Università degli Studi di Modena e Reggio Emilia
Dipartimento di Ingegneria "Enzo Ferrari"

International Doctorate in
Information and Communication Technologies
XXXIV Cycle

Innovative Energy-Efficient Circuits Enabled by Resistive Memory devices for Secure In-Memory Computing

Candidate:
Tommaso Zanotti

Advisor:
Prof. Francesco Maria Puglisi

Co-Advisor:
Prof. Paolo Pavan

Director of the PhD school:
Prof. Sonia Bergamaschi

Acknowledgments

I would like to express my deepest gratitude to my primary supervisor, prof. Francesco Maria Puglisi, for his guidance and continuous support during the three years of the PhD program.

Also, I would like to thank prof. Paolo Pavan for his support, advice, and proficuous discussions together with fellow researcher and master thesis students of the Electronics group at UniMORE.

My special thanks also go to my love, Chiara Bachechi, for her encouragement to pursue the PhD, for always keeping me motivated and for her love and support through the years.

Finally, I wish to extend my special thanks to my parents and the rest of my family, as this achievement would have never been possible without their endless encouragement and support.

Contents

Abstract

English version

The number of smart devices for the Internet of Things (IoT) is rapidly growing, and by 2025 almost 80 ZB of data per year will be generated by IoT devices alone, challenging the current cloud computing infrastructure. Thus, a shift to the edge computing paradigm, in which data are processed near their sources, is critical, but its implementation requires new energy efficient computing hardware. The approaching downscaling limit of transistor size implies the need for new nanoscale technologies and a departure from the conventional von Neumann architecture. Also, in-hardware security primitives need to be introduced at the silicon level.

Among the possible technologies, emerging non-volatile memories (eNVMs) are very promising and enable the realization of in-memory computing paradigms, in which computation is executed directly inside the memory, therefore bypassing the slow and energy inefficient data exchange over a communication bus, i.e., the main bottleneck of von Neumann architectures. However, the intrinsic stochastic nature of eNVMs presents several challenges which can impact the circuit functionality and reliability. On the other hand, it can be exploited to implement hardware-level security primitives such as True Random Number Generators (TRNGs) and Physical Unclonable Functions (PUF). Thus, appropriate design tools and methodologies are needed to

help circuit designers exploit eNVMs strengths while consciously addressing their limitations.

The optimization of circuit simulation tools and the development of appropriate methodologies to analyze and improve innovative circuits based on eNVMs for computing and security applications is the goal of this PhD thesis. Specifically, a physics-based Resistive RAM (RRAM) compact model (UniMORE compact model), was developed starting from a prototypical existing version, and refined to include self-consistently the role of variability, thermal effects, and Random Telegraph Noise (RTN). In addition, an automated parameter extraction procedure is developed and included. Such procedure requires only the results of a few experiments that are commonly employed in the device characterization, and was validated both experimentally and on three RRAM technologies from the literature. The procedure allows quick model calibration and helps in determining the strengths and weaknesses of different RRAM technologies for a dependable device-circuit co-optimization. The calibrated compact model is used to analyze the performance and reliability trade-offs of different in-memory computing paradigms. Specifically, the results of circuits simulations of state-of-the-art Logic-in-Memory (LiM) circuits based on the material implication (IMPLY) logic and RRAM technology enabled the development of design procedures for optimizing their reliability, which are here discussed. Also, a novel smart IMPLY (SIMPLY) LiM architecture, which solves the circuit reliability issues of conventional IMPLY architectures, is proposed. The reliability and performances of the SIMPLY architecture were thoroughly investigated considering different RRAM technologies and benchmarked on complex operations. Furthermore, the results of the study on RRAM-based low-bit precision neural networks (NNs) analog hardware accelerators are presented, highlighting specific reliability and

performance trade-offs. Also, a novel hybrid in-memory computing hardware accelerator in which both SIMPLY and the analog vector matrix multiplication framework coexist on the same memory crossbar array is demonstrated. Finally, challenges and opportunities for RTN-based TRNG circuits are presented. Specifically, the impact of different materials and fabrication processes on the quality of the generated RTN signal and consequently on the output of a TRNG circuit implementation is discussed.

Italian version

L'enorme mole di dati prodotta dai dispositivi per l'Internet of Things richiede una trasformazione dell'attuale infrastruttura di cloud computing: lo spostamento di parte dell'elaborazione dove i dati vengono generati (edge computing). Per attuare questo cambio di paradigma, lo sviluppo di nuove soluzioni hardware più efficienti per l'elaborazione dei dati è fondamentale. Inoltre, il sopraggiungere del limite fisico di miniaturizzazione dei transistor implica la necessità di sviluppare nuovi nanodispositivi e nuove architetture di calcolo, che si scostano dalla tradizionale architettura di von Neumann. Oltretutto, i sempre più stringenti requisiti di sicurezza richiedono l'introduzione di primitive di sicurezza direttamente a livello hardware. In questo panorama, le tecnologie emergenti nell'ambito delle memorie non volatili (eNVM) rappresentano una soluzione promettente e permettono l'implementazione di paradigmi di elaborazione in memoria, che eliminano il principale collo di bottiglia delle architetture di von Neumann, ovvero l'inefficiente scambio di dati tra la memoria e l'unità di elaborazione. Tuttavia, la natura stocastica dei dispositivi eNVM influisce sulla funzionalità e sull'affidabilità di questi circuiti, complicandone la progettazione. D'altra parte, questi fenomeni stocastici possono essere sfruttati per implementare primitive di sicurezza a

livello hardware. Pertanto, per sfruttare le potenzialità delle eNVM sono fondamentali nuovi strumenti e nuove metodologie di progettazione. L'obiettivo di questa tesi di dottorato è l'ottimizzazione di strumenti per la simulazione circuitale e lo sviluppo di metodologie appropriate per l'analisi ed il miglioramento di circuiti innovativi basati sulle eNVM per applicazioni di elaborazione e sicurezza. In particolare, viene proposto un modello compatto di RAM resistiva (RRAM) (modello compatto UniMORE), sviluppato a partire da una versione prototipale esistente e perfezionato per includere in modo autoconsistente il ruolo della variabilità, degli effetti termici e del Random Telegraph Noise (RTN). Inoltre, viene descritta una procedura di estrazione dei parametri automatica che richiede solo i risultati di alcuni esperimenti comunemente impiegati durante la caratterizzazione dei dispositivi, e che è stata validata sia sperimentalmente che su tre tecnologie RRAM dalla letteratura. Grazie al modello compatto calibrato, vengono analizzate le prestazioni e l'affidabilità circuitale di diversi paradigmi di elaborazione in memoria. In particolare, mediante simulazioni circuitali di circuiti Logic-in-Memory (LiM) basati sulla logica di implicazione materiale (IMPLY) e sulla tecnologia RRAM, sono state sviluppate procedure di progettazione volte a massimizzarne l'affidabilità circuitale. Inoltre, viene presentata una nuova architettura LiM chiamata smart IMPLY (SIMPLY), che risolve i problemi di affidabilità comunemente presenti nei circuiti convenzionali. L'affidabilità e le prestazioni dell'architettura SIMPLY sono state studiate dettagliatamente considerando diverse tecnologie RRAM e l'esecuzione di operazioni complesse. Inoltre, tramite simulazioni circuitali sono stati analizzati acceleratori hardware di reti neurali a bassa precisione, evidenziandone i compromessi esistenti tra affidabilità ed elevate prestazioni. Viene anche presentato un nuovo acceleratore hardware che combina sullo stesso array di memoria due diversi

paradigmi di elaborazione in memoria, ovvero SIMPLY e l'accelerazione in analogico del prodotto matrice vettore. Infine, vengono discusse le sfide e le opportunità per i circuiti True Random Number Generators basati su RTN, mediante una analisi approfondita dell'impatto che differenti materiali e processi produttivi hanno sia sulla qualità del segnale RTN generato che sulle prestazioni di una realizzazione circuitale di TRNG.

Chapter 1

Resistive memory devices for the future of computing and security applications

1.1 Ultra-low power computing and security functions

1.1.1 Beyond Moore's law

From its invention in 1947 by the researchers at Bell Labs, the transistor has completely revolutionized many aspects of our life and promoted an exponential growth of the electronic industry. The diffusion of electronic devices and computing capabilities has corresponded to a rapid increase in the amount of produced data accompanied by increasing demand for more computing power. Scaling the size of devices corresponded to an increase of the number of transistors that could be fabricated in a chip, which doubled every two years according to Moore's law. This increasing device density together with

an increase of the clock frequency led to an increase in computing performance. Thanks to this strategy, the increasing demand for computing power was satisfied up until the early 2000s, when the increasing power density of transistors at smaller sizes required limiting the clock frequency to avoid overheating the chips. Still, the demand for increased computing power has never stopped growing highlighting the need for innovations both at the device and circuit architecture level. In addition, new technologies such as the Internet of Things (IoT) are expected to exponentially grow in the next few years. The use of low-power smart devices for the IoT alone is expected to produce almost 80 ZB of data by 2025 [1], challenging the conventional cloud computing infrastructure and demanding a shift to the edge computing paradigm. In edge computing, computations are performed in close proximity to where the data is generated, thus reducing the burden of inefficient data transmissions, improving user security, and reducing latency. However, to sustain such a paradigm shift ultra-low-power computing solutions are required. Therefore, to achieve the goal of ultra-low-power computing hardware solutions efforts in the scientific community and in the industry have been directed towards the research of new nanotechnology devices and computing architectures which depart from the traditional von Neumann architecture.

1.1.2 In-memory computing

In the conventional von Neumann architecture, the processing (i.e., CPU) and the storage of the data (i.e., memory) are separated and connected by a communication bus, see Fig. 1.1a. This separation is the source of high time and energy inefficiencies, leading to the common problem known as the von Neumann bottleneck (VNB) [2]. In fact, while the CPU is extremely efficient and fast in executing operations, the memory access and data transfer over

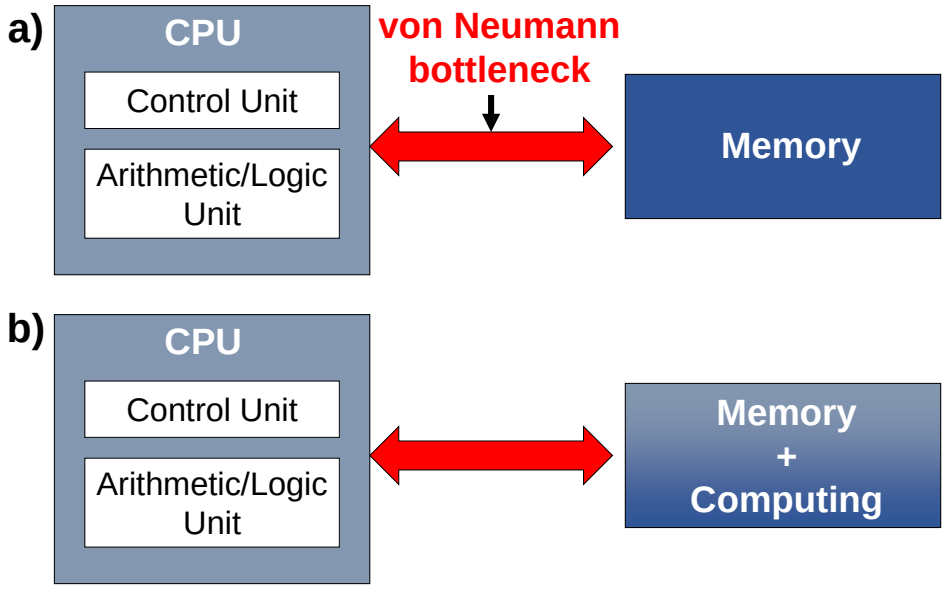


Figure 1.1: a) Sketch of the von Neumann architecture where the von Neumann Bottleneck is highlighted. b) Novel non-von Neumann architecture, which enables to execute operations directly in-memory, bypassing the von Neumann Bottleneck.

buses require orders of magnitudes higher energies and times [3]. Also, the growing adoption of machine learning and deep learning algorithms that are commonly characterized by data-intensive operations further exacerbates this issue.

In the strive for energy-efficient computing schemes, one of the most efficient solutions comes from nature, i.e., the human brain, which dissipates around 20W while performing 10^{15} calculations per second [4], [5]. Differently from the von Neumann architecture, in the brain processing and memory are co-located and take place in neurons and synapses. Such high energy efficiency is not the result of extremely fast and accurate neurons and synapses but rather of their dense interconnection. Thus, by taking inspiration from nature efforts have been directed towards the development of solutions bringing the computation and the storage closer to each other, realizing in-memory

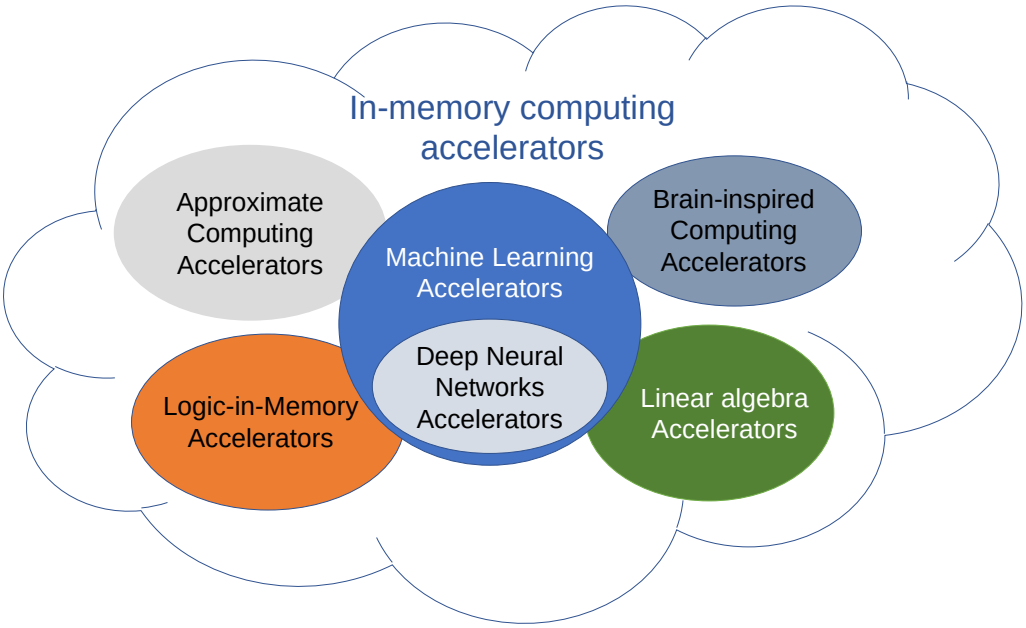


Figure 1.2: Different categories of in-memory computing hardware accelerators. Some overlap between different computing paradigms exist.

computing architectures [6]–[10], where the computation takes place inside the memory using the memory element both as a storing and as a computing element, see Fig. 1.1b.

These new in-memory computing solutions can be used to realize hardware accelerators and thus speed up the execution of specific tasks or operations. Different in-memory accelerators have been theorized in the last decade, which can be divided into the following five major groups, which are shown in Fig. 1.2. Specifically, Logic-in-memory (LiM) accelerators enable the efficient execution of logic operations directly in-memory [10]–[13]. Some applications may not require exact precision of the computations and tolerate higher error rates. Thus, such applications may benefit from the adoption of accelerators for the approximate computing paradigm [14], [15] in which high efficiency is achieved by trading off with lower accuracy of the compu-

tation. The rapid adoption of deep learning has prompted several solutions for the in-memory acceleration of deep neural networks (DNNs) [3], [7], [16]. Also, accelerators for linear algebra problems are being studied [9], [17], [18]. Finally, accelerators for biologically inspired computing paradigms such as spiking neural networks (SNNs), which mimic the neural networks in brains, composed of neurons and synapses, are being studied [4], [6], [8], [19], [20].

While some in-memory solutions exploiting well-known memory technologies (e.g., FLASH, DRAM, SRAM) have been proposed [21]–[25], new emerging nonvolatile memory (eNVM) technologies are currently the frontrunners for enabling the diffused adoption of these new computing paradigms, thanks to their low-cost, high-density, high switching speed, and the potential possibility of enabling the multi-bit storage on a single device. Although some prototypes have been successfully demonstrated [26]–[28], this field is still in its infancy. Before the diffused commercialization of these hardware accelerators a lot of research directed to improving the memory technology, the circuit architectures, and the development of new software primitives, still needs to be done.

1.1.3 Emerging nonvolatile memory technologies

Several new nonvolatile memory technologies enabled by innovations in the material sciences have been researched in the last decades, motivated by the need to fill a performance gap currently present in the memory hierarchy of computers. Currently, a performance gap exists [29] between the nonvolatile memory, commonly implemented with the relatively cheap and high capacity NAND flash technology, and the DRAM which provides the nonvolatile storage of information and higher speeds however at higher costs. Although eNVM technologies cannot compete in terms of cost per bit with NAND

flash and terms of speed with DRAM, they provide a good balance between the two, and thus were identified as a solution for implementing the storage class memory [30]–[32]. However, it was soon realized that such new eNVMs could enable the implementation of energy-efficient in-memory computing accelerators [3], [7], [9], [11], [26], [33].

In fact, the group of eNVM technologies commonly includes different two and three terminals nonvolatile memory devices, which by exploiting different physical mechanisms can be programmed to store a single bit or even multiple bits in a single device. These technologies include resistive random access memories (RRAM) such as metal-oxide (OxRAM) [34], [35] or conductive-bridging RAM (CBRAM) [36], [37], Phase Change Memories (PCM) which are based on chalcogenide materials [38]–[40], Ferroelectric RAM (FeRAM), Ferroelectric FET (FeFET) [41], Ferroelectric Tunnel Junction (FTJ) devices which exploit the properties of ferroelectric materials [39], [42], Spin-Transfer Torque (STT-MTJ) [39], [43] and Spin-Orbit Torque (SOT-MTJ) [44]–[48] magnetic RAM devices which employ ferromagnetic materials. In general, two-terminal devices are built by introducing a layer of a material (usually an insulator) with specific properties between two metal electrodes realizing a metal-insulator-metal (MIM) structure. Using electrical pulses it is possible to change the properties of the insulating material which produces a variation of the equivalent conductivity of the device. Thus, a bit of information can be stored with two different conductivity values, while for storing multiple bits in a single device, its conductance needs to be reliably programmed into more than two distinguishable values. Three terminal devices are commonly built as FET devices where specific materials are inserted between the gate electrode and the transistor channel. Thus, varying the properties of specific materials inserted under the gate electrode modulates the threshold voltage

of the FET device and thus the conductance between the source and drain terminals for a specific gate voltage.

For the fabrication of such eNVM devices, materials compatible with the back end of line (BEOL) fabrication processes are commonly employed, enabling the vertical exploitation of the integrated circuit, potentially stacking the memory array on top of the control logic that is fabricated in the front end of line (FEOL). Very high memory densities are commonly achieved. Specifically, two-terminal devices can be used to build 2D crossbar arrays where a memory device is placed at the intersection of each vertical and horizontal line. More crossbar arrays can be stacked on top of each other to increase the memory density (i.e., number of bits stored per unit area), however, 3D vertical arrays are more cost-effective but still only implementations based on resistive memories have currently been demonstrated [35].

A common characteristic of eNVM technologies is their intrinsic stochasticity which is introduced by defects caused by fabrication processes and by the intrinsic device behavior, which lead to several nonideal effects such as the resistance drift problem shown by PCM devices which determines a progressive increase of the device resistance or the cycle-to-cycle (C2C) and device-to-device (D2D) variation shown by resistive memory devices which causes the random distribution of their resistance. Thus, although some computing approaches such as the DNN, SNN, and linear algebra solver may benefit from the multi-bit storage, currently storing more than one bit in a single device is still challenging [35], [49].

Also, eNVM technologies are characterized by different performance and characteristics [38], [41], [47]–[51]. As shown in Table 1.1, although all the eNVM reported providing faster programming and reading speed and longer endurance compared to NAND flash memory technologies, still none of them

Table 1.1: Comparison of the performance of different eNVM technologies . Data from [38], [41], [47]–[51]

	Flash	**PCM**	**RRAM**	**STT-MTJ**	**SOT-MTJ**	**FeFET**
Endurance	$< 10^5$	$10^8 - 10^{11}$	$10^6 - 10^{12}$	$> 10^{15}$	$> 10^{15}$	$< 10^8$
Retention	>10y	>10y	>10y	>10y	>10y	>10y
ION/IOFF	$> 10^8$	$> 10^4$	$> 10^3$	>2	>2	$> 10^8$
Programming energy	$\approx$ 10 fJ	$\approx$ 10 pJ	$\approx$ 100 fJ	$\approx$ 100 fJ	$\approx$ 100 fJ	< 10 fJ
3D integration	yes	yes	yes	no	yes	yes
Write speed	<10 ms	<100 ns	<10 ns	<5 ns	<1 ns	<10 ns
Read speed	<10 μs	<10 ns	<10 ns	<10 ns	<10 ns	<10 ns

is able to meet all the ideally required performance criteria for every in-memory computing architecture [49]. In fact, an ideal memory device would require very high endurance (i.e., $> 10^{17}$), >10 years retention, low-voltage (i.e., <1 V) operation, large memory window (i.e., $I_{ON}/I_{OFF} > 10^2$), low-programming energy (1 fJ/bit), fast programming speed (i.e., <10 ns), and 3D integration. Thus, although improvements are still required from the technology perspective, in the short-term circuit designers should identify the most suitable technology for a specific application while compensating its weaknesses during the circuit design. Thus, device-circuit co-optimization strategies need to be developed together with fast and accurate compact models.

