanotechnol.*, vol. 18, pp. 635–646, 2019.

[80] E. Lehtonen and M. Laiho, “Stateful implication logic with memristors,” in *2009 IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2009*, 2009, pp. 33–36.

[81] Q. Chen, X. Wang, H. Wan, and R. Yang, “A Logic Circuit Design for Perfecting Memristor-Based Material Implication,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 36, no. 2, pp. 279–284, Feb. 2017.

[82] S. Kvatinsky *et al.*, “MAGIC - Memristor-aided logic,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[83] L. Xie *et al.*, “Scouting Logic: A Novel Memristor-Based Logic Design for Resistive Computing,” in *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, 2017, vol. 2017-July, pp. 176–181.

[84] L. Guckert and E. E. Swartzlander, “MAD gates - Memristor logic design using driver circuitry,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 64, no. 2, pp. 171–175, Feb. 2017.

[85] 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.

[86] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, “MRL - Memristor Ratioed Logic,” in *International Workshop on Cellular Nanoscale Networks and their Applications*, 2012.

[87] M. Maestro-Izquierdo *et al.*, “Experimental Verification of Memristor-Based Material Implication NAND Operation,” *IEEE Trans. Emerg. Top. Comput.*, vol. 7, no. 4, pp. 545–552, Oct. 2019.

[88] E. Lehtonen, J. Poikonen, and M. Laiho, “Implication logic synthesis methods for memristors,” in *ISCAS 2012 - 2012 IEEE International Symposium on Circuits and Systems*, 2012, pp. 2441–2444.

[89] M. Laiho and E. Lehtonen, “Cellular nanoscale network cell with memristors for local implication logic and synapses,” in *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 2010, pp. 2051–2054.

[90] M. Escudero-Lopez, E. Amat, A. Rubio, and P. Pouyan, “An experience with chalcogenide memristors, and implications on memory and computer applications,” in *2016 Conference on Design of Circuits and Integrated Systems, DCIS 2016 - Proceedings*, 2017, pp. 249–254.

[91] I. Vourkas and G. C. Sirakoulis, “High-Radix Arithmetic-Logic Unit (ALU) Based on Memristors,” Springer, Cham, 2016, pp. 149–172.

[92] C. Li *et al.*, “Analogue signal and image processing with large memristor crossbars,” *Nat. Electron.*, vol. 1, no. 1, pp. 52–59, Jan. 2018.

[93] P. L. Thangkhiew and K. Datta, “Scalable in-memory mapping of Boolean functions in memristive crossbar array using simulated annealing,” *J. Syst. Archit.*, vol. 89, pp. 49–59, Sep. 2018.

[94] D. N. Yadav, P. L. Thangkhiew, and K. Datta, “Look-ahead mapping of Boolean functions in memristive crossbar array,” *Integration*, vol. 64, pp. 152–162, Jan. 2019.

[95] M. Escudero, I. Vourkas, and A. Rubio, “Stuck-at-OFF Fault Analysis in Memristor-Based Architecture for Synchronization,” in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design, IOLTS 2019*, 2019, pp. 33–37.

[96] M. Escudero, I. Vourkas, and A. Rubio, “Alternative Memristor-Based Topologies for Chaotic Circuit Synchronization,” in *2019 Int. Conf. on Memristive Materials, Devices & Systems (MEMRISYS)*, 2019.

[97] M. Escudero, I. Vourkas, and A. Rubio, “Alternative Memristor-based Interconnect Topologies for Fast Adaptive Synchronization of Chaotic Circuits,” *Chaos, Solitons & Fractals (revise & resubmit)*, 2019.

[98] 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.

[99] Y. V. Pershin, S. N. Shevchenko, and F. Nori, “Memristive Sisyphus circuit for clock signal generation,” *Sci. Rep.*, vol. 6, no. 1, pp. 1–6, May 2016.

[100] C. Fernandez, I. Vourkas, and A. Rubio, “Shortest Path Computing in Directed Graphs with Weighted Edges Mapped on Random Networks of Memristors,” *Parallel Process. Lett.*, vol. 30, no. 01, p. 2050002, Mar. 2020.

[101] V. Ntinas, I. Vourkas, G. C. Sirakoulis, and A. Adamatzky, “Mimicking Physarum Space Exploration with Networks of Memristive Oscillators,” in *Handbook of Memristor Networks*, Springer International Publishing, 2019, pp. 1241–1274.

[102] D. Kuzum, S. Yu, and H. S. Philip Wong, “Synaptic electronics: Materials, devices and applications,” *Nanotechnology*, vol. 24, no. 38, 2013.