1.1.4 Hardware security building blocks

Among the requirements of IoT and edge computing, security together with energy efficiency is the most important. In fact, IoT systems need to be resilient to possible malicious attacks which aim to get access to private (e.g., interfere with bank transactions) and sensitive information or mine the reliability of a system. For instance, malicious attacks can come in the form

of physical or network attacks [52]. In IoT devices or systems, these attackable sources need to be protected by introducing appropriate security primitives. Although software-based security solutions have been developed, it was demonstrated that hardware-based security solutions are far more secure [53]–[55]. Still, even with state-of-the-art integrated circuit technologies, past cyber-attacks (e.g., the row hammer attack [56]), by succeeding, have demonstrated possible limitations of current solutions. Also, the adoption of new nanotechnologies, such as eNVM, requires the development of new hardware security primitives. Thanks to their intrinsic stochasticity, eNMV technologies offer new opportunities for reaching the high-security targets required by new systems, which however still need to be thoroughly investigated. In general, security solutions, such as the encryption of messages, user authentication to a system, require the generation of random private keys. In this regard, the stochasticity displayed by eNVM devices can be used as an entropy source. Different hardware security primitives based on eNVM have been proposed in the literature [54], [57]–[62]. Specifically, true random numbers generators are used to generate random bit streams which can be used to build random keys used for encrypting messages [52], [55], [58], and an example application is illustrated in Fig. 1.3a. Also, the Physical Unclonable Function (PUF) is a promising hardware security primitive and can be used in authentication applications [58], [62]. A PUF typically exploits the randomness introduced as a side effect during manufacturing [61] to create random challenge-response pairs that are unique for each fabricated chip, see Fig. 1.3b where an example application is described. Finally, machine learning systems based on eNVM have been proposed as a possible hardware solution that could help in identifying anomalies (i.e., possible attacks) during the device operation [52].

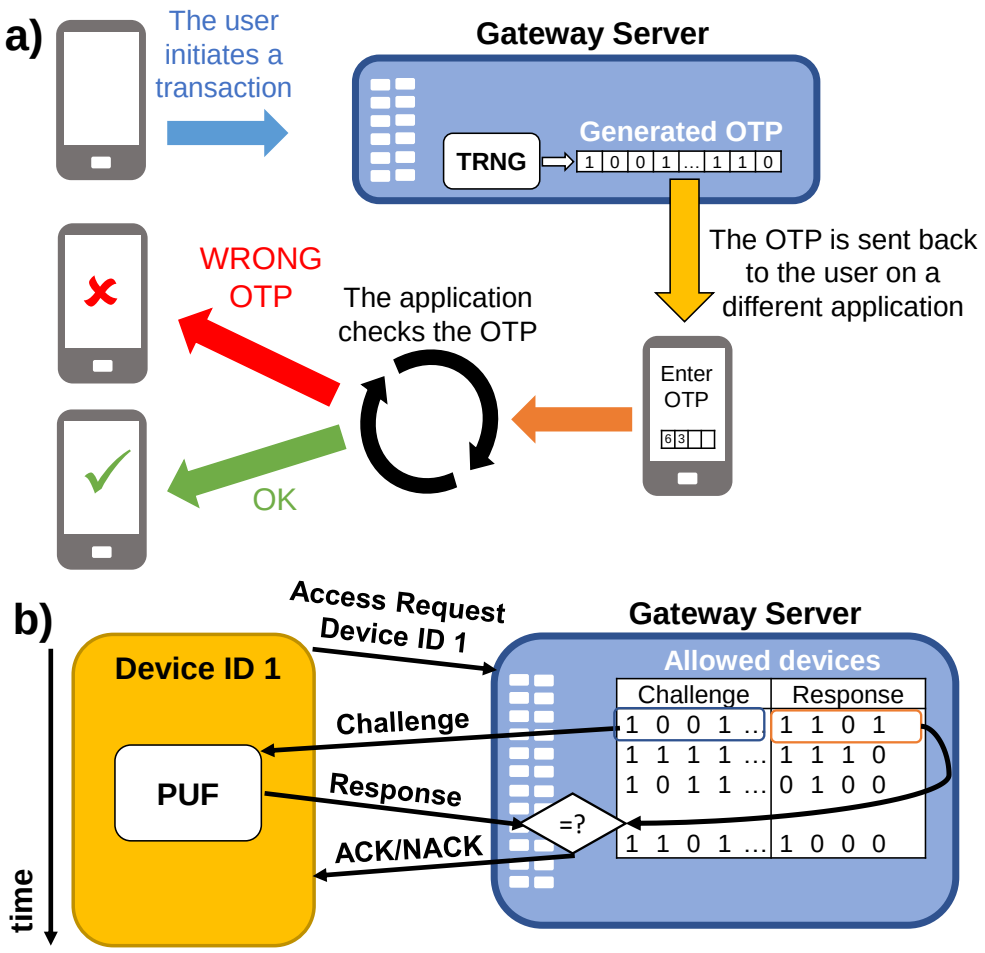


Figure 1.3: a) Example application of a TRNG circuit. A TRNG generates a OTP, which is used to authenticate a transaction in a two-factor authentication system. b) Use case application example of a PUF circuit, for authenticating a device in a system. The device sends an access request to the server, which responds by sending the challenge specific to the device ID requesting the access. In the device, the challenge is input to the PUF which outputs the response that is sent back to the server which compares it with the one stored in its internal memory, and grants or denies the access.

1.1.5 The focus of this thesis

Although a lot of research activity has been directed toward the development of novel solutions for the IoT by exploiting eNVM technologies, enabling their rapid diffusion still requires significant research efforts.

The work presented in this thesis addresses the need for ultra-low-power and secure computing solutions by studying and developing computing and security schemes exploiting the RRAM technology. Among the in-memory computing solutions described in Section 1.1.2, LiM and DNN hardware accelerator are analyzed and developed, while novel True Random Number Generator (TRNG) circuits are discussed as a promising hardware security primitive.

1.2 Resistive Random Access Memories

Among the different eNVM technologies, RRAM technologies are one of the most promising thanks to their simple fabrication, high switching speed, low programming energy, 3D integration, and compatibility with CMOS processes. An RRAM device consists of a two-terminal MIM structure. When considering the binary memory case as an example, the resistance of the device can be switched between a high resistive state (HRS) and a low resistive state (LRS) which are commonly associated with "0" and "1" logic values. When no external voltage is applied to the RRAM device it retains the stored value, which can be read by applying a small read voltage pulse across its terminals. Depending on the materials used during the fabrication of the MIM structure, different switching mechanisms can occur, such as the valence change mechanism (VCM) [34] and the electrochemical conduction mechanism (ECM) [36], which result in the OxRAM and CBRAM categories.

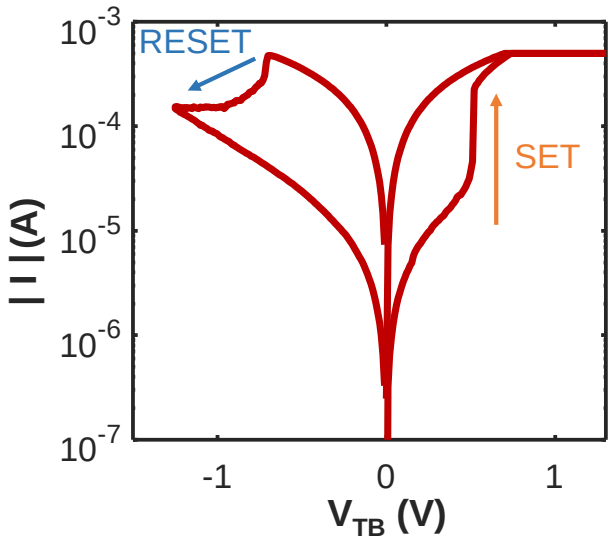


Figure 1.4: Example of the I-V characteristic of a bipolar RRAM device. The set and reset events are highlighted. V_{TB} is the voltage applied across the top and bottom electrodes.

In this thesis, we focus on the metal-oxide-based RRAM cell exhibiting filamentary switching.

1.2.1 Metal Oxide MIM devices

The first experiments on MIM structures date more than 40 years back, when it was observed that oxides that are nominally insulators can transition into a conductive state as a consequence of an abrupt switching event [63], [64]. However, the first experiments did not result in a sufficiently reliable resistive switching and thus could not be used for memory application. A resurgent and intensified research activity regarding RRAMs started after the demonstration by Samsung of NiO-based memory cells in 2004 [65]. By then several works have investigated the effect of different metal-oxide material combinations [34]. In fact, the resistive switching behavior depends on the oxide material, the used metal electrodes, and their interfacial properties [34].

In general, resistive switching is achieved by applying across the MIM electrodes appropriate voltages. To switch a device from a LRS to a HRS (i.e., reset operation) a V_{RESET} voltage is delivered to the device, while the opposite transition (i.e., set operation) requires the use of a V_{SET} voltage. Depending on the polarity of V_{SET} and V_{RESET}, devices can be divided in unipolar and bipolar switching categories. In unipolar switching, V_{SET} and V_{RESET} have the same polarity and the execution of a reset or set operation depend on the amplitude of the applied voltage. In bipolar switching, V_{SET} is positive while V_{RESET} is negative, resulting in a butterfly shaped IV relation, as shown in Fig 1.4. Also, by modulating the amplitude of the reset voltage pulse, it is possible to program a device in different nominal HRS values. In this thesis we consider bipolar switching devices.

From a device physics perspective, the switching mechanism sketched in Fig. 1.5 is associated with the formation and dissolution of conductive filament (CF). To initiate resistive switching, fresh devices undergo a forming process, which corresponds to a soft breakdown of the dielectric. As a result of the application of high electric fields, oxygen ions drift to the top electrode (TE) interface where are accumulated [34]. After the forming step a CF of oxygen vacancies is created and the device is in LRS.

When performing a reset operation, the oxygen ions migrate back to the dielectric where they recombine with the oxygen vacancies, thus partially dissolving the CF and restoring the HRS [34].

The set operation, is similar to the forming process, however it requires lower voltages. Also, in OxRAM technologies the forming and the set processes are commonly abrupt, thus requiring the introduction of a current compliance. The value of the current compliance determines the size of the CF filament and thus the LRS resistance value.

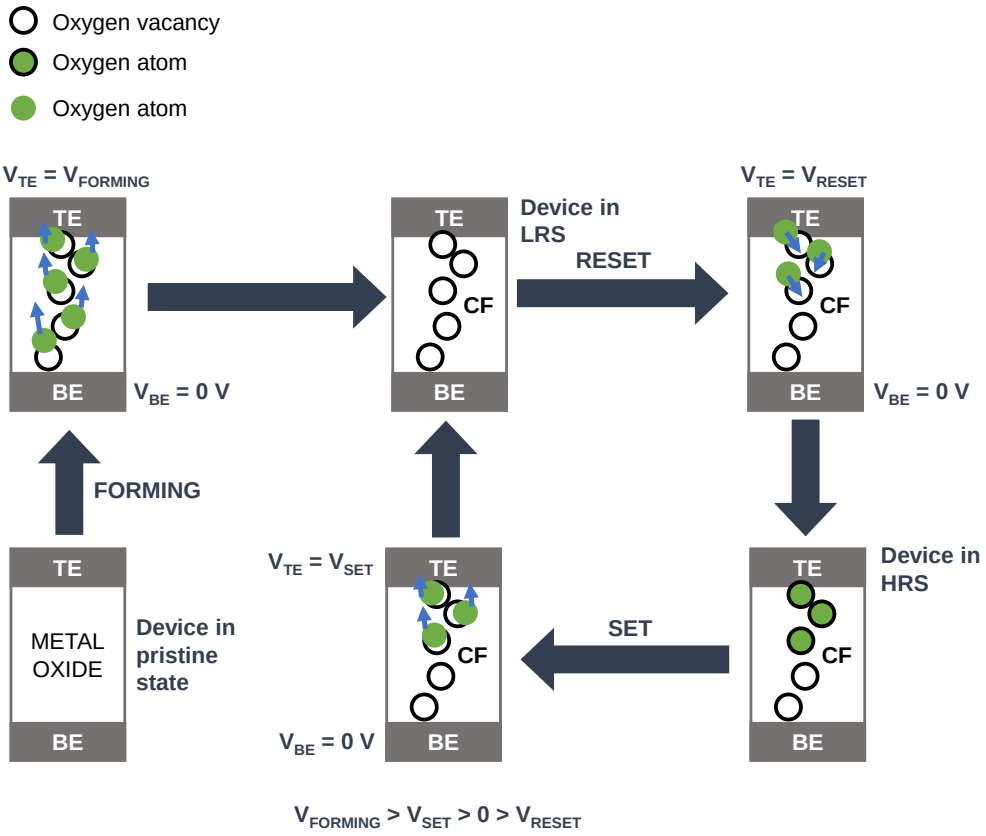


Figure 1.5: Sketch representation of the switching processes occurring in RRAM devices, which is represented as a dielectric material stacked between a metallic top and bottom electrodes (i.e., TE and BE, respectively). The forming step results in the creation of a CF due to a soft dielectric breakdown that causes oxygen ions to drift to the TE. During reset, the oxygen ions recombine with the vacancies resulting in a HRS. During set, similarly to the forming step, results in the oxygen ions drift to the TE and the CF restoration. Figure adapted from [34].

Charge transport in the device has been demonstrated to be mainly assisted by oxygen vacancies [34], [66]–[68]. Specifically, in devices in LRS an Ohmic-like drift conduction through the defect rich CF is observed, while in HRS the current conduction is believed to be assisted by trap assisted tunneling (TAT) through the oxygen vacancies in the oxide [34], [66]–[68].

1.2.2 Stochastic behavior of RRAM devices

RRAM devices present different challenges for circuit designers. In fact, many nonideal effects influence the device behavior and potentially result in reliability issues if not appropriately considered during the design phase. The main challenges, currently come from variability and random telegraph noise (RTN), which induce random resistance variations which currently hinder the reliable multi-bit storage [69], [70].

Although D2D variations could be reduced by improving the fabrication processes, C2C variations are intrinsic to the device operation and result in randomly distributed LRS and HRS resistances [70]. In fact, the diffusion and recombination of oxygen ions during reset and the bond breaking during a set are partially stochastic processes [49], [69]. Thus, the shape and the number of oxygen vacancies composing the CF change from C2C, leading to a normally distributed LRS resistance [69]. Lower current compliance values reduce the size of the CF, thus making the charge transport more affected by variations of the number of vacancies building the CF, increasing the standard deviation of the LRS resistance distribution [49]. Also, when a device is reset, the position and number of vacancies in the oxide layer assisting the TAT charge transport change from C2C, leading to log-normally distributed HRS resistances [71]. An example is shown in Fig. 1.6a, where a considerable overlap between HRS distributions for different reset voltages

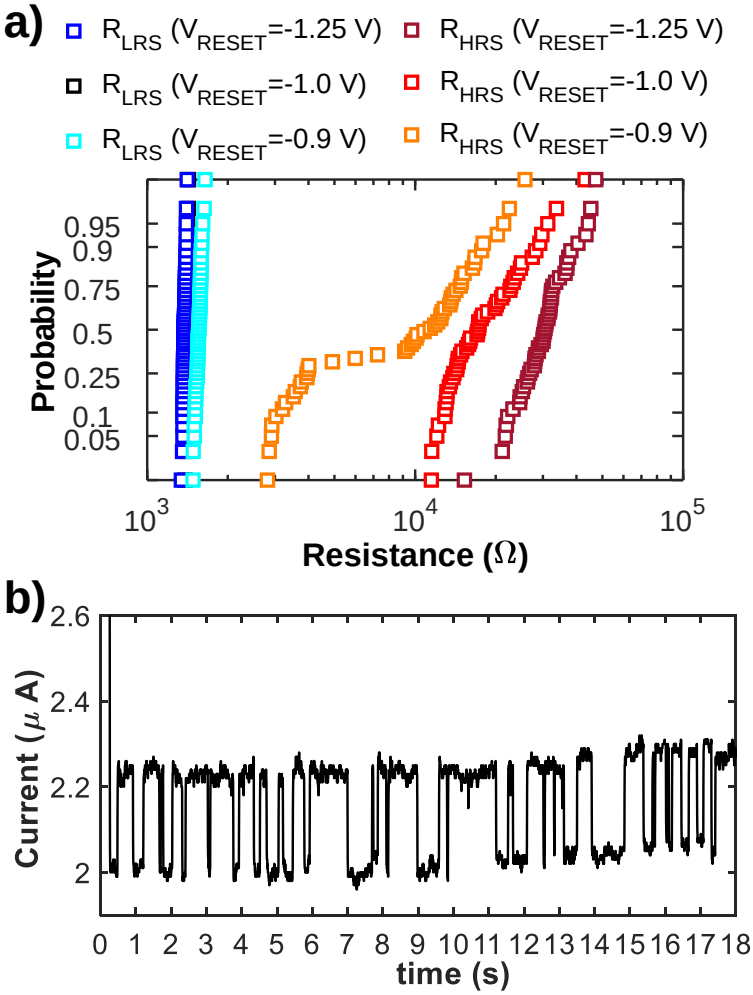


Figure 1.6: a) Example of the LRS and HRS distributions resulting from programming an RRAM device with different reset voltages ($V_{RESET} = -0.9V, -1.0V, -1.25V$). Due to C2C variations, a considerable overlap exists between the HRS distributions. b) Example of a two-level RTN signal from an RRAM device.

is clearly visible. Closed-loop programming algorithms, such as write and verify programming schemes [72], [73], can be used to limit the spread of the resistance distributions however at the cost of increased complexity and area on the chip.

RTN introduces unpredictable current fluctuations between two (i.e., two-level RTN) or more (i.e., multi-level RTN) discrete levels, which negatively impact the circuit reliability, especially during device read operations, because it reduces the available readout margin [69], [71]. For a device in HRS, RTN is commonly linked to the temporary activation and de-activation of oxygen vacancies that normally assist the TAT charge transport, by the effect of charge trapping and de-trapping in slower oxygen ions defects not assisting charge transport, that are located in their proximity. In LRS, RTN current fluctuations are connected to the trapping and de-trapping of charges in defects nearby the CF which perturb the potential in their surroundings, partially screening the charge transport in the CF due to the Coulomb interaction [74]. An example of a two-level RTN current signal is shown in Fig. 1.6b.

In general, all these stochastic effects together with other devices' non-idealities (e.g., conduction nonlinearity, self-heating) need to be taken into account during the circuit design phase to achieve reliable circuits operation [75], [76]. Thus, physics-based compact models capable of reproducing such effects are fundamental tools to enable reliable circuit simulations.

1.3 In-memory computing with state-of-the-art RRAM devices

Different in-memory computing hardware accelerators based on RRAM devices have been proposed in the literature. As discussed in the previous sections, variability and RTN limit the number of bits that can be reliably stored in a single device, thus making the practical realization of some in-memory computing paradigms very challenging and hindering their commercialization in the near future. For instance, although the in-memory training of DNNs would considerably reduce the energy needed to train the large number of parameters of DNNs it would require at least 6 bits (i.e., 64 discrete resistance levels) to be stored on a single RRAM device [28]. Thus, in this thesis in-memory computing paradigms which can exploit current state-of-the-art RRAM technologies are discussed. Specifically, LiM and DNN inference hardware accelerators.

1.3.1 Logic-in-Memory

LiM computing architectures are currently being researched as a solution for implementing reconfigurable hardware accelerators which enable the execution of logic operations using the memory element both for storage and computing. Although, some non in-memory implementations of hybrid CMOS-eNVM approaches in which logic gates combining both technologies have been theorized [77], LiM technologies represent a more promising solution, since they circumvent the VNB and offer intrinsic parallelism and reconfigurability, enabling the implementation of single instruction multiple data (SIMD) computing architectures.

Different LiM solutions based on RRAM devices, or memristor in general,

have been proposed [10]–[12], [26], [78]–[80], and some common features are here described. Specifically, a first difference with respect to conventional digital logic gates is the use of devices' nonvolatile resistances as the inputs of computations. Different LiM solutions usually differ for their set of core operations, which commonly comprise a complete logic group thus enabling the computation of any logic operation. For instance, material implication (IMPLY)- based architectures execute IMPLY and FALSE operations [10], [26], [81], the Memristor aided logic (MAGIC) [11] the NOR and NOT operations, while the scouting logic [79], [80] AND, OR, NOT logic operations. Therefore, while more complex logic functions are usually implemented by using more transistors in conventional CMOS gates, in LiM approaches it equates to the execution of longer sequences of core operations (i.e., increased latency in the computation). Depending on the type of scheme adopted for the driving signals, LiM computing paradigms can be divided into stateful and non-stateful.

Stateful LiM paradigms

LiM architectures are stateful when both the inputs and outputs of the computation are encoded in the non-volatile resistance of RRAM devices. This category includes the IMPLY-based and MAGIC architectures. In general, in stateful computing paradigms appropriate voltage pulses are delivered to multiple memory devices encoding the inputs and output of the computation. Depending on the resistive state of the input devices, the output changes state according to the truth table of the specific core operation that is implemented. Although these approaches, represent an interesting opportunity which does not require the use of additional control logic, the concurrent execution of the computing and the conditional programming steps leads to

complex design trade offs and reliability issues [10], [79], [82]. Such reliability issues can be easily overlooked when performing circuit simulations with simplified device compact models leading to unreliable circuits [82].

Non-stateful LiM paradigms

LiM architectures are non-stateful when the inputs of the logic operation are encoded with devices' resistive states while their outputs with voltages. Also, the peripheral circuitry of the memory array is used in the computations of logic operations. An example is the scouting logic [80], where the output of a voltage divider between the inputs RRAMs is compared with multiple thresholds. However, the effects of C2C and D2D variability, and RTN, result in considerable overlap between the voltage distributions for different inputs configurations (e.g., it is not possible to distinguish the case when both inputs are in LRS or when only one is in LRS), resulting in unreliable circuit operation and high bit error rates (BERs). The enhanced scouting logic [79] was introduced to improve the circuit reliability however at the cost of a much larger circuit area by employing 2T1R devices arrays, limiting their competitiveness.

1.3.2 Analog computing with memory arrays for deep neural networks

DNNs require the execution of a few operations that are however repeated on multiple data. Considering the simple multi-layer perceptron neural network shown in Fig. 1.7, each layer of the network is composed of several artificial neurons each characterized by different parameters (i.e., the weights multiplying the inputs and a bias term) that are learned during training. When performing an inference task, each neuron computes the sum of the

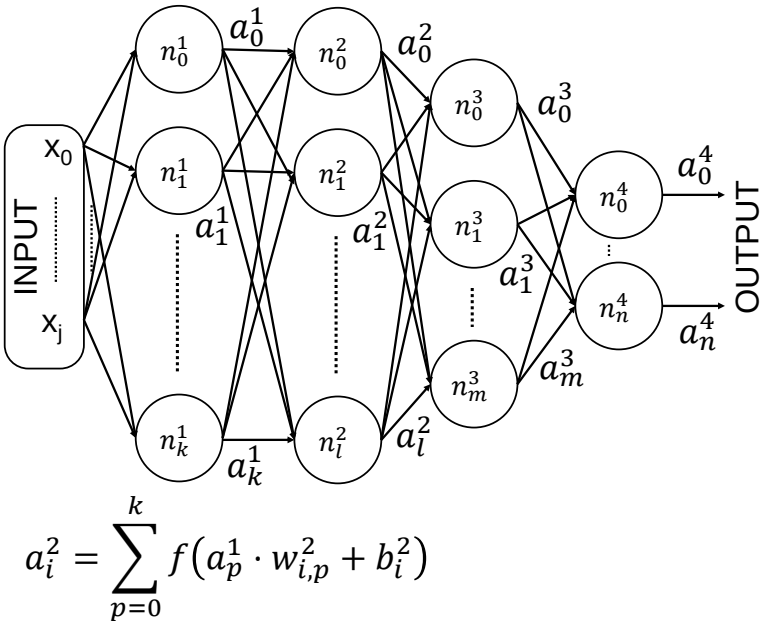


Figure 1.7: Structure of a multilayer perceptron neural network with three hidden layer. The operations computed by each neuron are reported.

products between its inputs and its weights (i.e., the multiply and accumulate operation (MAC)), and a bias term is added to the result. Finally, a nonlinear function computes the output activation of the neuron. Since most of the common networks topologies are composed of several layers with many neurons, the MAC operations use most of the computing resources and memory bandwidth due to the need of retrieving the neurons' weights. Thus, hardware accelerators based on RRAM devices for accelerating the MAC operation are being intensively studied [7], [16], [83], [84]. Indeed, the in-memory computation of the MAC operations would provide considerable energy efficiency improvement with respect to conventional solutions.

As shown in Fig. 1.8, by mapping the weights of each neuron to resistance levels of RRAM devices in the columns of a crossbar array, and encoding the input activation in the row voltages of the crossbar, the results of the MAC operation is computed in a single step for each neuron, by exploiting the

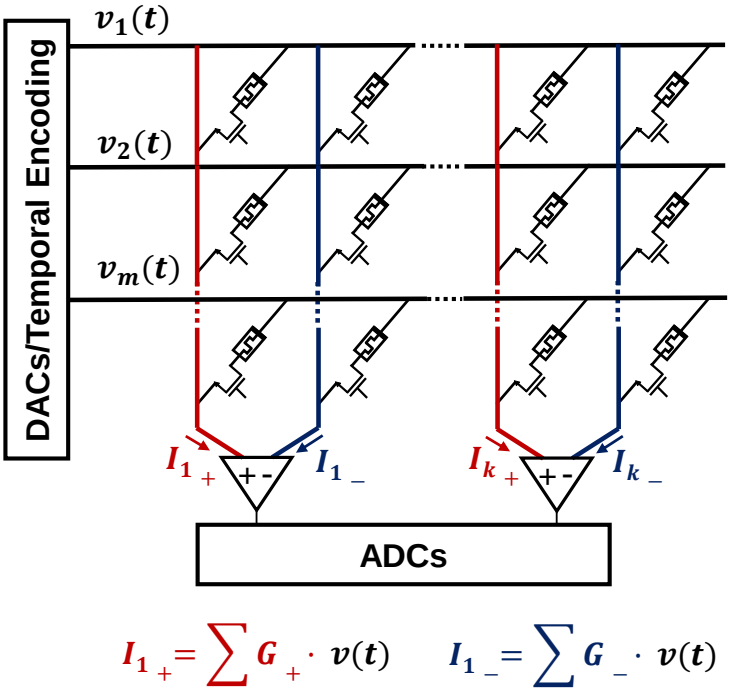


Figure 1.8: Representation of the core structure of an in-memory accelerator of the MAC operations based on 1T1R memory arrays. Each weight of a layer of the network is mapped onto the resistance of two devices located on the same row and adjacent columns (red and blue columns), enabling to encode both positive and negative weights. The row decoder, equipped with digital to analog converters (DACs), delivers the input activations as voltages to the rows of the memory array. The result of the MAC operation is computed as the difference between the currents flowing in each pair of columns (each pair of red and blue columns) which is digitized with ADCs.

Kirchhoff's current law and the Ohm's law. The result of the MAC operation is encoded in the current flowing in each column. Such currents are commonly converted to voltages with transimpedence amplifiers and digitized with analog to digital converters (ADCs) [7], [16], [83], [84]. Also, to map positive and negative weights to positive resistances, two devices are commonly used to encode both positive and negative weights, as shown in Fig. 1.8.

Due to the current limitation of the multi-bit storage, DNN approaches that employ a reduced number of bits during inference should be preferred. For instance, binarized neural networks (BNN) [85] have been shown to provide high accuracy despite the use of a single bit to represent weights and activations. Also, slightly increasing the number of bits encoding the activations, realizing low-bit precision neural networks (LBPNN) have been shown to result only in a slight accuracy drop compared to full precision networks. Some examples of RRAM-based BNN and LBPNN accelerators have been proposed in the literature [28], [57], [86], [87], however, still, a clear analysis of the circuit reliability in the presence of RTN and variability is missing, together with a clear understanding of the circuit design trade-offs.

These are parts of the focus of this thesis, and the results of the circuit reliability analysis of a LBPNN implementation enabled by the UniMORE RRAM physics-based compact model are discussed in Chapter 3 together with appropriate design strategies.

1.4 True Random Number generator circuits

As mentioned in the previous sections, TRNG circuits are a fundamental hardware security primitive which is used to generate bitstreams of random

bits. For instance, TRNG circuits are used for the generation of temporary one-time passwords (OTP) that are used when strong authentication algorithms are needed [88]. Ideally, a TRNG circuit should output random bits with i) high entropy, ii) sufficiently high throughput, iii) consuming low energy, iv) providing high security to potential malicious attacks.

To achieve these goals, different RRAM-based TRNG circuits have been proposed in the literature, each exploiting different stochastic phenomena that are intrinsic to RRAM devices. Specifically [55]:

- The probability of a resistive state transition after the application of an appropriate bias voltage
- The time required to switch a device between two resistive states
- C2C resistance variations
- RTN current fluctuations

Thus, C2C variations and RTN represent the two main entropy sources. In fact, by programming a device with voltage pulses with different duration and amplitude, it was shown that for some pulse duration and amplitude pairs a 50% of chance of switching a device from a HRS to a LRS can be achieved [89]. Thus, a random bit can be obtained by first performing a device set, followed by a probabilistic reset operation, after which the resistive state of the device is read with a small read voltage pulse.

Also, the time required to switch a device changes from C2C, and it can be collected with the circuit in Fig. 1.9a presented by Jiang et al. [90]. In this circuit, a digital counter measures the time required for a device to switch from a HRS to a LRS after the application of a programming pulse.

The circuit in Fig. 1.9b, shows the TRNG implementation from [91] which exploits the resistance variations from C2C. Specifically, the two devices are

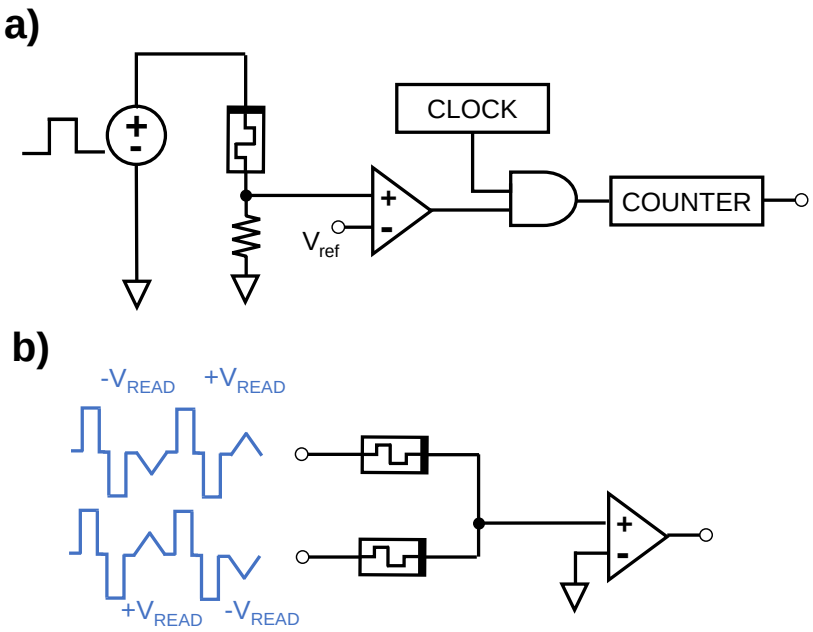


Figure 1.9: a) Example of a circuit from [90] extracting the time required to reset a device. After resetting the counter a reset pulse is delivered to the RRAM device. The counter starts to count with an appropriate clock frequency. A comparator detects when the resistance of the device passes a threshold, stopping the counter. b) Example of a TRNG circuit from [91] exploiting the C2C resistance variations of two RRAM devices as an entropy source. After resetting the devices two complementary read pulses are delivered to the two devices. By inverting the polarity at each read cycle, potential bias issue are prevented.

programmed independently in the same resistive state, and a random bit is generated by applying two complementary voltage pulses simultaneously to the two devices and comparing the common node with 0 V. Thus, a "0" logic bit is output when the resistance of the device that is read with a negative voltage is lower than that of the other device, otherwise, "1" logic bit is output.

Although these approaches can result in TRNG circuits with relatively high throughput, they are based on the switching of RRAM devices, thus potentially incurring into reliability issues due to the limited device endurance. Also, the switching of a device requires more energy compared to a device read operation.

Thus, RTN-based TRNG circuits, despite the lower achievable throughput which is limited by the defects capture and emission times, are a promising and more energy-efficient, low-cost solution that does not incur endurance issues since they do not require the switching of the RRAM devices. Although, a few examples of RTN-based circuit architectures using the RTN signal have been proposed [92]–[96], a clear understanding of the impact of material, and fabrication processes on the quality of the RTN signal are still missing. Ensuring a high-quality RTN signal is fundamental for achieving reliable and robust TRNG circuits. Conversely, identifying possible limitations of RTN signals produced by a specific technology could promote the activity of circuit designers to devise appropriate compensation strategies at the circuit level. Adding this piece of information to the literature is one of the objectives of this thesis and it is discussed in Chapter 4.

1.5 Thesis structure

In this thesis we advance the state-of-the-art, addressing several problems that are currently present in the field of ultra-low-power computing and security applications with RRAM devices, starting from the circuit simulations to the development of new circuital solutions.

Specifically, in Chapter 2 we present the UniMORE RRAM physics-based compact model. This model is implemented in Verilog-A, following the programming best practices, and includes the effect of RRAM devices nonidealities (i.e., self-heating, C2C, D2D, and RTN), making it an optimal solution for accurate circuit reliability simulations. A simple automated parameter extraction procedure is presented and verified on experimental data of four RRAM technologies, thus enabling the use of device-circuit co-optimization strategies.

In Chapter 3, we use the UniMORE RRAM physics-based compact model to i) assess the main reliability issues affecting conventional IMPLY-based LiM architectures, ii) to propose the novel smart-IMPLY (SIMPLY) LiM architecture, demonstrating its improved reliability and benchmarking its performance on different tasks, iii) to identify and evaluate performance and reliability trade-offs in a LBPNN implementation and proposing appropriate design strategies, and iv) to devise a novel in-memory computing architecture enabling the coexistence on the same memory array of the SIMPLY and the analog vector-matrix multiplications (VMM).

In Chapter 4, we present the result of the analysis of RTN-based TRNG circuits, i) showing the impact of different MIM stack of materials and fabrications techniques on the quality of the generated RTN considering its application as an entropy source and ii) proposing a solution for increasing the throughput of the generated output bitstream.

Finally, a general discussion of the results presented in this thesis together with an overview of possible future research directions is presented in the "Conclusions" section.

Chapter 2

RRAM physics-based compact modeling

2.1 RRAM compact models features and requirements

The study of RRAM-based circuits and architectures requires fast and accurate compact models, that can run on common SPICE or SPECTRE circuit simulators. Also, to enable the study of the reliability and performance of common RRAM-based circuits and applications compact models should reproduce both the device behavior in different operating conditions, and the effect of device nonidealities such as self-heating, variability, and RTN [75]. For instance, when studying memory array applications the effect of variability influence the number of bits that can be reliably stored in a single device while RTN reduces the read margin and thus increases the BER. In the last decades, several RRAM compact models have been proposed in the literature [97], each characterized by different strengths and weaknesses, but

none of them incorporate all the desired features, as shown in Table 2.1. In general, to be used in circuit simulators, compact models should be implemented either in SPICE or in Verilog-A, with the latter being the standard programming language for compact modeling in the semiconductor industry due to its high flexibility [98]. Indeed, most RRAM models are implemented either in SPICE or in Verilog-A, but other features are also important.

Specifically, general-purpose memristor models [99]–[102] can reproduce the median device characteristic under specific operating conditions (e.g., commonly the device IV characteristic) through simple equations. However, their accuracy is questionable when used to reproduce other operating conditions with the same set of parameters, potentially leading to errors in the circuit design. Moreover, thermal effects are not commonly included in general-purpose models, suggesting that different parameter calibrations could be required to reproduce the device behavior at different operating temperatures. Also, since the models parameters are not directly linked to the device physics, the parameter extraction procedure is non-trivial. Nevertheless, parameters extraction procedures have been developed for the Yakopcic [103] and Messaris [104] general-purpose compact models. However, the former focuses only on calibrating the device IV characteristic, thus providing limited usefulness in simulation where RRAM devices are programmed. Also, the parameter extraction procedure proposed for Messaris general-purpose compact model focuses specifically on neuromorphic computing applications, calibrating the model on the pulsed response. Despite these limitations, if appropriately used general-purpose models, thanks to their simplified modeling approach, are still a valuable tool for performing large-scale simulations without focusing on circuit reliability.

On the other hand, RRAM physics-based compact models include direct

Table 2.1: Comparison between RRAM compact models

RRAM Compact Models	General Purspose	Physics-based	Joule-heating (Quasi-Static/Dynamic)	C2C Variability	RTN
Yakopcic [99], [103]	✓		Not Considered		
TEAM/VTEAM [100], [101]	✓		Not Considered		
Messaris [102], [104]	✓		Not Considered		
Stanford-PKU [75], [105], [106]		✓	Quasi-static	✓	
JART VCM [107], [108]		✓	Quasi-static	✓	
Granada [109]–[111]		✓	Quasi-static	✓	✓
UniMORE [112]		✓	Dynamic	✓	✓

RRAM Compact Models	Verilog-A/SPICE	Clear Parameter Extraction Procedure	Implemented Following Verilog-A Programming Best Practices
Yakopcic [99], [103]	SPICE	✓	NA
TEAM/VTEAM [100], [101]	Verilog-A/SPICE		
Messaris [102], [104]	Verilog-A	✓	Mostly
Stanford-PKU [75], [105], [106]	Verilog-A/SPICE	Partially	Mostly
JART VCM [107], [108]	Verilog-A		
Granada [109]–[111]	SPICE		NA
UniMORE [112]	Verilog-A	✓	✓

links between model parameters and physical variables, potentially leading to more consistent parameter extraction procedures and the possibility to reproduce the device behavior in different operating conditions without the need to change the model parameters. Still, the available RRAM physics-based compact models either implemented in Verilog-A [75], [105]–[108] or SPICE [109]–[111] present some limitations which are highlighted in Table 2.1.

Specifically, while the device joule-heating is commonly included in such models, only the quasi-static heat equation is used. With this approximation, the device is always considered at thermal equilibrium and with a uniform temperature profile. Although this approach reduces the number of required differential equations, its accuracy when simulating sub-ns voltage pulses is questionable. Also, the effect of C2C and D2D variations is commonly reproduced by introducing noise sources on specific model parameters, but a variability model that can effectively reproduce the experimental variability in multiple operating conditions is still missing. Moreover, when investigating the circuit reliability the effect of multilevel RTN is of primary importance. However, RTN is commonly overlooked in RRAM compact models except the UniMORE [74] and Granada RRAM models, see Table 2.1, that are implemented in Verilog-A and SPICE, respectively. Another important characteristic of physics-based compact models is their numerical stability. The system of differential equations that are required to model the intrinsic RRAM devices' behavior results in a stiff mathematical problem, potentially causing convergence issues for the simulator. To improve the convergence of the simulator it is fundamental that the code implementing the compact model is designed following Verilog-A programming best practices [98], [113]. However, while Standford-PKU [75], [105] follows most of the best practices,

it formulates the system of differential equations in the integral form (i.e., implemented using the integral operator `idt()`), instead of the more numerically stable differential formulation (i.e., implemented using the integral operator `ddt()`), thus potentially resulting in simulation convergence issues, as described in [98], [113]. Finally, to implement device-circuit co-design strategies it is of foremost importance to define clear and simple parameters extraction procedure, thus enabling to quickly test the performance of different RRAM technologies for specific applications.

In the rest of this chapter, we present our UniMORE RRAM compact model [112] together with the developed automated parameter extraction procedure. As shown in Table 2.1, the proposed UniMORE RRAM compact model, is physics-based, implemented in Verilog-A following the programming best practices [98], [113], reproduces the device behavior in different operating conditions using a single set of parameters, considers joule-heating including the effect of thermal capacitance and separating the CF and barrier components, includes the effect of self-heating, C2C, and D2D variability, and multilevel RTN, and thus is an optimal tool to perform circuit simulations with a focus on circuit performance and reliability.

2.2 The UniMORE RRAM physics-based compact model

2.2.1 Compact model description

The UniMORE compact model is fully physics-based and supported by the results of advanced physical multi-scale simulations [66]. The model is implemented in Verilog-A and thus can run in common circuit simulators. A

first version of the model is already available on nanoHub [112], while in this thesis we present an updated version of the model aimed at improving the simulator convergence and the accuracy of the variability model. Specifically, the compact model reproduces the RRAM behavior considering the effect of self-heating, of multi-level RTN and variability both in LRS and HRS. In the model implementation, such nonideal effects and the nominal device behavior can be divided into three internal functional blocks which share the state of internal variables, as shown in Fig. 2.1c. In the following sections, each functional block is described.

RRAM module

The RRAM module reproduces the median device behavior including the effect of self-heating while leaving the modeling of nonideal effects to the other two functional modules. The equations of the RRAM module take full inspiration from the physical mechanism occurring in RRAM devices which are simplified by introducing reasonable approximations to enable a compact implementation. Thanks to the physics-based modeling approach most of the model parameters have a clear physical meaning, as shown in Table 2.2, thus simplifying the model calibration, as discussed later in Section 2.3. As shown in Fig. 2.1a, and b, the device resistance is split into a metallic-like CF composed of oxygen vacancies and a dielectric barrier component. In the compact implementation the CF resistance is modeled as an Ohmic conductor with an effective CF cross-section S, as described in Equations (2.1), (2.2), and (2.3), where also the effect of the CF temperature is included.

$$R_{LRS} = \frac{\rho \cdot t_{OX}}{S} \tag{2.1}$$

$$\rho = \rho_0 \cdot [1 + \alpha \cdot (T_{CF} - T_0)] \tag{2.2}$$

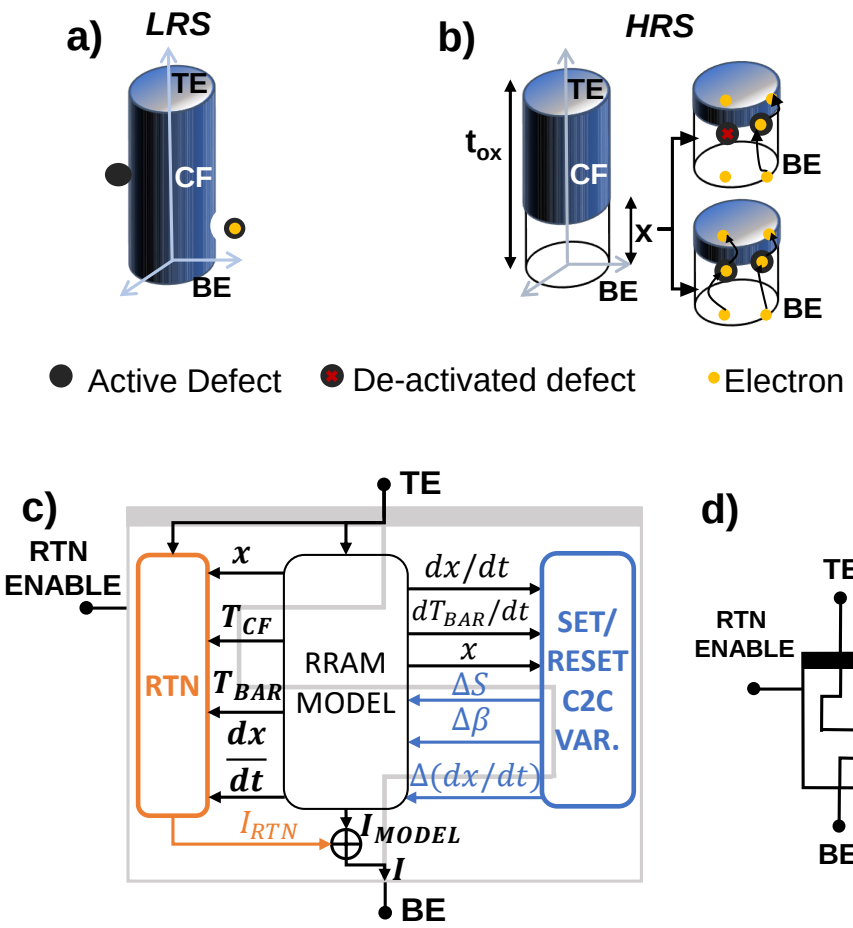


Figure 2.1: Sketch of an RRAM device in a) LRS, and b) HRS approximated with the compact model. The defects causing RTN and their activation and de-activation mechanism are reported for a device in LRS and a device in HRS. c) Functional block diagram of the complete compact model. The effects of RTN and variability are included with specific modules which exchange a few internal parameters with the core RRAM model. d) Compact model symbol used in circuit simulators. The RTN ENABLE terminal is used to inject RTN noise only during read operations.