[103] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, “An overview of reservoir computing: theory, applications and implementations,” in *European Symposium on Artificial Neural Networks*, 2007, no. April, pp. 25–27.

[104] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, “Reservoir computing using dynamic memristors for temporal information processing,” *Nat. Commun.*, vol. 8, no. 1, pp. 1–10, Dec. 2017.

[105] L. E. Suarez, J. D. Kendall, and J. C. Nino, “Evaluation of the computational capabilities of a memristive random network (MN3) under the context of reservoir computing,” *Neural Networks*, vol. 106, pp. 223–236, Oct. 2018.

[106] J. Burger and C. Teuscher, “Variation-tolerant computing with memristive reservoirs,” in *Proceedings of the 2013 IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2013*, 2013, pp. 1–6.

[107] R. Zhang *et al.*, “Nanoscale diffusive memristor crossbars as physical unclonable functions,” *Nanoscale*, vol. 10, no. 6, pp. 2721–2726, Feb. 2018.

[108] H. Nili *et al.*, “Hardware-intrinsic security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors,” *Nat. Electron.*, vol. 1, no. 3, pp. 197–202, Mar. 2018.

[109] J. Yang *et al.*, “A low cost and high reliability true random number generator based on resistive random access memory,” in *Proceedings - 2015 IEEE 11th International Conference on ASIC, ASICON 2015*, 2016.

[110] Z. Wei *et al.*, “True random number generator using current difference based on a fractional stochastic model in 40-nm embedded ReRAM,” in *Technical Digest - International Electron Devices Meeting, IEDM*, 2017, pp. 4.8.1-4.8.4.

[111] T. Zhang *et al.*, “High-speed true random number generation based on paired memristors for security electronics,” *Nanotechnology*, vol. 28, no. 45, p. 455202, Oct. 2017.

[112] Y. Pang, B. Gao, B. Lin, H. Qian, and H. Wu, “Memristors for Hardware Security Applications,” *Adv. Electron. Mater.*, vol. 5, no. 9, p. 1800872, Sep. 2019.

[113] Y. Li, L. Zhao, W. Chi, S. Lu, and X. Huang, “Implementation of a new memristor based chaotic system,” in *Proceedings of the 5th International Workshop on Chaos-Fractals Theories and Applications, IWCFTA 2012*, 2012, pp. 92–96.

[114] G. Wang, M. Cui, B. Cai, X. Wang, and T. Hu, “A chaotic oscillator based on HP memristor model,” *Math. Probl. Eng.*, vol. 2015, 2015.

[115] L. V. Gambuzza, M. Frasca, L. Fortuna, V. Ntinas, I. Vourkas, and G. C. Sirakoulis, “Memristor crossbar for adaptive synchronization,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 64, no. 8, pp. 2124–2133, Aug. 2017.

[116] G. Boeing, “Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction,” *Systems*, vol. 4, no. 4, p. 37, Nov. 2016.

[117] A. Buscarino, L. Fortuna, M. Frasca, and G. Sciuto, *A concise guide to chaotic electronic circuits*. .

[118] M. P. Kennedy, “Three Steps to Chaos—Part II: A Chua’s Circuit Primer,” *IEEE Trans. Circuits*

Syst. I Fundam. Theory Appl., vol. 40, no. 10, pp. 657–674, 1993.

[119] ANTONIOU A, "REALISATION OF GYRATORS USING OPERATIONAL AMPLIFIERS, AND THEIR USE IN RC-ACTIVE- NETWORK SYNTHESIS," in *Proceedings of the Institution of Electrical Engineers*, 1969, vol. 116, no. 11, pp. 1838–1850.

[120] F. Ling and J. Proakis, *Synchronization in Digital Communication Systems*. Cambridge University Press, 2017.

[121] L. M. Pecora and T. L. Carroll, "Master stability functions for synchronized coupled systems," *Phys. Rev. Lett.*, vol. 80, no. 10, pp. 2109–2112, Mar. 1998.

[122] L. Huang, Q. Chen, Y. C. Lai, and L. M. Pecora, "Generic behavior of master-stability functions in coupled nonlinear dynamical systems," *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 80, no. 3, p. 036204, Sep. 2009.

[123] P. De Lellis, M. Di Bernardo, and F. Garofalo, "Synchronization of complex networks through local adaptive coupling," *Chaos*, vol. 18, no. 3, p. 037110, Sep. 2008.

[124] J. H. Fewell, "Social insect networks," *Science (80-.).*, vol. 301, no. 5641, pp. 1867–1870, Sep. 2003.