Table 2.2: RRAM MODEL parameters description

Param.	Description
ρ	CF resistivity
t_{ox}	Dielectric barrier thickness
S	CF cross-section
T_{AMB}	Ambient temperature
T_0	CF resistivity reference temperature
E_R	TAT activation energy
k	Typical tunneling length
$V_{0_{HRS}}$	Non-linearity adjustment of the conduction of the dielectric barrier
$V_{0_{LRS}}$	Non-linearity adjustment of the conduction of the CF
α	CF resistivity thermal coefficient
β	Proportionality constant estimated from experimental data
c_0	Bond vibration frequency
$C_{P_{BAR}}$	Dielectric barrier thermal capacitance
$C_{P_{CF}}$	CF thermal capacitance
$k_{T_{BAR}}$	Dielectric barrier thermal conductance
$k_{T_{CF}}$	CF thermal conductance
k_{EX}	Barrier/CF mutual thermal conductivity
E_D	Oxygen ions diffusion activation energy
g	Field enhancement factor of oxygen ions diffusion
a	Barrier growth rate parameter depending on the current barrier thickness
b	Barrier growth rate parameter depending on the current barrier thickness
E_G	Bond breaking activation energy
f	Bond breaking field enhancement factor

$$R_{CF} = R_{LRS} \cdot \frac{t_{OX} - x}{t_{OX}} \tag{2.3}$$

As suggested by several works [34], [69], [114], [115], the current conduction mechanism incurring the dielectric barrier is commonly associated to the TAT. Thus, the barrier resistance (i.e., R_{BAR}) changes exponentially with the barrier thickness, and is modeled with Equation (2.4), where the -1 is added to make $R_{BAR} = 0$ when the barrier thickness $x = 0$.

$$R_{BAR} = R_{LRS} \cdot \beta \cdot (e^{\frac{x}{k}} - 1) \cdot e^{\frac{E_R}{k_b \cdot T_{BAR}}} \tag{2.4}$$

Therefore, to model the current conduction in the dielectric barrier a common solution is to employ the Simmons' tunneling barrier model [116] which is based on the hyperbolic sine function (i.e., *sinh*) and a nonlinearity parameter (i.e., $V_{0_{HRS}}$). Although in most RRAM devices the current conduction in the CF filament is linear, in our compact model a nonlinearity parameter (i.e., $V_{0_{LRS}}$) is included also in the equation modeling the current conduction in the CF since some RRAM devices can show some degree of nonlinearity also when a device is in LRS. Nevertheless, when $V_{0_{LRS}}$ is sufficiently high, its effect is negligible and the conduction in LRS is linear. The resulting overall current conduction model is described by Equations (2.5), (2.6), and (2.7), where V_{TB}, V_{BAR}, and V_{CF} are the voltage across the device and barrier and CF components, respectively.

$$I = \frac{V_{0_{LRS}}}{R_{CF}} \cdot \sinh\left(\frac{V_{CF}}{V_{0_{LRS}}}\right) \tag{2.5}$$

$$V_{BAR} = V_{0_{HRS}} \cdot sinh^{-1}\left(I \cdot \frac{R_{BAR}}{V_{0_{HRS}}}\right) \tag{2.6}$$

$$V_{BAR} = V_{TB} - V_{CF} \tag{2.7}$$

The state variables of the model are the barrier thickness (i.e., x), and the CF (i.e., T_{CF}) and barrier (i.e., T_{BAR}) temperatures, that are modeled dynam-

ically with the differential Equations (2.8),(2.9), (2.10), and (2.11). Specifically, Equations (2.8) and (2.9) reproduce the device behavior during the reset and set operations, respectively, by considering the field-driven and temperature-assisted oxygen ions drift and recombination during reset and the field- and temperature driven bond breaking and related defect generation during set [34], [69], [74], [112]. Thermal effects are modeled dynamically, considering the effect of the thermal conductance (i.e., $k_{T_{CF}}$, $k_{T_{BAR}}$, and $k_{T_{EX}}$) and thermal capacitance (i.e., $C_{P_{CF}}$, $C_{P_{BAR}}$) of both the CF and the barrier components, thus enabling accurate predictions also when using very short pulses [117].

$$\frac{dx}{dt} = -c_0 \cdot e^{-\frac{E_D - (g - a \cdot x^b) \cdot \frac{V_{TB}}{t_{OX}}}{k_b \cdot T_{CF}}} \qquad (reset) \tag{2.8}$$

$$\frac{dx}{dt} = c_0 \cdot e^{-\frac{E_G - f \cdot \frac{V_{TB}}{x}}{k_b \cdot T_{CF}}} \qquad (set) \tag{2.9}$$

$$\frac{dT_{BAR}}{dt} = \frac{1}{C_{P_{BAR}}} \cdot [V_{BAR} \cdot I - k_{T_{BAR}}(T_{BAR} - T_{AMB}) - k_{ex}(T_{BAR} - T_{CF})] \tag{2.10}$$

$$\frac{dT_{CF}}{dt} = \frac{1}{C_{P_{CF}}} \cdot [V_{CF} \cdot I - k_{T_{CF}}(T_{CF} - T_{AMB}) - k_{ex}(T_{CF} - T_{BAR})] \tag{2.11}$$

The resulting compact model fully captures the DC and ultra-fast pulsed characteristics with a single set of parameters, as it is shown later in Section 2.3.

RTN module

The RTN module is used to introduce multilevel RTN fluctuations in the device output current when enabled through a model parameter (i.e., `RTN_ON`) and the RTN ENABLE terminal shown in Fig. 2.1c, d. Also, the RTN module is physics-based, and its parameters are described in Table 2.3, and its

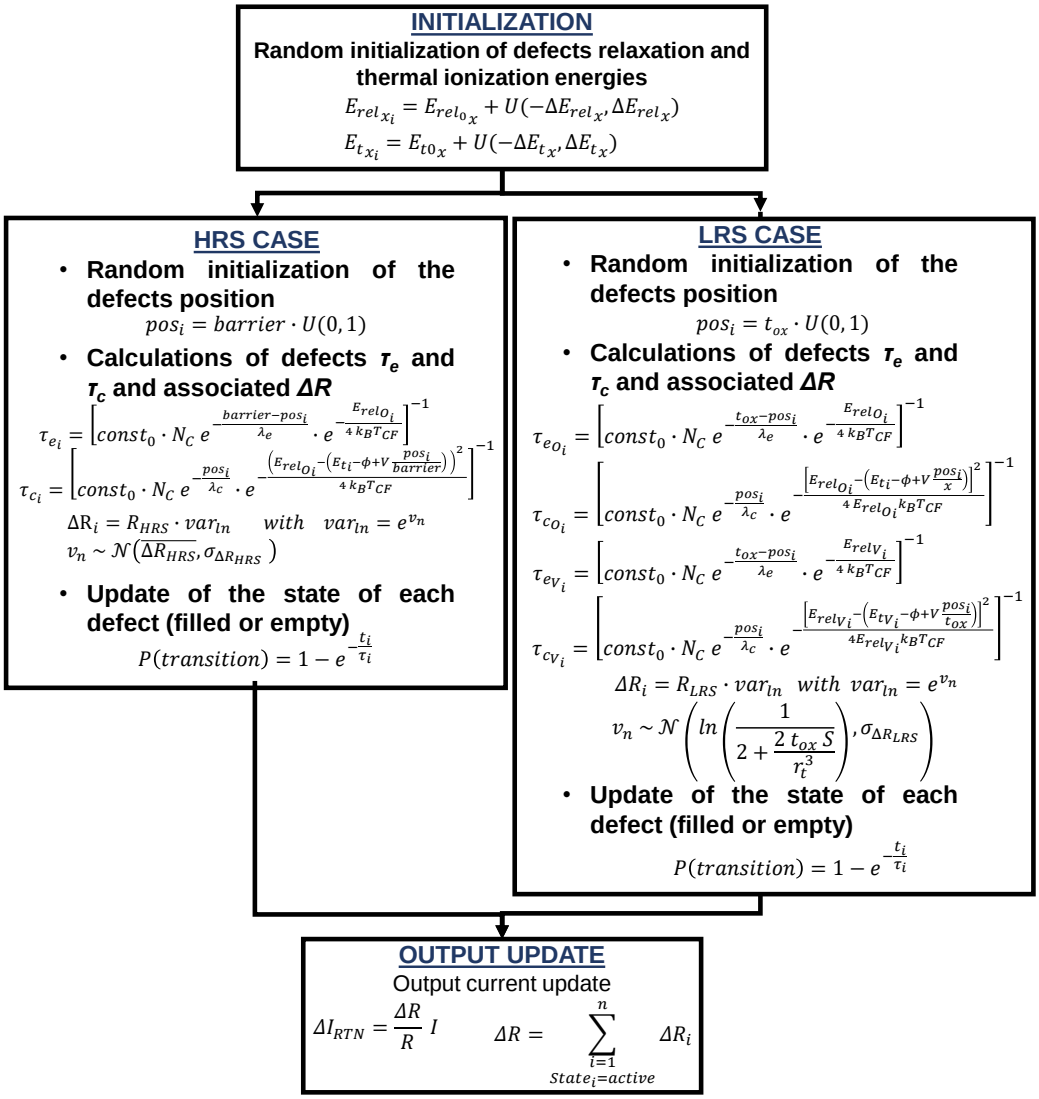


Figure 2.2: Flowchart of the RTN module. Both cases when a device is in LRS or HRS are considered. In the equations, N_V and N_O represent the oxygen vacancies and oxygen ions densities, respectively. τ_c and τ_e correspond to the capture and emission times associated with each defect. t_i is the time passed from the last defect state transition.

Table 2.3: RTN module parameters description

RTN model Param.	**Description**
$const_0$	Capture and emission times constant
N_C	Density of states at the bottom of the conduction band
ϕ	Energy barrier for injected electrons
λ_c	Typical tunneling length (capture)
λ_e	Typical tunneling length (emission)
$E_{rel_{0O}}$	Nominal oxygen ions relaxation energy
$E_{t_{0O}}$	Nominal oxygen ions thermal ionization energy
$E_{rel_{0V}}$	Nominal oxygen vacancies relaxation energy
$E_{t_{0V}}$	Nominal oxygen vacancies thermal ionization energy
ΔE_{rel_O}	Spread of the oxygen ions relaxation energy distribution
ΔE_{t_O}	Spread of the oxygen ions thermal ionization energy distribution
ΔE_{rel_V}	Spread of the oxygen vacancies relaxation energy distribution
ΔE_{t_V}	Spread of the oxygen vacancies thermal ionization energy distribution
$\overline{\Delta R_{HRS}}$	Mean of the normal distribution associated to the logNormal distribution of the R_{HRS} due to RTN
$\sigma\Delta R_{HRS}$	Standard deviation of the normal distribution associated to the logNormal distribution of the R_{HRS} due to RTN.
$\sigma\Delta R_{LRS}$	Standard deviation of the normal distribution of the R_{LRS} due to RTN.
r_t	Screening length of trapped charge in a defect
ρ_O	Oxygen ions density in the barrier
ρ_V	Oxygen vacancies density around the CF

equations are shown in Fig. 2.2. Such equations model the physical mechanisms that are associated with the origins of RTN, however considering appropriate approximations to enable a compact implementation. The RTN compact model considers the different RTN mechanisms that cause RTN for a device in LRS and HRS. Specifically, in LRS RTN is associated with the charge trapping and de-trapping in defects that are located in close proximity of the CF (i.e., within one Debye length). As shown in Fig. 2.1a such charged defects perturb the potential in their surroundings, thus introducing a screening effect on the CF portion in their proximity [74]. As shown in Fig. 2.1b, for a device in HRS the temporary de-activation of defects (i.e., usually oxygen vacancies) in the dielectric barrier that would normally assist charge transport is the main source of RTN current fluctuations. The origin of the de-activation of such defects is commonly linked to the trapping of charge in relatively slow defects (i.e., oxygen ions), that do not assist significant charge transport [74], [118]. Thus, in the RTN model, both oxygen ions and vacancies are considered. To mimic the defect generation and recombination phenomena associated with barrier growth and collapse, each time the barrier thickness changes or during each device initialization, such defects are randomly distributed in space along the vertical dimension depending on their density. When instantiated, each defect is characterized by a random initial state (empty or filled), a random vertical position (i.e., `pos`) with respect to the BE, an associated random resistance variation ΔR, and specific relaxation and thermal ionization energies (i.e., E_T, and E_{REL}, respectively), that are extracted from uniform distributions. The values of such physical parameters describing the defects can be taken from previous works from the literature [112], [119]. The RTN model then continuously computes each defect's capture and emission times (τc and τe, respectively)

depending on the values of specific variables that are shared with the RRAM MODEL. Specifically, the applied voltage across the device (i.e., V_{TB}), and the CF and barrier temperatures. A Markov chain is used to model the state transition. Specifically, the state change is modeled by using a random variable with a transition probability that depends both on the computed τc and τe, and on the time elapsed from the last state transition. Finally, the effect of all active defects is summed and the resulting RTN current is added to the overall device current.

Variability module

Among the intrinsic non-ideal characteristics of RRAM devices, variability is one of the most challenging and needs to be accounted for when analyzing the reliability of all RRAM-based circuits. In fact, when programming an RRAM device, C2C and D2D variations result in randomly distributed resistive states, which can reduce the memory window and the number of bits that can be reliably stored in a single RRAM device when using RRAMs for memory applications, but also introduce complex design constraints when considering in-memory computing applications, as described later in Chapter 3. From the device physics point of view, C2C and D2D variations were demonstrated to be associated with morphological variations in the CF structure, with the creation of multiple small CFs, and in the dielectric barrier [120], which lead to LRS and HRS variations, respectively.

Accordingly, as shown in Table 2.1, all the physics-based compact models replicate the effect of variability introducing random noise sources on internal variables. Specifically, in the Standford-PKU compact model noise sources are introduced in the system of differential equations [75], specifically on tunneling gap length, the CF radius, and on the energy barriers. Conversely,

in the JART compact model noise sources are introduced on the minimum and maximum oxygen ions concentrations in the dielectric barrier region, its length, and on the CF filament radius [108]. In the Granada compact model random variations are introduced on the CF filament volume [109], [111].

However, all the proposed modeling approaches and implementations for the variability result in two main shortcomings:

- The Verilog-A implementations result in long simulation run times
- Variability parameters need to be calibrated for each specific experimental condition. For instance, the variability parameters used to reproduce variations in quasi-static DC conditions are different than those used when considering fast voltage pulses.

In the following, we present two possible variability implementations which address the above-mentioned limitations of previous approaches and were included in the UniMORE compact model. Specifically, a "simplified variability model" and a "comprehensive variability model" are presented. The former introduces appropriate approximations to enable the inclusion of the effects of variability in the model, however only in specific operating conditions and thus is suitable when only a few experimental data are available. The "comprehensive variability model" instead, enables the reproduction of the effects of variability in multiple device operating conditions (i.e., quasi-static and pulsed operation) using a single set of parameters.

Simplified variability model

LRS variability

Similarly to the modeling approaches followed in other compact models, the LRS and HRS variability is reproduced by introducing zero-mean Gaus-

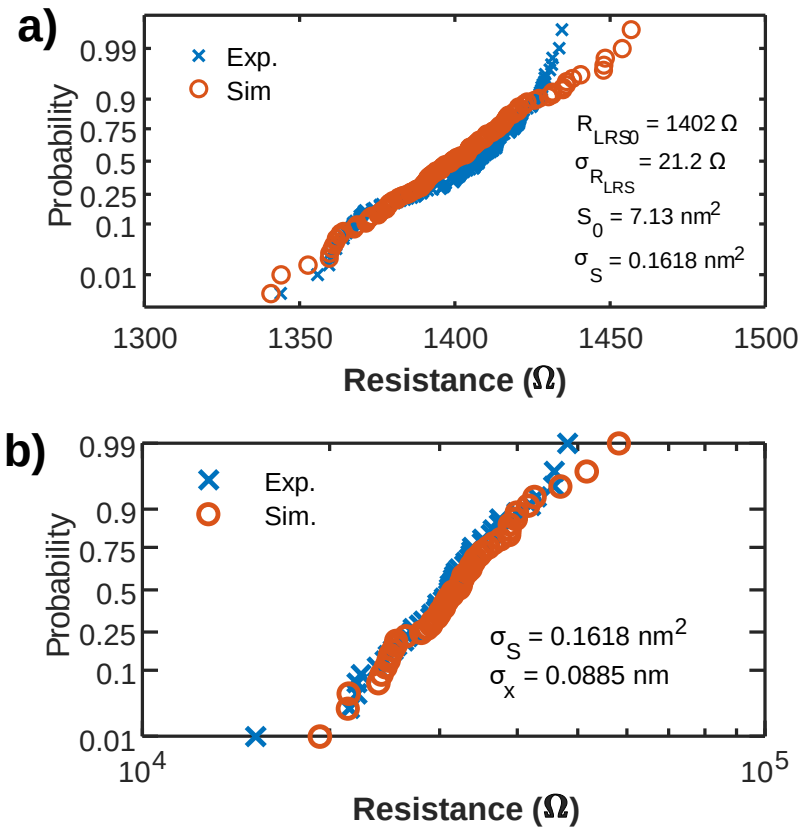


Figure 2.3: Experimental (blue crosses) and simulated (orange circles) a) R_{LRS}, and b) R_{HRS} resistance distributions. In a) the Experimental mean and standard deviation of the R_{LRS} and computed S distributions are reported. In b) the σ_S, and σ_x used in the simulations are reported.

sian random noise sources on the CF cross-section (i.e., S, see Table 2.2) and on the dielectric barrier thickness (i.e., x, see Fig. 2.1a). D2D and C2C variations are replicated by introducing such noise sources during the simulation initialization or the running phase, respectively. As shown in Fig.2.3, introducing a zero-mean Gaussian noise on the CF cross-section and by appropriately tuning its standard deviation (i.e., σ_S), enable to well reproduce the experimental R_{LRS} distribution. The value of σ_S can be estimated ana-

lytically from the experimental standard deviation of R_{LRS} (i.e., $\sigma_{R_{LRS}}$), as shown in Equation (2.12).

$$\sigma_S = \sigma_{R_{LRS}} \frac{dS}{dR_{LRS}} = \sigma_{R_{LRS}} \rho \frac{t_{OX}}{R_{LRS}^2} \tag{2.12}$$

HRS variability

As shown in Fig. 2.3b, the R_{HRS} follows a logNormal distribution. This effect can be reproduced by introducing a zero-mean Gaussian noise on the dielectric barrier thickness. In fact, as shown in Equation (2.13), an exponential relation exists between R_{BAR} and the barrier thickness x. Also, σ_x can be analytically estimated from experimental data, provided that the other compact models parameters have been already calibrated. As shown in Equation (2.13), R_{BAR} is first estimated by subtracting from the measured R_{HRS} the R_{LRS}, measured before the reset operation, to remove the effect of variations associated to the CF cross-section. Then σ_x can be directly estimated from Equation (2.14). As shown in Fig. 2.3b, also the HRS is well reproduced in simulations by introducing both the noise sources on S and x.

$$R_{BAR_{meas.}} = R_{HRS_{meas.}} - R_{LRS_{meas.}} \tag{2.13}$$

$$\sigma_x = \sigma \left[l \cdot ln \left(\frac{R_{BAR}}{\beta R_{LRS} e^{\frac{E_r}{k_B \cdot T_{BAR}}}} \right) \right] \tag{2.14}$$

In Verilog-A to implement the described variability model requires a particular care to avoid convergence issues that could result in a crash or in a slow down of the simulations. Introducing noise on the barrier thickness during a reset event is particularly complex as the noise is introduced on the barrier time derivative (i.e., $\frac{dx}{dt}$). Thus, the noise is integrated by the simulator depending on the simulation time-step. A common approach for introducing noises in simulations, which is used by other compact models, exploits the

Verilog-A function `$rdist_normal()`, which updates the noise value at each time-step. However, this approach is not appropriate to simulate RRAM devices due to the stiffness of the system of differential equations, which requires very short time-steps to simulate an abrupt set event, resulting in slow simulations, and longer time-steps to simulate a device read or reset operations, increasing the simulation speed. Ideally, circuit simulators, such as Spectre, can automatically adapt the simulation time-step, shrinking the time-step only when necessary. Changing the noise value at each time-step during a device reset or set, results in a perturbation of the mathematical problem at each time-step to which the solver responds by reducing the simulation time-step, thus slowing down the simulations. Also, in case of a set event, a positive feedback can easily arise, in which the simulator adopts shorter and shorter time-steps until the simulation fails. In other compact models, this is partially solved by using a fixed and sufficiently short simulation step, however reducing the simulation speed where such a short time-step is not required. Despite slowing down the simulations, the use of a fixed time-step solves another problem, which is the drift of the average of the noise introduced on the dielectric barrier thickness. Ideally, the average of the noise introduced on x should be zero, however, the use of adaptive time-step causes each noise value to be integrated for a different time shifting the average of the introduced noise. A better solution is to perform `transient noise` simulations, which enable to set the noise type (i.e., white, flicker noise) power, bandwidth, and update time-step. By setting the noise update parameter depending on the type of simulation that is performed, the simulator is no more constrained to use a fixed time-step to correctly introduce the noise in the simulations, but the adaptive time-step can be used instead, thus speeding up the simulations. Updating the noise value with a fixed frequency causes

each new noise sample to be integrated for the same time, avoiding the drift of the noise accumulated on x.

Variations on x and S must be introduced only during a reset and a set event, respectively, so when the barrier or the CF are changing. In this variability model, these two events are detected by checking when the barrier thickness time derivative crosses appropriate empirical thresholds (i.e., `th_set` and `ddt_x_clip_th` in [112], for the set and reset events, respectively). In OxRAM devices the set event is typically abrupt, thus detecting a set event is particularly easy, since the barrier rapidly drops to zero resulting in high negative $\frac{dx}{dt}$ (see Equation (2.9)). The Verilog-A `@cross()` function, which detects when a variable crosses 0 either with a positive or negative slope, can be used to check whether $\frac{dx}{dt}$ crosses `th_set`.

The detection of a reset event is activated when $\frac{dx}{dt}$ crosses `ddt_x_clip_th`. Since the noise is introduced on $\frac{dx}{dt}$, its value could intermittently drop below the threshold while still in a reset event, interrupting the noise injection and potentially drifting the average of the introduced noise. Thus an additional condition on the voltage across the device is considered. After $\frac{dx}{dt}$ crosses `ddt_x_clip_th`, the noise is injected as long as the applied voltage is decreasing.

While this approach improves the simulation speed and accuracy, the model variability parameters need to be adapted to reproduce variability when different experimental conditions are simulated, also when considering the same type of stimulus with just a different reset voltage. Nevertheless, this modeling approach is still particularly useful when calibrating the compact model on data from the literature, when a limited amount of experimental data are available (e.g., variability data in a single experimental condition are available), and only a specific condition is used in the circuit

simulations (e.g., single-bit storage in each RRAM).

Comprehensive variability model

LRS variability

The target of this comprehensive model is to reproduce variability in multiple experimental conditions. For this purpose, a complete dataset regarding variability is needed. To model the LRS variations a similar approach to the one shown before is employed, including an additional effect. Specifically, from experimental results in the literature, it is known that LRS variations are inversely proportional to the used current compliance (i.e., $\sigma_S \propto I_C^{-1}$) [121]. A higher I_C results in more or larger CF filaments making the device resistance in LRS less susceptible to small variations. As a first-order approximation, this effect can be included in the compact model updating the value of S during a set event with Equation (2.15), where S_0, $S_{0_{VAR}}$, indicate the simulated nominal CF cross-section and the CF cross-section at which σ_S has been experimentally measured, respectively.

$$S = S_0 + \mathcal{N}(0,1) \cdot \frac{\sigma_S \cdot S_{0_{VAR}}}{S_0} \tag{2.15}$$

If the data for the characterization of the LRS variability at different I_C are available, more complex models could be employed. For instance, considering the results from [121], Equation (2.16) could be employed to map the value of σ_S with respect to the current CF cross-section, where c is a constant that is determined by plotting $\frac{\sigma_S}{S}$ over S in a log-linear plot.

$$\sigma_S = S_0 \cdot \frac{\sigma_S}{S_{0_{VAR}}} \cdot 10^{c \cdot (S_0 - S_{0_{VAR}})} \tag{2.16}$$

HRS variability

The oxygen ions drift and recombination during reset depends both on the field and on the device internal temperature. Thus, also when considering the C2C variability, the effect of the internal device temperature can be used as a variable to introduce noises, especially during reset. Also in the Standford-PKU RRAM compact model, a temperature-dependent noise source is introduced in the barrier differential equation [122]. However, in such compact model, a simple temperature modeling was employed, using quasi-static heat equations and a single effective temperature combining both the barrier and CF components, therefore limiting the accuracy of the compact model when simulating fast voltage pulses. Here we present a new empirical modeling approach, based on experimental measurements, for reproducing the C2C variability when pulsed reset operations are performed. Also, in the proposed Verilog-A implementation, conditional statements are implemented with the more numerically robust smoothing function, instead of hard conditional structures (i.e., `if` clause), thus improving the convergence and complying with the programming best practices, as discussed in details in Section 2.2.2.

Pulsed reset variability

To devise the new empirical model, we experimentally measured the response of $TiN/Ti/HfO_x/TiN$ devices from Sematech to trains of reset pulses with different amplitudes (i.e., -1.3V and -1.1V) and different pulse widths (i.e., $10\mu s$, $1\mu s$, and $100ns$). As it can be observed from the experimental data in Fig. 2.4, the pulse number, width, and amplitude, influence the amount of the observed C2C variability (i.e., the value of $\frac{\sigma_R}{R}$). To evaluate the effect of the CF and barrier temperature components on the observed $\frac{\sigma_R}{R}$, the compact model parameters were calibrated on the device median behavior, thus omitting the effect of variability, following the parameter extraction

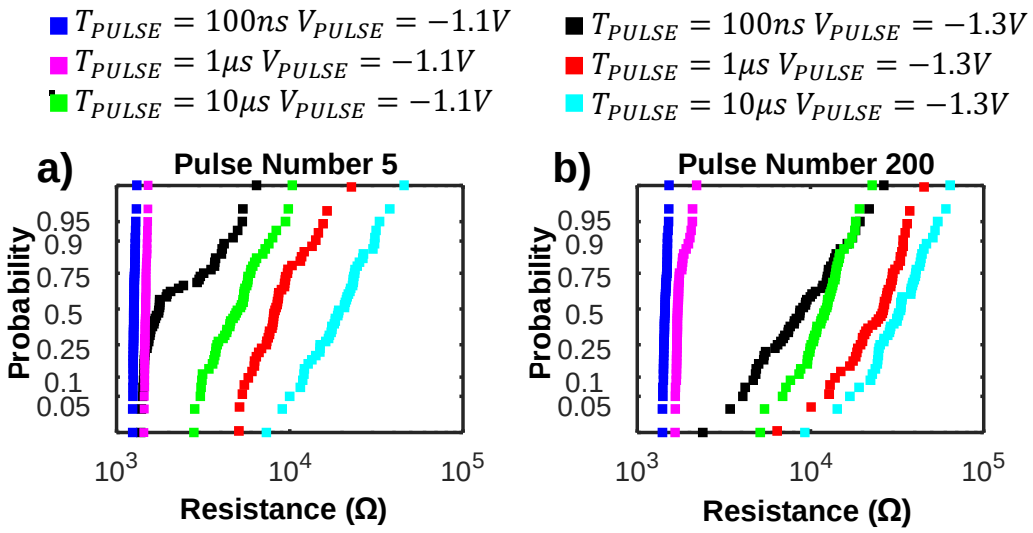


Figure 2.4: Probability plot showing the experimental resistance distributions after a) 5 and b) 200 pulses have been applied to the RRAM devices for different pulse widths (T_{PULSE}) and amplitudes (V_{PULSE}).

procedure described in Section 2.3. In general, the CF is heated when the barrier is not completely formed (i.e., low resistance values), thus higher CF temperatures are reached during the initial reset pulses and for shorter pulse widths. After the barrier is completely formed, the device temperature is governed by the barrier thermal characteristics. Exploiting these considerations, a linear relation between the CF and barrier temperatures and $\frac{\sigma_R}{R}$ was found, when considering the first ten reset pulses. As shown in Fig. 2.5a, $\frac{\sigma_R}{R}$ is linearly correlated to the value of the CF temperature time-derivative, averaged during a reset pulse (i.e., $\overline{\frac{dT_{CF}}{dt}}$), when the first ten reset pulses are considered. Instead, when considering the subsequent reset pulses, a linear relation between $\frac{\sigma_R}{R}$ and the value of the barrier temperature time-derivative, averaged during a reset pulse (i.e., $\overline{\frac{dT_{BAR}}{dt}}$) exists. The slopes of the two fitting lines in Fig. 2.5 (i.e., m_1, and m_2, in Fig. 2.5 a and b, respectively) are used as parameters of the comprehensive variability model for estimating the

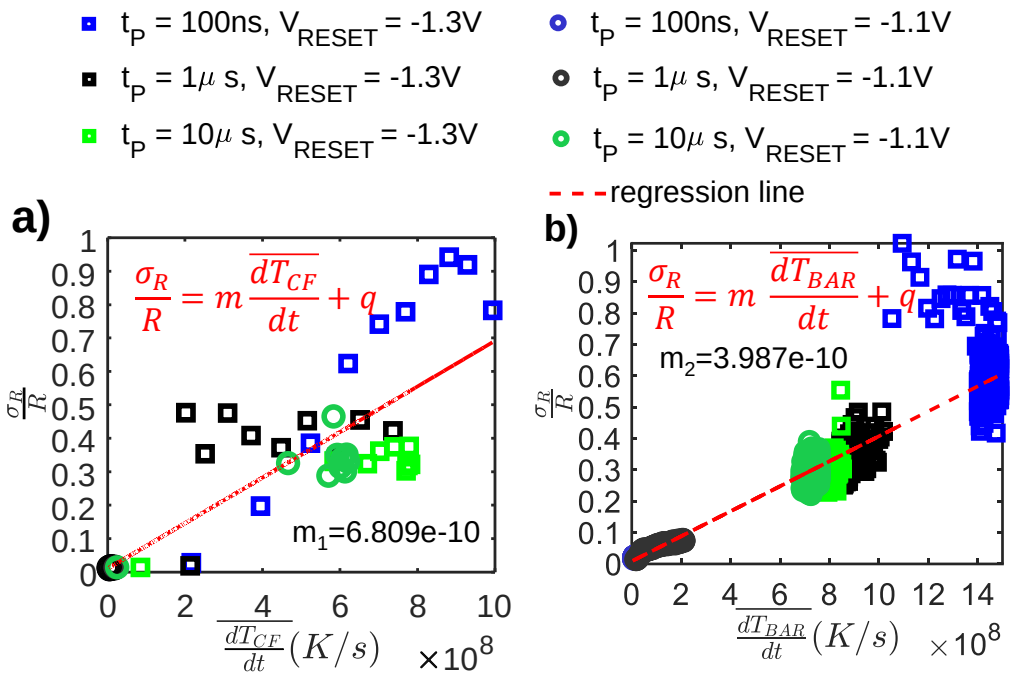


Figure 2.5: $\frac{\sigma_R}{R}$ as a function of a) $\overline{\frac{dT_{CF}}{dt}}$ and b) $\overline{\frac{dT_{BAR}}{dt}}$, considering different reset voltages and pulse widths. In a), and b), the first 10 reset pulses and the remaining 190 pulses of the experimental train of 200 reset pulses are considered, respectively. The slopes of the red fitting lines are reported.

value of $\frac{\sigma_R}{R}$, as shown in Equations (2.17), and (2.18), where S_P is a variable that is 1 when a negative voltage is applied to the device and 0 otherwise.

$$\frac{\sigma_R}{R}_1 = S_P \cdot \frac{dT_{CF}}{dt} \cdot m_1 \tag{2.17}$$

$$\frac{\sigma_R}{R}_2 = S_P \cdot \left(\frac{dT_{BAR}}{dt} + \left| \frac{dT_{BAR}}{dt} \right| \right) \cdot m_2 \tag{2.18}$$

The estimated $\frac{\sigma_R}{R}$ are then mapped to the standard deviation (i.e., σ_x) of the corresponding zero-mean Gaussian noise source introduced on the dielectric barrier thickness time derivative, as shown in Equations (2.19), and (2.20).

$$\sigma_{x1} = ln \left(\frac{\frac{\sigma_R}{R}_2 \cdot (R_{BAR} + R_{CF})}{R_{LRS} \cdot \beta \cdot e^{\frac{E_a}{k_b T_{BAR}}}} + 1 \right) \cdot l \tag{2.19}$$

$$\sigma_{x2} = ln \left(\frac{\frac{\sigma_R}{R}_1 \cdot R_{CF} \cdot S}{\rho \cdot [1 + \alpha \cdot (T_{CF} - T_0)]} \right) \tag{2.20}$$

Quasi-static IV HRS variability

With this modeling of C2C variability, it is possible to correctly reproduce the effect of variability for different reset voltages and pulse widths. However, the variability of RRAM devices in quasi-static IV for different reset voltages cannot be reproduced correctly. This is clearly shown in Fig. 2.6 and is due to the mathematical formulation. When quasi-static IV curves are considered these noise sources are not effective due to the slower variations on the device internal temperature, causing less than expected variations to be introduced on R_{HRS}. A solution is to introduce noise on a different parameter. A good candidate parameter whose variations have less impact on the numerical stability of the simulations is β, which maps x to a corresponding value of R_{BAR}. Thus introducing variations on β is equivalent to mimicking variations on the barrier composition. As shown in Fig. 2.6, noise sources introduced on the barrier time derivative have a negligible impact on the variability of the

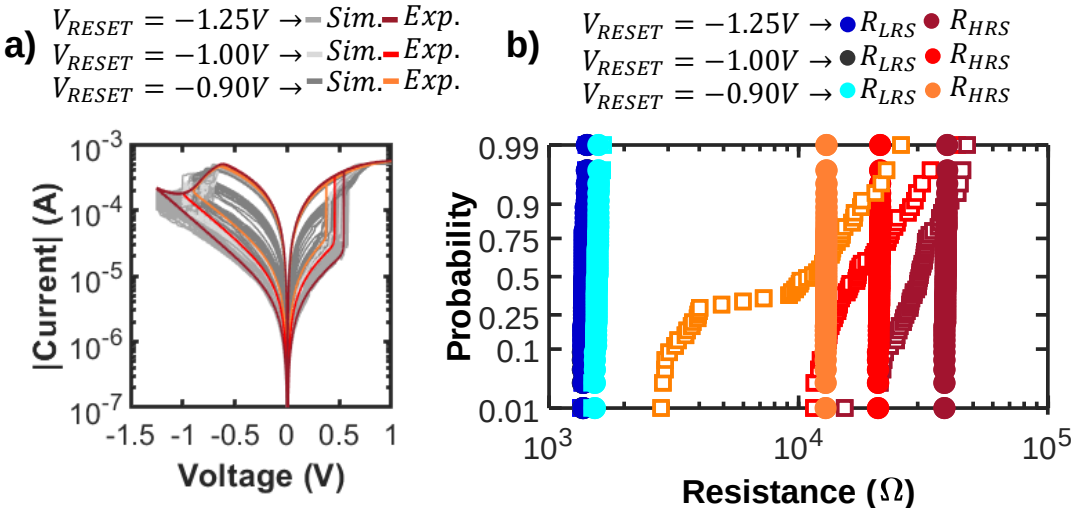


Figure 2.6: a) IV with variability without introducing noise on β (Sim.). b) Probability plot of the resistance (read at $V_{READ} = 100$ mV) LRS and HRS distributions, both experimental (squares) and simulated (circles). The introduced HRS variations are negligible.

quasi-static IV. Hence, the noise source on β and the noise sources on $\frac{dx}{dt}$ can be tuned separately to calibrate the model of the variability on experimental quasi-static IV and pulsed measurements, respectively. First the distribution of β is estimated from the experimental R_{LRS} and R_{HRS} distributions and Equation (2.21), where $\overline{S}$ and $\overline{x}$ are the estimated average CF cross-section and barrier thickness, respectively.

$$\beta = \frac{R_{BAR} \cdot \overline{S}}{\rho \cdot t_{OX} \cdot (e^{\frac{\overline{x}}{l}} - 1) \cdot e^{\frac{E_T}{k_B \cdot T_{BAR}}}} \tag{2.21}$$

β follows the logNormal distribution, and as shown in Fig. 2.7a, σ_β, which is the standard deviation of the Gaussian random variable (i.e., $Y \sim \mathcal{N}(0,1)$) associated to the logNormal distribution of β, changes with V_{RESET}, with larger values of σ_β for higher V_{RESET}s. Still, due to the effect of the differential equations, the resulting variability on R_{HRS} is typically lower than expected when considering lower reset voltages. Thus, the σ_β vs V_{RESET} relation needs to be manually adjusted, as shown in Fig. 2.7b. Such rela-

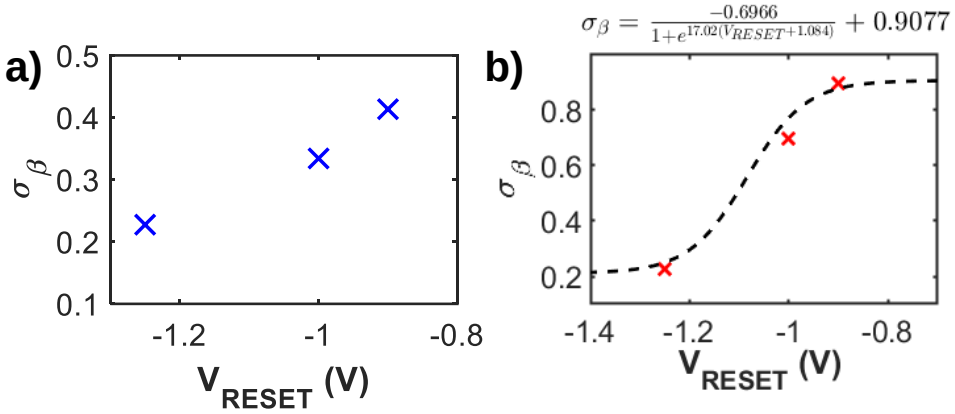


Figure 2.7: a) σ_β of the log-normally distributed β as a function of V_{RESET} that is extracted from experimental data using Equation (2.21). b) Adjusted σ_β vs V_{RESET} (red crosses) and corresponding empirical fitting function that is used to update the value of σ_β with respect to V_{RESET}. The values of the calibrated parameters a, b, c, and d of Equation (2.22) are reported.

tion is then fitted with Equation (2.22), where a, b, c, and d are calibrated with optimization algorithms. Equation (2.22) is used because σ_β needs to saturate for high negative V_{RESET} voltages.

$$\sigma_\beta = \frac{a}{1 + e^{b \cdot (V_{TB_{min}} - c)}} + d \tag{2.22}$$

In the Verilog-A compact model, the value of β is updated with Equation (2.23) at each set event. Updating the value of β during a set event is preferable to avoid convergence issues during reset. At each reset event, using a peak detector circuit the value of the reset voltage is stored in an internal variable of the model (i.e., $V_{TB_{min}}$), which is used to update the value of σ_β.

$$\beta_k = e^{ln(\overline{\beta}) - ln(\sqrt{e^{\sigma_\beta}}) + \sigma_\beta \cdot Y} \tag{2.23}$$

2.2.2 Implementation following the best practices

In circuit simulators, the Verilog-A code implementing the compact model can influence the simulation convergence and speed. For this reason, Verilog-A programming best practices have been developed over the years [98], [113]. However, in RRAM compact models available in the literature such best practices are often overlooked. A common deficiency can be found in the implementation of the system of differential equation that is commonly solved using integral formulations based on the `idt()` operator, instead of the differential formulation that is based on the `ddt()` operator. This approach is detrimental for the circuit simulator, and compromises its convergence, and limits the model's ability to support all analyses. Since circuit simulators are nonlinear, coupled differential algebraic equations solvers, the differential formulation should always be preferred. Thus, in the UniMORE compact model, the differential formulation is implemented. Also, whenever possible, the system of equations of the compact model is implemented with voltage-controlled current elements (i.e., $I(V)$) and the time derivative of voltage-controlled charge elements (i.e., $\frac{dq(V)}{dt}$). As already mentioned, the abrupt nature of the set event in RRAM devices results in a stiff mathematical problem. To limit the simulator convergence issues, setting a fixed simulation time step should be avoided. Also, due to the stiffness of the mathematical problem, exponential function can potentially reach unrealistically high values, thus the `limexp()` function is used in place of the `exp()`, limiting convergence issues. Furthermore, numerical errors that can result in negative or larger than t_{OX} barriers should be prevented. Thus the approach proposed in [113] is employed, where the additional terms in Equations (2.25), (2.26), and (2.27) are added to $\frac{dx}{dt}$. t_A intervenes when x is approaching either 0 or t_{OX} and sets $\frac{dx}{dt}$ to 0. t_B and t_C increase or decrease x when $x < 0$

or $x > t_{OX}$, respectively. When x is in the correct range (i.e., from 0 nm to t_{OX}) these terms are zero and their effect on $\frac{dx}{dt}$ is negligible.

$$smoothing_2(x, smooth) = 0.5\left(1 + \frac{x}{\sqrt{x^2 + smooth}}\right) \tag{2.24}$$

$$t_A = -\frac{dx}{dt}\cdot[smoothing_2(-x, smooth) + smoothing_2(x - t_{OX}, smooth)] \tag{2.25}$$

$$t_B = safeexp\left(10^4 \cdot (-x), max_{\frac{dx}{dt}}\right) \tag{2.26}$$

$$t_C = safeexp\left(10^4 \cdot (x - t_{OX}), max_{\frac{dx}{dt}}\right) \tag{2.27}$$

As suggested in [98], conditional statements are implemented with the numerically robust smoothing function (see Equation (2.28)) when required while conditionals based on parameters values are introduced to avoid evaluating parts of the model when they should have no effects (e.g., activate the RTN module only when `RTN_FLAG` is 1).

$$smoothing(x, th, smooth) = \frac{1}{1 + e^{\frac{th - x}{smooth}}} \tag{2.28}$$

In RRAM compact models, multiple physical quantities, such as currents, temperatures, and length are included in the model. These quantities can assume largely different values concerning currents and voltages, thus potentially leading to convergence failures if not correctly defined. A good programming practice is to define a different Verilog-A discipline for each quantity, setting appropriate tolerances and convergence criteria. In the UniMORE compact model, additional disciplines for the temperature and barrier thickness are defined, as shown in Appendix II.

2.3 Automated parameter extraction procedure

Here, follows a detailed description of the developed automated parameter extraction procedure. The set of experimental measurements that are

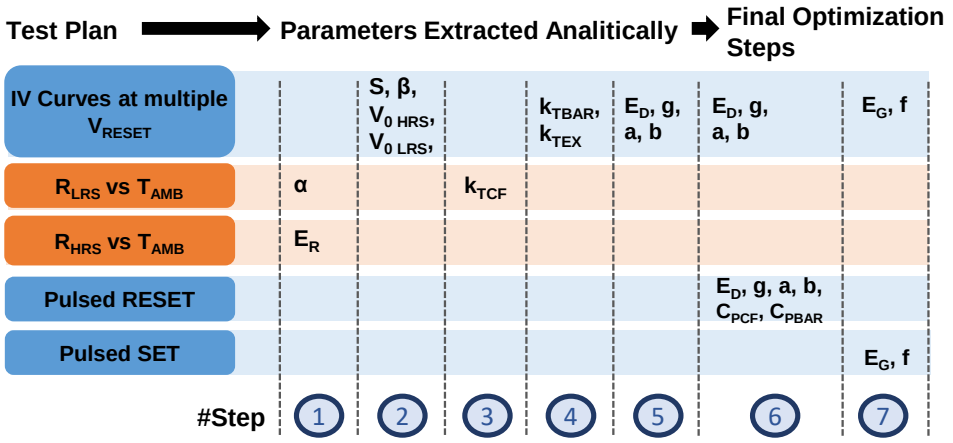


Figure 2.8: Flow chart showing the set of experiments performed to extract the model parameters. The sequence of steps required to extract all the parameters is reported. For each step, the list of the extracted parameters is aligned with the experimental data used in the extraction procedure. When the same parameter appears in multiple lines, it indicates that the data from multiple experiment is used during the extraction procedure (e.g., at steps 6 and 7).

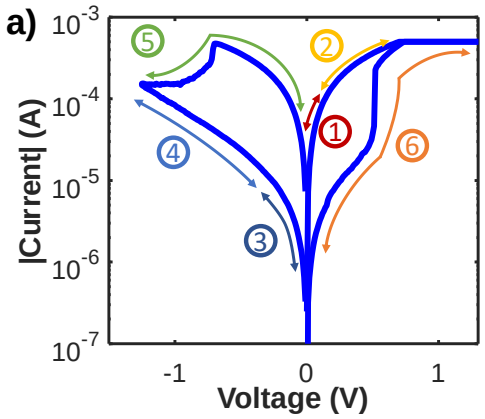
required for complete model parameters calibration is outlined. Finally, a simplified parameter extraction procedure that is suitable to calibrate the model when only limited data is available is discussed and verified on three RRAM technologies from the literature.

2.3.1 Design of experiments

To extract the parameters of the compact model from experimental data, the list of experiments shown in Fig. 2.8 needs to be performed. Such experiments require only instrumentation that is commonly available in laboratories performing device characterization. As reported in Fig. 2.8, each experiment is exploited to extract different model parameters. Specifically, most of the model parameters can be extracted from the measured IV at multiple reset voltages (i.e., V_{RESET}). The extraction of parameters modeling the device

behavior with temperature, requires measurements to be performed at different temperatures (i.e., R_{LRS} vs T_{AMB}, R_{HRS} vs T_{AMB}, see Fig. 2.8). Finally, measurements in which the RRAM devices are stimulated with voltage pulses are needed to calibrate and optimize the model parameters influencing the barrier and temperatures dynamics.

To evaluate the effectiveness of the proposed parameter extraction procedure, we experimentally collected data on $TiN/Ti/HfO_x/TiN$ devices from Sematech, using a Keithley 4200-SCS semiconductor parameter analyzer and a temperature-controlled thermal chuck. In all the experiments a current compliance of $500\mu A$ was used. IV curves were collected for three different reset voltages (i.e., V_{RESET} equal to -0.9 V, -1 V, and -1.25 V) and a slew-rate of $0.0911V/s$. Such a slow slew rate is required to simplify the parameter extraction, as described in the next section. The HRS and LRS resistance was evaluated at $100mV$ and temperature ranging from $30°C$ to $50°C$ and from $30°C$ to $90°C$, respectively. For a device in LRS, the IV characteristic was also measured at a temperature ranging from $30°C$ to $50°C$, applying a voltage ramp from 0 to $0.6V$ with a slew rate of $1mV/s$. Finally, to evaluate the device response for reset voltage pulses, the RRAM device was first programmed in LRS then trains of 200 voltage pulses were applied. The device resistance was measured with 1 ms long, 50 mV read voltage pulses. The experiment was repeated for different reset voltages (i.e., V_{RESET} equal to -1.1 V, and -1.3 V) and pulse widths (i.e., T_{PULSE} equal to $10\mu s$, $1\mu s$, $100ns$). Also, the abrupt behavior of the set operation was verified after resetting the device with a V_{RESET} of -1.25V, and connecting a series 680Ω resistor, using set voltage pulses with $1V$ amplitudes and $1\mu s$ duration. Every experiment was replicated at least 50 times to include the effect of C2C variations.



	Conditions	Extracted Param.
1	$\Delta T_{cf} \approx 0$	S, $V_{0\ LRS}$
1 + 2	$x \approx 0$	k_{Tcf}
6		E_G, f
3	$\Delta T_b \approx 0$	β , $V_{0\ HRS}$
3 + 4	$\Delta x \approx 0$	k_{Tb}
5		E_D, g, a, b

Figure 2.9: a) RRAM IV curve divided in segments, numbered from 1 to 6, where different approximations can be assumed. b) Table mapping the considered approximation in each segment and the resulting set of extracted parameters.

2.3.2 Data processing

Thanks to the physics-based compact modeling, some of the device parameters that are specific to the material stack used to fabricate the device can be extracted from technological information and materials repositories in the literature. For instance the oxide layer thickness (i.e., t_{OX}) is a fabrication parameter while values for the CF resistivity (i.e., ρ) and the typical tunneling length (i.e., k) can be taken from the literature [114]. The remaining parameters can be extracted from the experimental measurements described in Section 2.3.1. Here we describe an automated calibration procedure that enables the extraction of most of the parameters analytically. A few other parameters are optimized using common optimization algorithms. This procedure addresses a critical limitation of current physics-based compact models in the literature, which lack a clear parameter extraction procedure, see Table 2.1, therefore limiting their usability. To calibrate such models only strategies that require manually changing each parameter value, and thus high expertise and a lot of time, have been proposed [78], [106], [123]. First,

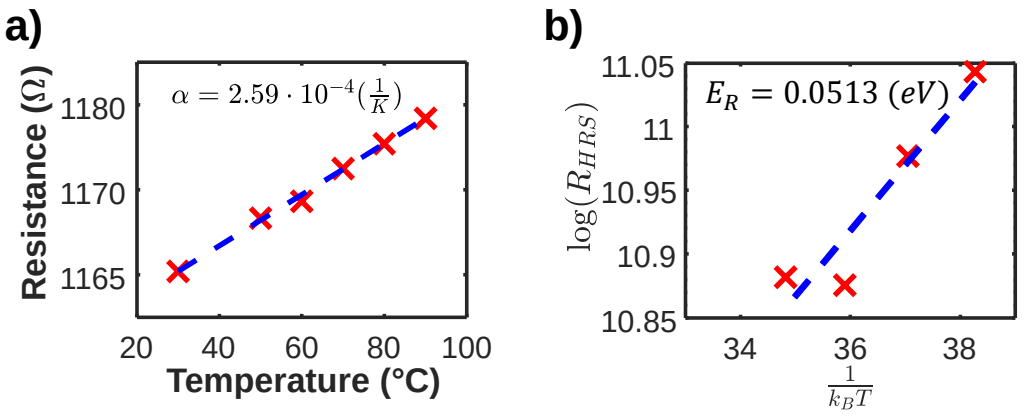


Figure 2.10: a), b) Experimentally measured data (symbols) and fitting lines, from which the parameters α, and E_R are extracted, respectively.

parameters α and E_r, which model the CF and barrier resistance variations with temperature [114], [124], are estimated using the measured R_{LRS} and R_{HRS} at different ambient temperatures. Such parameters are estimated as the slopes of the regression lines in Fig. 2.10a and b, respectively. To estimate the other parameters, the quasi-static IV curves are exploited. As shown in Fig. 2.9, the IV curve can be split into different segments in which different approximations are considered to simplify the mathematical problem. For instance, in segment 1 (i.e., the red segment in Fig. 2.9) the barrier and CF temperature variations due to self-heating are considered to be zero since small voltages are applied to a device in LRS. Segment 1 is used in the second step of the parameter extraction procedure to determine the $\frac{\rho}{S}$ ratio. Since t_{OX} is a technology parameter and a value of ρ can be taken from the literature (i.e., between $400\Omega nm$ and $10^4\Omega nm$ for HfO_x based devices [125]), the nominal CF cross-section is determined by means of a linear regression as shown in Fig. 2.11a. The current conduction in the HRS is nonlinear and commonly related to TAT mechanism that is commonly modeled considering the Simmons' tunneling barrier model [116] which is based on the hyperbolic

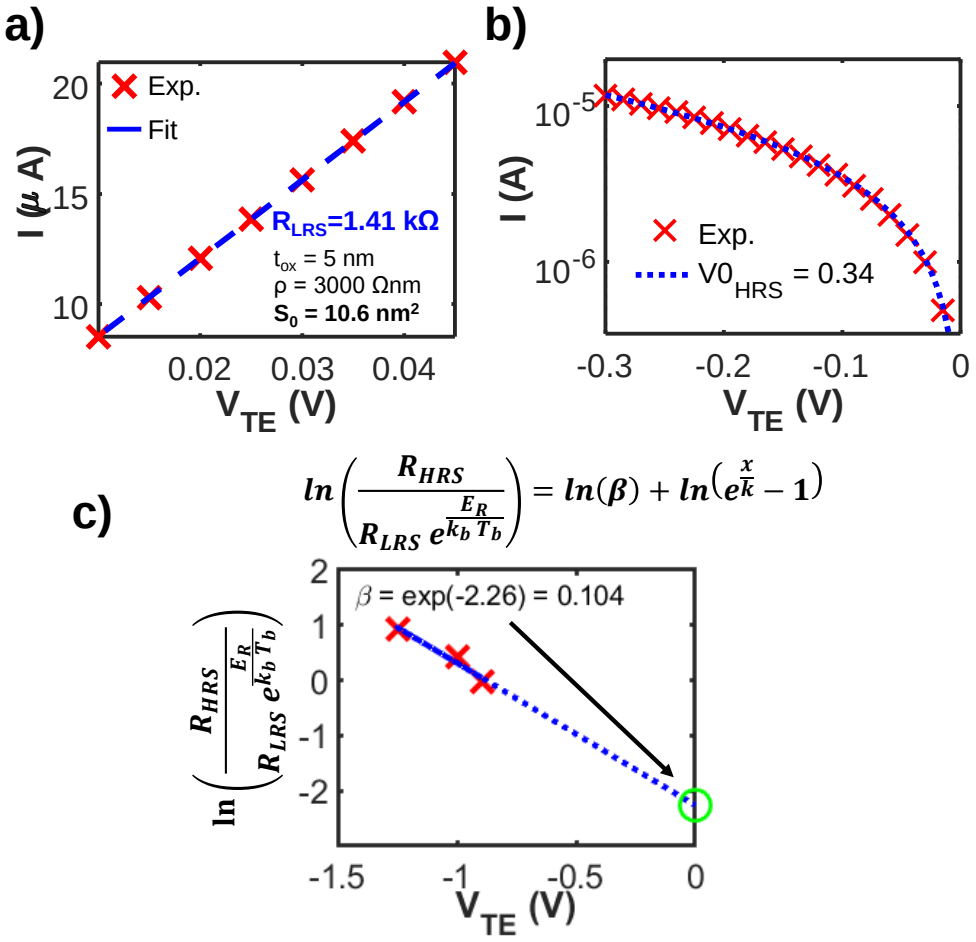


Figure 2.11: a), b), c) Experimentally measured data (symbols) and fitting lines, from which the parameters S_0, $V_{0_{HRS}}$, and β are extracted, respectively.

sine function (i.e., *sinh*) and a nonlinearity parameter [97] (i.e., $V_{0_{HRS}}$ in this model). The conduction in LRS can also show some nonlinearity due to the imperfect metallic-like CF consisting of oxygen vacancies, and so also in this case it can be modeled using the *sinh* function and a nonlinearity parameter $V_{0_{LRS}}$. $V_{0_{LRS}}$ is always higher than $V_{0_{HRS}}$ since the degree of nonlinearity in LRS is lower than in HRS. Both the nonlinearity parameters $V_{0_{HRS}}$ and $V_{0_{LRS}}$ can be extracted by means of nonlinear solvers (e.g., such as `MATLAB®` `fsolve` function) considering the portion of the I/V relation where the effect of self-heating is negligible for a device in HRS and LRS, respectively, (see segments 3 and 1 in Fig. 2.9). The resulting $V_{0_{HRS}}$ is shown in Fig. 2.11. Next, the data of I/V curves at different V_{RESET} can be used to estimate β, that is the parameter modeling the relation between the device resistance in HRS and the barrier thickness. Using the values of R_{HRS} measured at different V_{RESET}s, and approximating the total device resistance with R_{HRS}, Equation (2.4) can be manipulated so that β is estimated by a simple linear regression, as the point at which the regression line intersects the y-axis at $V_{TE} = 0$, as shown in Fig. 2.11c. The CF thermal conductance (i.e., $k_{T_{CF}}$) is estimated using the measured IV at different ambient temperatures for a device in LRS discussed in the previous section, which corresponds to the segments 1 and 2 in Fig. 2.9. In this region of the IV, the barrier is zero, and by exploiting the quasi-static approximation, it is assumed that the steady-state CF temperature is reached for each applied voltage, meaning that $\frac{dT_{CF}}{dt} \approx 0$. Using these assumptions and by substituting, the expression for T_{CF} from Equation (2.5) in Equation (2.11) the expression reported in Fig. 2.12 is obtained, and thus $k_{T_{CF}}$ is estimated as the slope of the regression line fitting the dissipated power with respect to the temperature difference between the CF and the ambient. With the estimated value for $k_{T_{CF}}$, the thermal con-

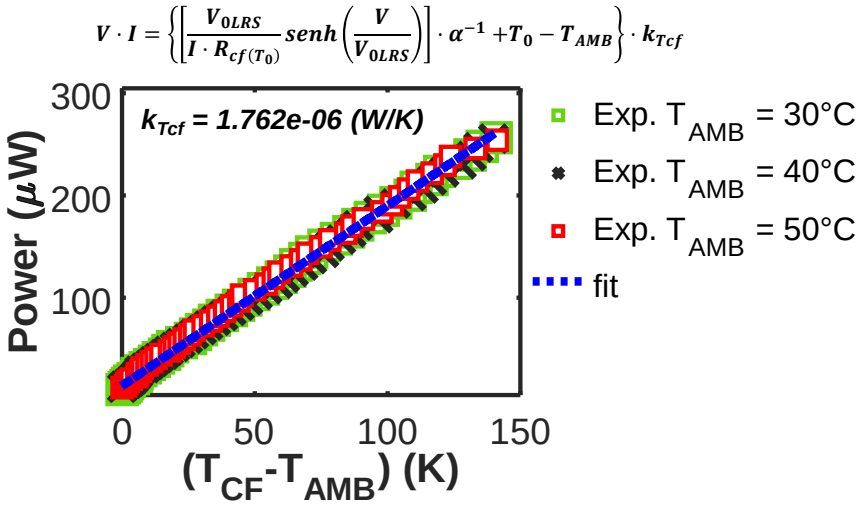


Figure 2.12: Experimentally measured data (symbols) and fitting lines, from which the parameters $k_{T_{CF}}$ is extracted.

ductivity of the dielectric barrier (i.e., $k_{T_{BAR}}$) can be estimated by solving the nonlinear System of Equations (2.29), by means of a nonlinear solver (e.g., `MATLAB®` `fsolve` function) and of an optimization algorithm (e.g., particle swarm algorithm). The system unknowns (i.e., T_{CF}, T_{BAR}, V_{CF}, V_{BAR}, and $k_{T_{BAR}}$) are solved by using as inputs the IV segments 3 and 4 for different reset voltages, and considering the barrier thickness constant, and imposing that temperature cannot decrease when sweeping the voltage from zero to V_{RESET}. The optimization algorithm searches the optimal value for $k_{T_{BAR}}$, while the nonlinear system solver is used to compute the cost function shown

in Equation 2.30.

$$\begin{cases} I - \frac{V_{0_{HRS}} sinh\left(\frac{V_{BAR}}{V_{0_{HRS}}}\right)}{R_{LRS}\beta\left(e^{\frac{\overline{x}}{k}}-1\right)e^{\frac{E_r}{k_B T_{BAR}}}} = 0 \\ I - \frac{V_{0_{LRS}} sinh\left(\frac{V_{CF}}{V_{0_{LRS}}}\right)}{\rho\frac{t_{OX}-x}{S}[1+\alpha(T_{CF}-T_0)]} = 0 \\ I \cdot V_{BAR} - k_{T_{BAR}}(T_{BAR} - T_{AMB}) = 0 \\ I \cdot V_{CF} - k_{T_{BAR}}(T_{CF} - T_{AMB}) = 0 \\ V - V_{BAR} - V_{CF} = 0 \\ 1 - tanh\,(T_{BAR} - T_{AMB}) = 0 \end{cases} \tag{2.29}$$

$$\epsilon = \frac{1}{\#V_{RESET}} \sum_{j=1}^{\#V_{RESET}} \frac{1}{n_j} \sum_{i=1}^{n} \left[I_{exp_i} - \frac{V_{0_{HRS}}}{R_{BAR_j}} sinh\left(\frac{V_{BAR_{ij}}}{V_{0_{HRS}}}\right)\right] \tag{2.30}$$

In the thermal Equations (2.10), and (2.11) of the compact model the additional parameter $k_{T_{EX}}$ can be used to consider the heat exchange between the barrier and the CF. As a first approximation, $k_{T_{EX}}$ can be set to zero, but for some RRAM technologies, a more accurate calibration is achieved when optimizing for all three thermal conductance. Only few variations are required to the system of equation (2.29), to include also $k_{T_{EX}}$, as shown in Equation (2.31). The main difference is in the cost function of the optimization algorithm, see Equation (2.32). The optimization algorithm optimizes at the same time the values of $k_{T_{CF}}$, $k_{T_{BAR}}$, and $k_{T_{EX}}$, considering as a constrain the value of $k_{T_{CF}}$ extracted in the previous step, that however, with

the inclusion of $k_{T_{EX}}$, corresponds to the difference $k_{T_{CF}} - k_{T_{EX}}$.

$$\begin{cases} I - \frac{V_{0_{HRS}} sinh\left(\frac{V_{BAR}}{V_{0_{HRS}}}\right)}{R_{LRS}\beta\left(e^{\frac{\overline{x}}{k}}-1\right)e^{\frac{E_r}{k_B T_{BAR}}}} = 0 \\ I - \frac{V_{0_{LRS}} sinh\left(\frac{V_{CF}}{V_{0_{LRS}}}\right)}{\rho\frac{t_{OX}-x}{S}[1+\alpha(T_{CF}-T_0)]} = 0 \\ I \cdot V_{BAR} - k_{T_{BAR}}(T_{BAR} - T_{AMB}) - k_{T_{EX}}(T_{CF} - T_{BAR}) = 0 \\ I \cdot V_{CF} - k_{T_{BAR}}(T_{CF} - T_{AMB}) - k_{T_{EX}}(T_{BAR} - T_{CF}) = 0 \\ V - V_{BAR} - V_{CF} = 0 \\ 1 - tanh\,(T_{BAR} - T_{AMB}) = 0 \end{cases} \tag{2.31}$$

$$\begin{aligned} \epsilon = \frac{1}{\#V_{RESET}} \sum_{j=1}^{\#V_{RESET}} \frac{1}{n_j} \sum_{i=1}^{n} \left[I_{exp_i} - \frac{V_{0_{HRS}}}{R_{BAR_j}} sinh\left(\frac{V_{BAR_{ij}}}{V_{0_{HRS}}}\right)\right] + ... \\ ... + \left| (k_{T_{CF}} - k_{T_{EX}}) - \widehat{k_{T_{CF}}} - \widehat{k_{T_{EX}}} \right| \end{aligned} \tag{2.32}$$

2.3.3 Parameter optimization

The remaining parameters that need to be calibrated are the parameters connected to the differential equations in the compact model which influence the device set/reset and dynamic temperature variations. Optimizing at the same time all the parameters without a starting point close to the best solution can result in a time-consuming and more complex problem. A solution is to split the optimization problem into three steps.

In the first step, the optimization algorithm (i.e., particle swarm) extracts the best solution for the parameters influencing the device reset when the quasi-static approximation can be used. Specifically, the reset portion of the quasi-static IV characteristic measured at various V_{RESET} (i.e., segments 3, 4, and 5 in Fig. 2.9) is input to a particle swarm algorithm which optimizes the parameters E_{AD}, g, a, and b. It is important to note, that the appropriate boundary needs to be set on E_D, which is a parameter connected to the device physics and should be between $1eV$ and $2eV$ [123], [126]. The cost function

is not smooth and presents several local minima, thus it is important to restrict the solution to a range that is physically plausible. The cost function for the optimization algorithm is based on the Mean Absolute Percentage Error (MAPE), as shown in Equation (2.33).

The estimated E_D, g, a, b, are used as the initial point for the subsequent optimization step in which also $C_{P_{CF}}$, and $C_{P_{BAR}}$ are optimized. The thermal capacitance influences the pulsed reset characteristic and thus the measurement of trains of reset pulses at different reset voltages and pulse widths are used to compute a new cost function. This new cost function consider both the MAPE computed on the quasi-static IV and pulsed characteristics, that correspond to Equations (2.33), and (2.34), respectively. These two errors and the absolute value of their difference are added together to assemble the complete cost function, as shown in Equation (2.35). The latter term is needed to balance the error between the quasi-static IV and pulsed characteristics, avoiding overfitting only one characteristic while showing poor performance on the other.

$$cost_1 = \frac{1}{\#V_{RESET}} \sum_{j=1}^{\#V_{RESET}} \frac{1}{n_j} \sum_{i=1}^{n} \left| \frac{I_{sim_{ij}} - I_{meas_{ij}}}{I_{meas_{ij}}} \right| \tag{2.33}$$

$$cost_2 = \frac{1}{\#t_{PULSE}} \sum_{k=1}^{\#t_{PULSE}} \frac{1}{\#V_{RESET}} \sum_{j=1}^{\#V_{RESET}} \frac{1}{n_j} \sum_{i=1}^{n} \left| \frac{\frac{R}{R_{LRS}}_{sim_{ij}} - \frac{R}{R_{LRS}}_{meas_{ij}}}{\frac{R}{R_{LRS}}_{meas_{ij}}} \right| \tag{2.34}$$

$$cost = cost_1 + cost_2 + |cost_1 - cost_2| \tag{2.35}$$

After calibrating the parameter modeling the device reset and temperature characteristics, the parameters influencing the set operation (i.e., E_G, g) can be adjusted to reproduce the correct set voltage when considering the IV characteristic and the set speed when considering the pulse operation. The final calibration of the median characteristic of the RRAM device is shown

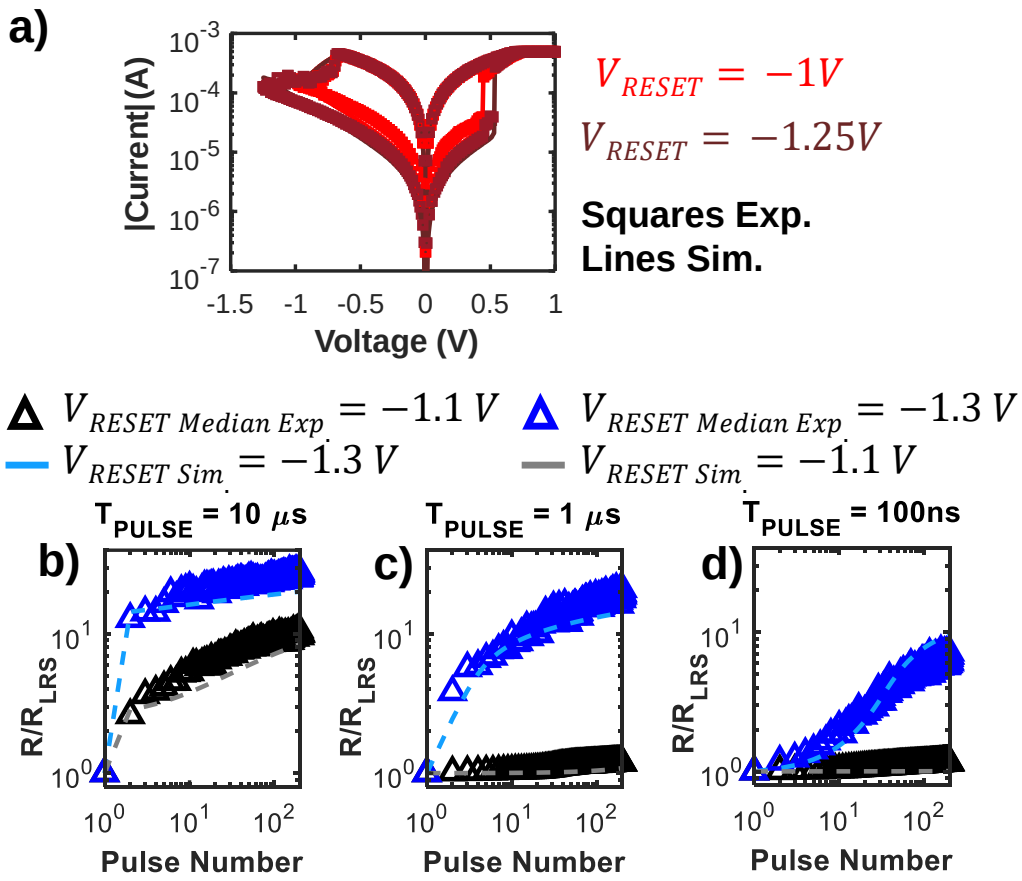


Figure 2.13: Experimental (symbols) and simulated (lines) a) quasi-static DC IV, b), c), and d) pulsed reset characteristic for different reset voltages (V_{RESET}) and pulsed widths (T_{PULSE}).

in Fig. 2.13. Finally, to include the effect of variability, the parameters of the comprehensive variability model described in Section 2.2.1 need to be determined. Specifically, the following steps need to be followed:

- Extract σ_S from the experimental R_{LRS} distribution
- Extract $\sigma_\beta(V_{RESET})$ from the IV experimental data
- Manually adjust σ_β values for low V_{RESET} to reproduce the R_{HRS} variability
- Extract σ_{x1} and σ_{x1} (see Equations (2.20), (2.19)) from the pulsed reset experimental data

As a result, both the variability for the quasi-static IV and the pulsed reset characteristics are well reproduced by the compact model, as shown in Fig. 2.14 and Fig. 2.15, 2.16, respectively.

2.3.4 Simplified parameter extraction procedure from data from the literature

The parameters of the compact model can also be calibrated on data of RRAM technologies available in the literature. However, the complete set of experimental data that is described in the previous sections is commonly not reported in scientific papers. Nevertheless, the parameters of the compact model can still be calibrated if at least the data for the IV and pulsed characteristics, preferably at multiple V_{RESET}, are available. Still, the parameter extraction procedure previously described needs to be simplified. In the following, this simplified parameter extraction procedure is described and tested on three RRAM technologies from the literature, specifically a $Pt/TiO_x/HfO_x/TiO_x/HfO_x/TiN$ RRAM [127], a $TiN/HfO_2/Ti/TiN$

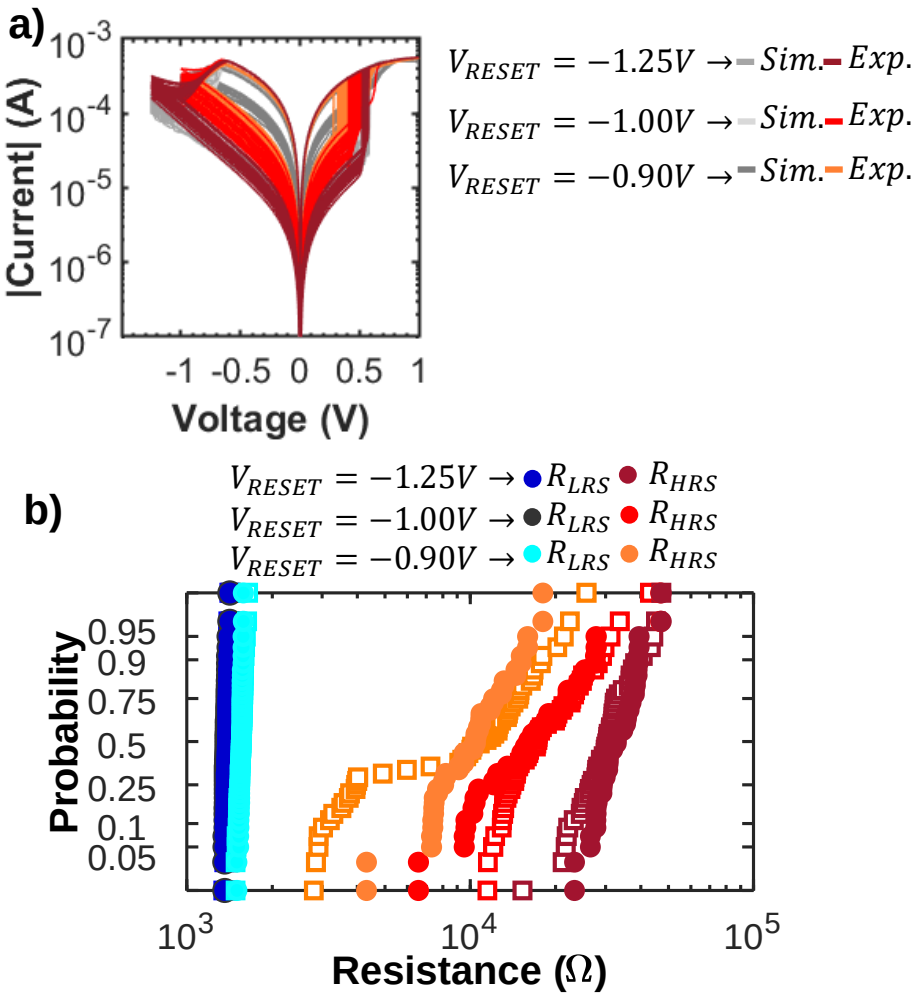


Figure 2.14: a) IV with variability (Sim.). b) Probability plot of the resistance (read at V_{READ} = 100 mV) distribution, both experimental and simulated.

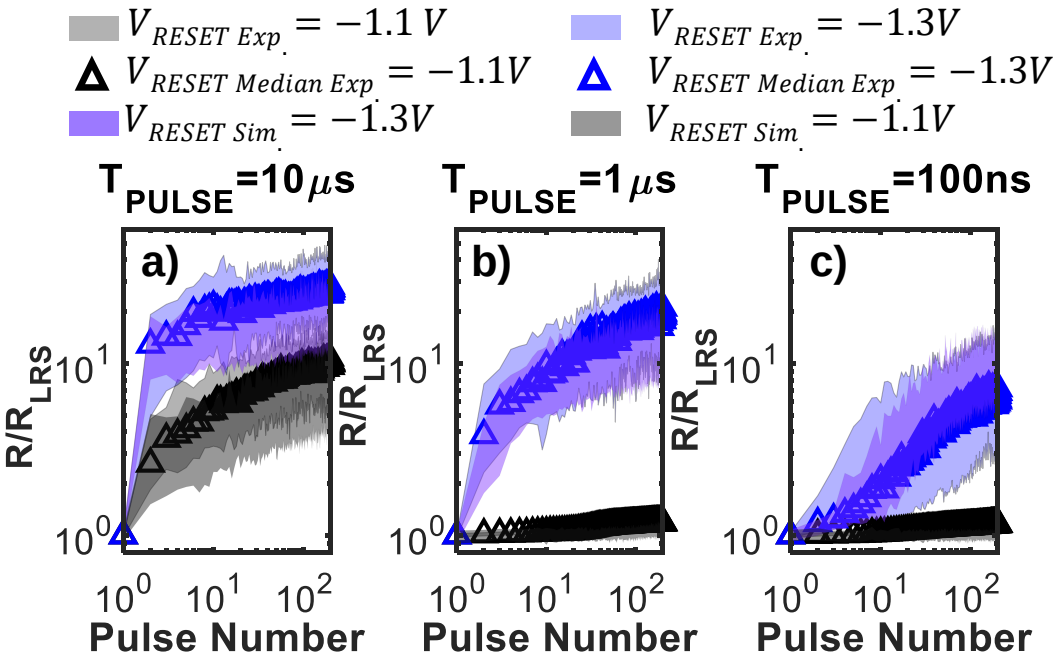


Figure 2.15: a), b), c) Experimental (symbols and blue and gray shaded regions) and simulated (dark violet and black regions) pulsed reset considering the effect of C2C variability, decreasing pulse widths (T_{PULSE}).

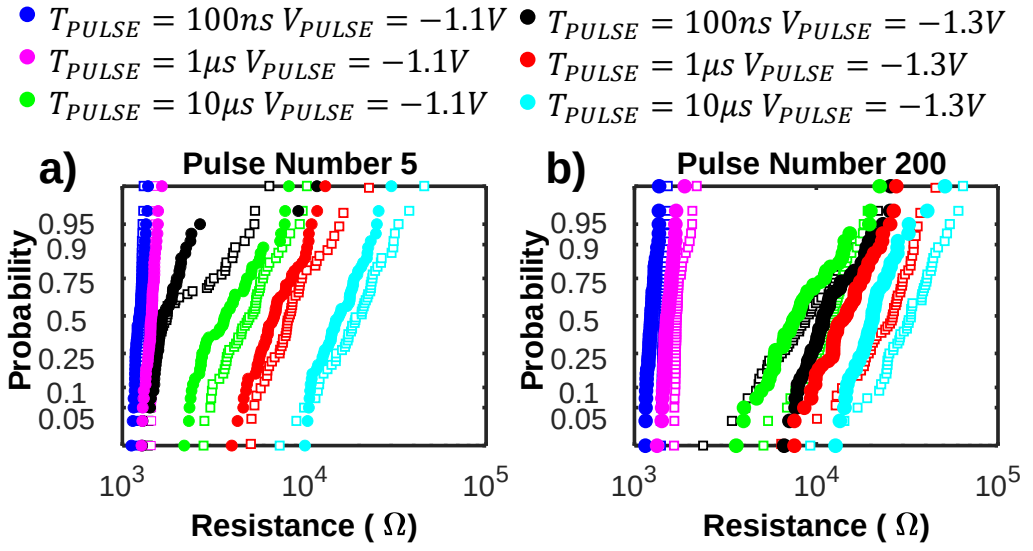


Figure 2.16: a), b) Probability plot showing the experimental (empty squares) and simulated (full circles) resistance distribution after a) 5 and b) 200 pulses have been applied to the RRAM devices for different pulse widths (T_{PULSE}) and amplitudes (V_{PULSE}).

RRAM [128], and a $TiN/HfO_x/AlO_x/Pt$ RRAM [129], which are referred to as technology 1, technology 2, and technology 3 in this thesis, respectively.

To compensate for missing information regarding the device characteristic for temperature variations more parameters need to be extracted from the literature or material repositories. Thus, in addition to t_{OX}, and ρ, also α and E_r, should be taken from the literature. Since, all the technologies share the same oxide material, the same values for α and E_r can be used. Specifically, considering the data reported [114], [124] an α of $2.541/K$ and an E_r of $0.12eV$ are used for all three technologies. The values of S, $V_{0_{HRS}}$, $V_{0_{LRS}}$, and β can be extracted from the IV characteristic measured at different $V_{RESET}s$ as described in Section 2.3.2. Also, $k_{T_{CF}}$ or $k_{T_{CF}} - k_{T_{EX}}$ can be extracted from the IV curve segments 1 and 2 in Fig. 2.9, however exploiting the data available only at a single ambient temperature (i.e., T_{AMB}), thus resulting in possible small accuracy drops. The rest of the model parameters (i.e., $k_{T_{BAR}}$, $k_{T_{EX}}$, E_D, g, a, b, $C_{P_{CF}}$, and $C_{P_{BAR}}$) can be extracted following the same methods described in sections 2.3.2, and 2.3.3, and the resulting calibrated compact model for the three technologies from the literature are shown in Fig. 2.17. Finally, the simplified variability module parameters described in 2.2.1 can be calibrated to reproduce the variability in the experimental conditions, following these steps:

- Extract σ_S from the experimental R_{LRS} distribution
- Extract σ_x from the experimental R_{BAR} distribution

As shown in Fig. 2.18 both the experimental HRS and LRS distributions [128], [129], [131] are well reproduced with the compact model for the three RRAM technologies. For technology 2, the variability data [128] are available only at a lower I_C (i.e., $20\mu A$ instead of $200\mu A$). Still, thanks to the physics-

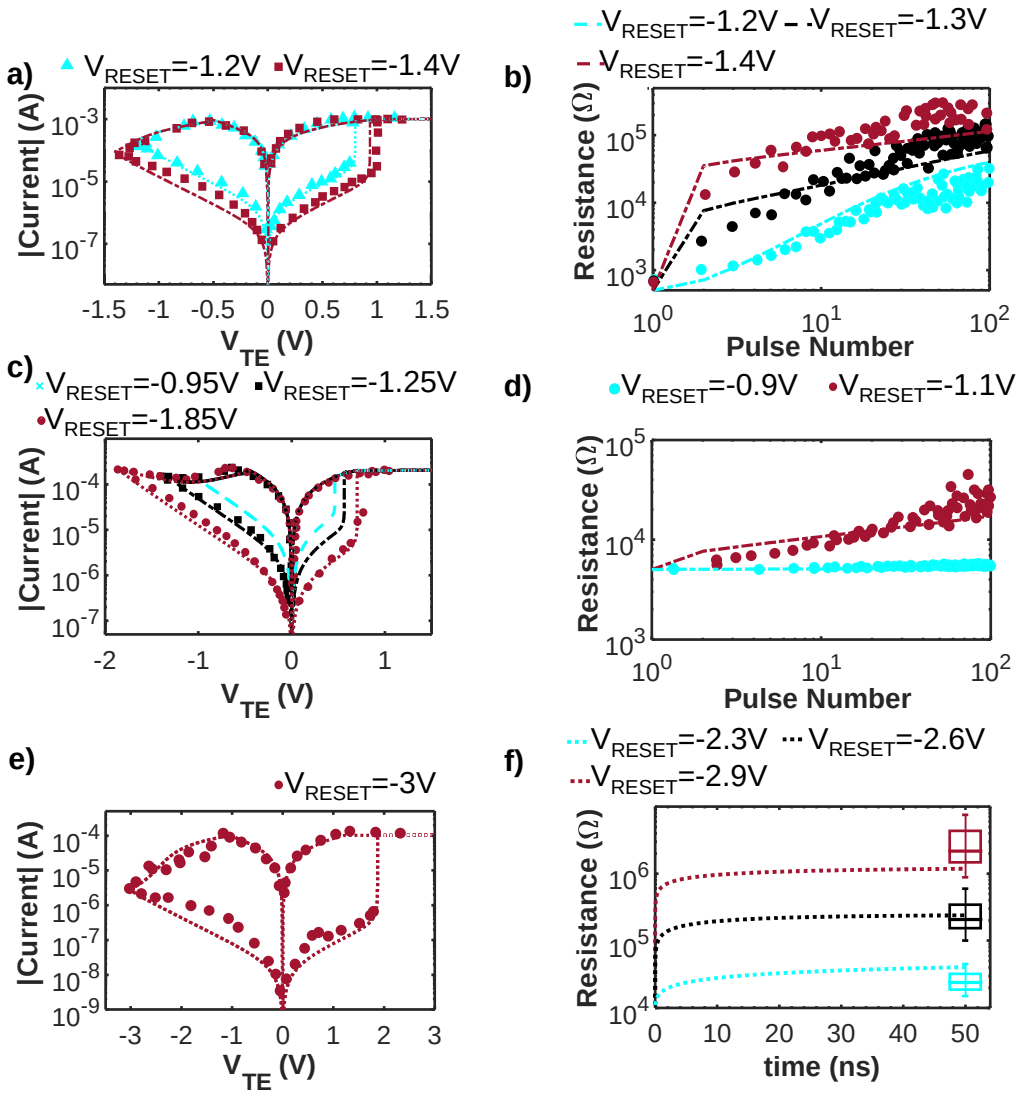


Figure 2.17: Calibrated model on three RRAM technologies from the literature. a), c), e) Simulated (lines) and experimental (symbols) quasi-static DC IV curve at different V_{RESET}, when possible. b), d), f) Simulated (lines) and experimental (symbols) response to fast reset pulses for different V_{RESET}. a), b) Refer to a $Pt/TiO_x/HfO_x/TiO_x/HfO_x/TiN$ RRAM, data from [127]. c), d) Refer to a $TiN/HfO_2/Ti/TiN$ RRAM, data from [130]. e), f) Refer to a $TiN/HfO_x/AlO_x/Pt$ RRAM, data from [129].

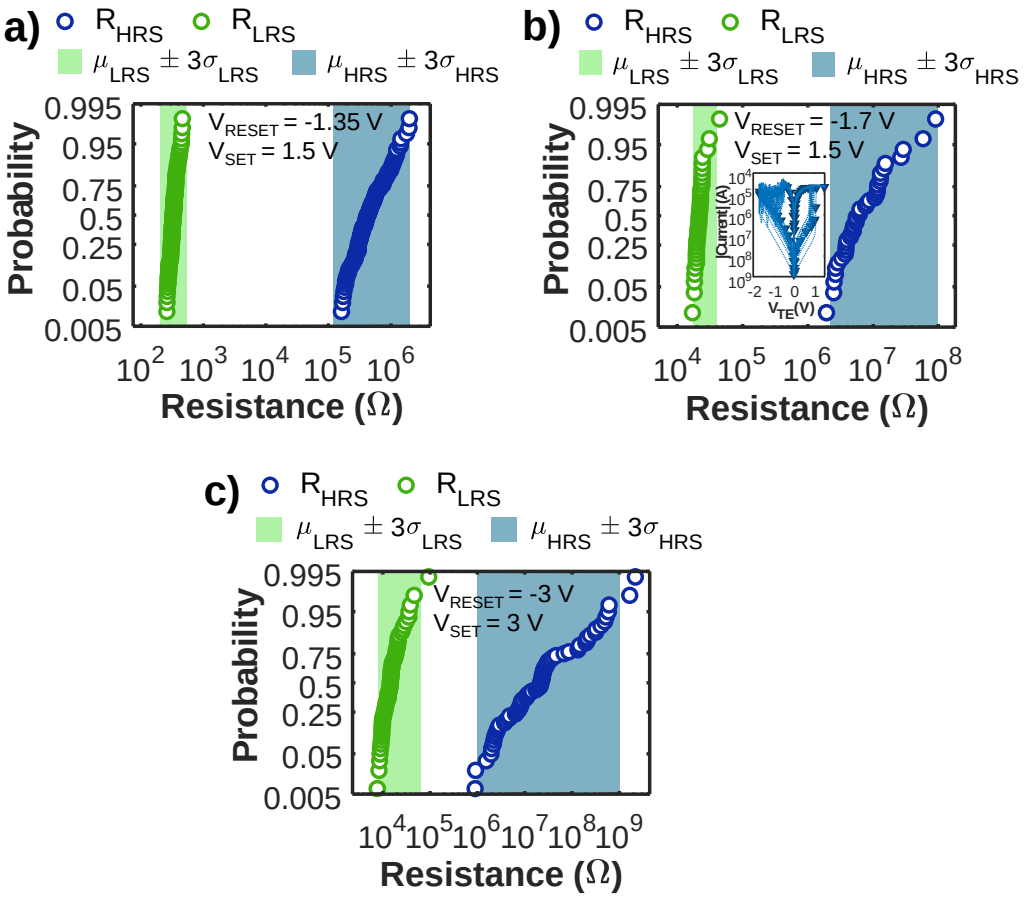


Figure 2.18: Probability plots of R_{HRS} and R_{LRS} experimental (green and blue bands) and simulated (green and blue squares) distributions, due to C2C variability under quasi-static DC condition. The experimental set and reset voltages used in the simulations are reported. Specifically, a) shows the C2C variability for technology 1 (simulation parameters $\sigma_x = 0.35nm$, $\sigma_S = 2.7nm^2$). Data from [131]; b) shows the C2C variability for technology 2 with $I_C = 20\mu A$, as for this technology experimental variability data [128] are available only at $I_C = 20\mu A$ (variability parameters $\sigma_x = 0.8nm$, $\sigma_S = 24.9nm^2$). The inset shows the experimental (triangles symbols) and simulated (lines) DC IV curves, where only $k_{T_{CF}}$ ($k_{T_{CF20\mu A}} = k_{T_{CF200\mu A}}/100$), $k_{T_{EX}}$ ($k_{T_{EX20A}} = k_{T_{EX200\mu A}}/10$) and S ($S_{20\mu A} = S_{200\mu A}/10$) parameters were changed to account for the different characteristic of the very narrow CF obtained with such a low current compliance; c) shows the variability for technology 3 (simulation parameters $\sigma_x = 0.81nm$, $\sigma_S = 0.7785nm^2$). Data from [129].

based modeling only few parameters of the compact model which model the CF characteristic (i.e., S, $k_{T_{CF}}$, and $k_{T_{EX}}$) need to be scaled to reproduce the device behavior at the lower I_C, as shown in the inset of Fig. 2.18b.

2.4 Related published works

Some of the results presented in this Chapter were published during the PhD program in the following journals [117], [132], international conferences [133], [134] and repository [112].

Chapter 3

Study and development of RRAM-based in-memory computing architectures

This chapter focuses on the design and study of the performance and reliability of different in-memory computing paradigms exploiting the UniMORE physics-based RRAM compact model described in Chapter 2. RRAM devices' characteristics and nonidealities typically result in complex circuits, demanding the use of appropriate tools to identify specific performance and reliability trade-offs.

The rest of the chapter is structured as follows. In Section 3.1, the conventional LiM computing architecture based on the IMPLY logic is described and our results from the circuit reliability analysis are presented. In Section 3.2, we present the developed SIMPLY LiM architecture and demonstrate its performance and reliability improvements with respect to the conventional IMPLY architecture by means of circuit simulations. In Section 3.3, we present the results of the performance and reliability analysis of a LBPNN

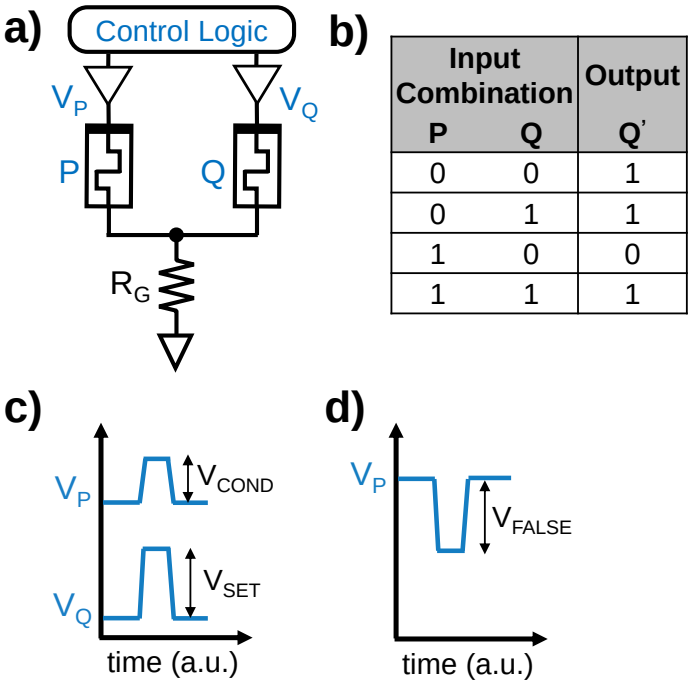


Figure 3.1: a) Basic circuit of the conventional IMPLY gate. The control logic and the analog tri-state buffers are included. b) Truth table of the IMPLY operation where Q' represents the state of the device Q after an IMPLY operation execution. c) Voltage pulses delivered to the TEs of devices P and Q during an $IMPLY(P,Q)$ execution. The amplitude of the voltage pulses is reported. d) Voltage pulse delivered to the TE of device P to execute the $FALSE(P)$ operation.

hardware accelerator implemented on 1T1R memory arrays. Finally, in Section 3.4, we propose a novel in-memory computing architecture merging the SIMPLY and analog VMM in-memory computing paradigms on the same 1T1R memory array.

3.1 IMPLY LiM architectures

IMPLY logic is a functionally complete type of logic that is based on two core operations [135], [136], i.e., the $IMPLY$ which is described by the truth

table shown in Fig. 3.1b, and the $FALSE$ which always results in a "0" logic value. Such logic was first theorized by Whitehead and Russell [137] at the beginning of the 20^{th} century but failed to find application in electronics circuits, due to the higher complexity of its transistors-based implementation compared to the traditional logic functions (e.g., AND, OR, NOT, etc.). It is only recently that the need for new in-memory computing architectures and the development of RRAM devices has reinvigorated the interest of the electronic community for different logic primitives, and in 2010 Borghetti et al. [26] experimentally demonstrated, using RRAM devices, an IMPLY-based LiM circuits. This work was followed by other experimental works [27], and several theoretical works aimed at understanding the limitations of the circuit implementations [10] and to optimize the throughput of the computation [136]. Still, most of such works relied on mathematical approximations such as the use of simple device mathematical models (e.g., general-purpose memristor models) leading to inaccurate estimates of the circuit performance and reliability.

3.1.1 Working principle

To implement the IMPLY operation with RRAM devices a simple circuit as the one shown in Fig. 3.1a is required. In this circuit, the resistive state of RRAM devices is the input and output of any IMPLY and FALSE operations, with the HRS and LRS representing a "0" logic and "1" logic values, respectively. The BE of RRAM devices is short-circuited and connected to ground through a resistor R_G. Their TE is connected to tri-state analog buffers that are controlled by a control logic to deliver appropriate voltage pulses to the RRAM devices. Such circuit can also be implemented on RRAMs linear or crossbar arrays, enabling the computation of any complex operation as a se-

quence of IMPLY and FALSE operations between the devices in the array [136].

To execute a $FALSE$ operation on device P (i.e., $FALSE(P)$), a negative voltage pulse with amplitude V_{FALSE} (see Fig. 3.1d) is delivered to the TE of P, thus always restoring the HRS (i.e., "0" logic) of the device P, while the other tri-state buffers are kept in high impedance (Hi-Z). Conversely, to execute an $IMPLY$ operation between two devices P and Q (i.e., $IMPLY(P,Q)$), two simultaneous positive voltage pulses are delivered to the TE of the RRAMs P and Q, as shown in Fig. 3.1c. Since positive pulses are used, devices can only switch from an HRS to a LRS. Still, the amplitude of the two pulses determines the correct circuit operation. Specifically, the amplitude of the voltage on Q (i.e., V_{SET}) must be high enough to switch the state of Q when both the devices are initially in HRS, while at the same time it must be sufficiently low to prevent the switching of Q when $P = 1$ and $Q = 0$. At the same time, the amplitude of the voltage on P (i.e., V_{COND}) must be sufficiently low to always prevent the switching of P but also high enough to counter the switching of Q when $P = 1$ and $Q = 0$. Thus, strict trade-offs exist for the selection of V_{SET} and V_{COND}, leading to high design complexity and reduced circuit reliability, as discussed in the following section.

3.1.2 Reliability analysis

IMPLY gate

The strict trade-offs on the driving voltages selection introduce an additional undesired effect, i.e., the logic state degradation (LSD), which affects each RRAM device that should preserve an HRS during an IMPLY operation.

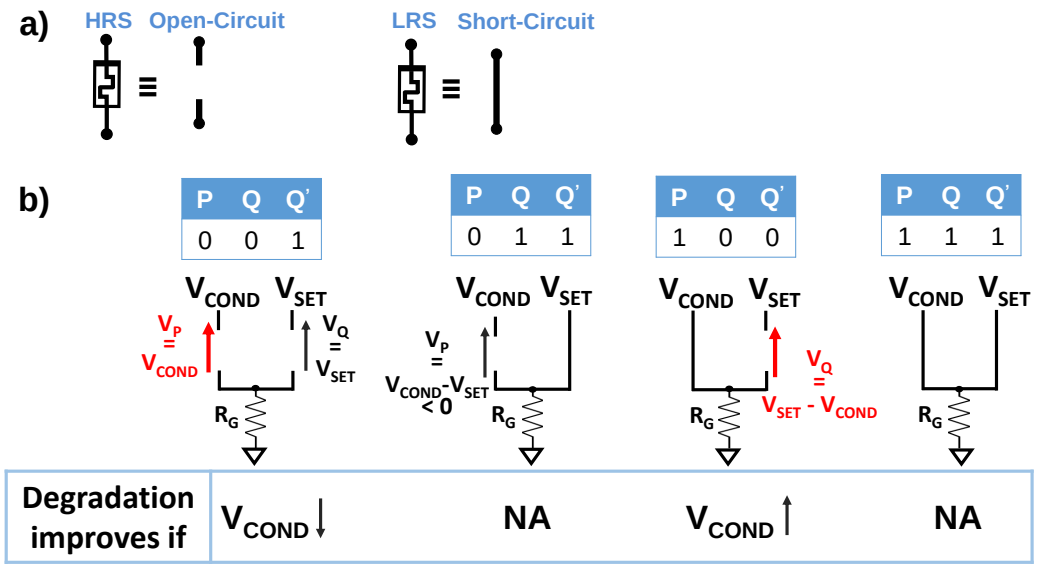


Figure 3.2: a) Sketch of the approximation that is used to illustrate the problem of LSD. A device in HRS and in LRS is approximated as an open- and a short-circuit, respectively. b) Sketch of the equivalent circuits for the four cases of the IMPLY truth table using the approximation described in a). LSD affects devices P and Q in cases 1 and 3 of the truth table, respectively. The trade-off on the choice of V_{SET} and V_{COND} for achieving correct gate operation and low LSD.

As illustrated in Fig. 3.2 by approximating a device in LRS with a short-circuit and a device in HRS with an open-circuit, during the execution of an $IMPLY(P, Q)$ operation, positive voltages are applied to devices P and Q when considering the first and third cases of the truth table, respectively. Although such positive voltages cannot completely switch the device into a LRS, they can cause a slight drop in the device resistance. Thus, repeatedly performing IMPLY operation using such devices as inputs results after a limited number of operations in a bit corruption, i.e., the resistance of the device drops to a LRS. The existence of the strict trade-off on V_{COND} makes it impossible to completely solve the problem of LSD, since lower and higher V_{COND} values are required to lower the effect of LSD in the first and third case of the truth table, respectively, as shown in Fig. 3.2b. Thus, time and energy-consuming periodic memory refresh cycles which consist in temporarily copying the content of the entire memory to another location, resetting all devices to an HRS, and rewriting the original memory content, are needed, thus limiting the overall architecture performance. Also, as shown in Fig. 3.3a,b, the rate at which the device HRS degrades depends on the initial device resistance, thus suggesting that the effect of the intrinsic RRAM devices variability further complicates this issue. Moreover, even small voltage variations on the driving voltages can considerably change the circuit performance. This is clearly shown in Fig. 3.4, where we report the results of circuit simulations of the IMPLY gate tested for different pairs of V_{SET} and V_{COND} voltages. As shown in Fig. 3.4a, c even when considering the ideal device with nominal HRS and LRS resistances without variations, the IMPLY gate correctly works only for a few V_{SET} and V_{COND} pairs (i.e., colored region). Also, an additional trade-off exists between energy consumption, LSD, and tolerance to driving voltages variations. To minimize LSD and maximize the

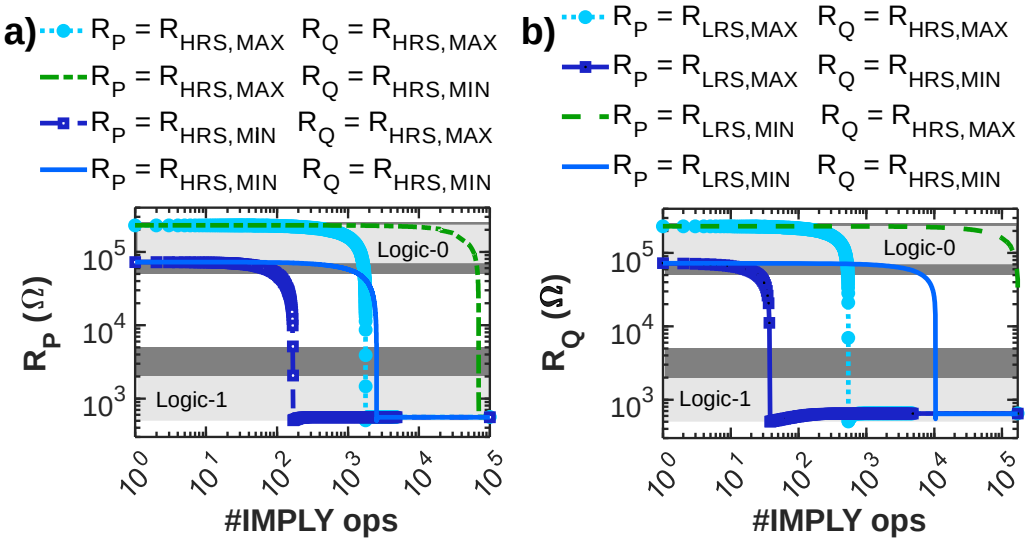


Figure 3.3: a) LSD on device P, showing a clear drop in the device resistance (R_P) after executing the $IMPLY(P, Q)$ operation when $P = Q = 0$ for a sufficient number of times. The effect of LSD is reported for the 4 corner cases on the initial devices resistances, highlighting the effect of variability. In the worst case (blue dot-dashed line with squares), the stored bit is corrupted after only ≈ 100 IMPLY operations. b)LSD on device Q when repeatedly executing the $IMPLY(P, Q)$ operation when $P = 1$ and $Q = 0$. The results are reported for the 4 corner cases on the initial devices resistances, highlighting the effect of variability. In the worst case (blue dot-dashed line with squares), R_P drops rapidly after just ≈ 30 IMPLY operations, leading to a bit corruption. Circuit simulations were performed using Technology 1.

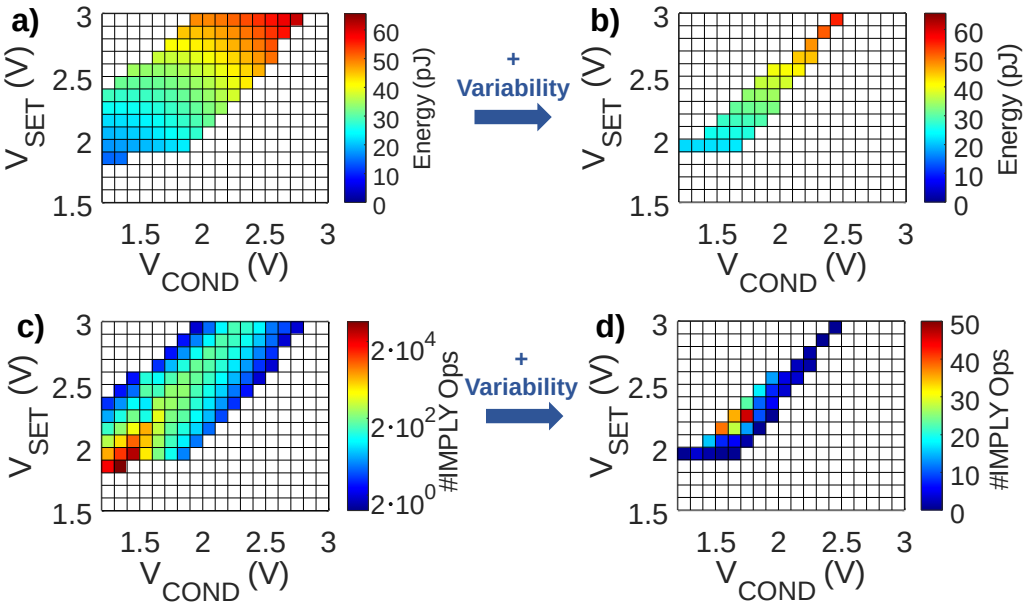


Figure 3.4: a), c) Map of the energy per IMPLY operation a) and maximum number of cycles before bit corruption b) for different V_{SET} and V_{COND} pairs when considering RRAM devices with the nominal HRS and LRS resistances. In the white regions, the gate does not correctly work. b) Map of the energy per IMPLY operation b) and maximum number of cycles before bit corruption b) for different V_{SET} and V_{COND} pair when the effect of variability is included in the circuit simulations. The region of correct circuit operation shrinks considerably. Circuit simulations were performed using Technology 1.

energy efficiency one should prefer low V_{SET} and V_{COND} voltages (i.e., light green and red/magenta regions in Fig. 3.4a, and c, respectively), however leading to smaller tolerated voltage variations (i.e., a few tens of mV voltage drops would result in errors). Also, the inclusion of the effect of resistive state variability in the circuit simulations shows that a far worse scenario is expected with real devices, since the region of correct circuit operation shrinks considerably, see Fig. 3.4b, d. Also, as shown in Fig. 3.4d the number of IMPLY operations before bit corruption drops by more than 10^3 times, underlining the importance of physics-based simulations which include the effect of the resistive state variability when evaluating the circuit reliability. In fact, by using a simplified model one could design the circuit with a V_{SET} and V_{COND} pair that falls outside the region of correct circuit operation when considering the effect of variability or employ an excessively long memory refresh rate, thus leading to high BER or a faulty circuit. Additionally, nonideal effects in the circuit layout and in the control logic should be taken into considerations. Specifically, the previous reliability analysis was carried out considering ideal and perfectly simultaneous voltage pulses however, including small delays between the delivered voltage pulses can result in errors when computing an IMPLY operation. For instance, if V_{SET} is delivered even a few picoseconds before V_{COND} and the inputs (i.e., P and Q devices) are in the case 3 configuration, the device Q could erroneously switch or be affected by an increased rate of LSD. Thus, we analyzed the tolerance of the IMPLY gate to positive (i.e., $t_{delay_{Q\rightarrow P}} > 0$) and negative (i.e., $t_{delayQ\rightarrow P} < 0$) time skew between the inputs for different driving voltage amplitudes [82], and the results of the analysis are reported in Fig. 3.5. As expected, the case 3 input combination is the most sensitive to time skews since by reducing the overlap between the driving signals the switching inhibition effect of V_{COND}

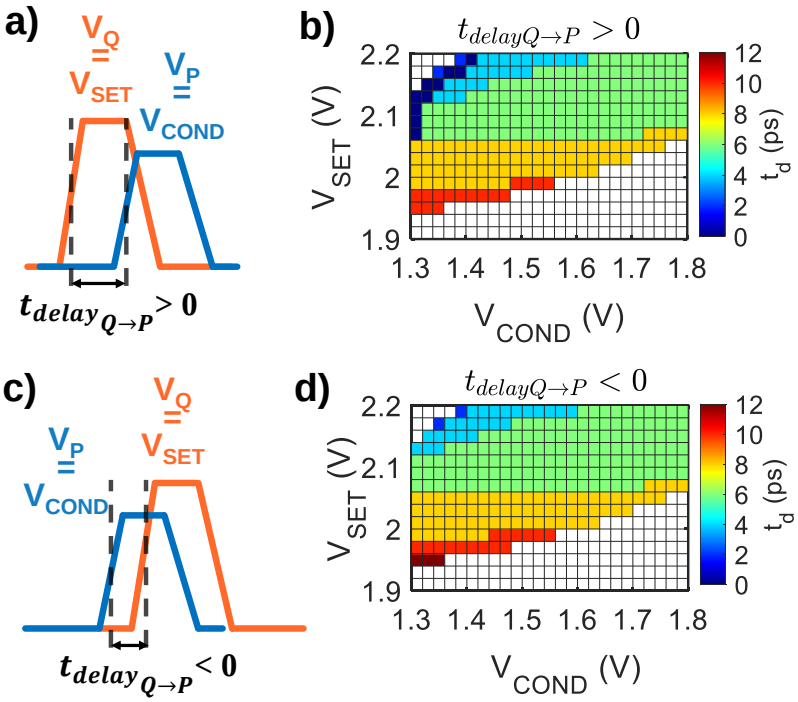


Figure 3.5: a), c) Definition of positive a) and negative c) time delay between the V_{SET} and V_{COND} signals. b), d) Map of the maximum tolerated positive and negative skews between the driving signals, respectively. Higher V_{SET} or V_{COND} voltages result in lower tolerated skew values. Circuit simulations were performed using Technology 1.

is reduced, casing the switching of Q especially when higher V_{SET} voltages are applied.

IMPLY gate on RRAM linear arrays

Real use cases for the IMPLY architectures require more than 2 RRAM devices to implement complex logic operations. Thus, linear RRAM arrays, such as the one in Fig. 3.6a could be employed. Although in this circuit the effect of line coupling parasitic capacitance (i.e., c_{PAR}) can be neglected due to its extremely low value, the effect of line parasitic resistance (i.e., r_{PAR}) must be taken into account. In fact, from the analysis of the single IMPLY gate, it is clear that voltage drops on parasitic resistances reduce the effective voltage which drops on RRAM devices thus shifting the designed operating point and potentially causing the circuit to fail. In a linear RRAM array, the direction and the amount of offset of the operating point depend both on the relative position of the devices in the array and their distance from the resistor R_G and thus analyzing specific corner cases for increasing array size (i.e., n) enables to determine the maximum array size for reliable circuit operation. Still, due to RRAM devices' nonlinearity in HRS and possible switching events, the effective V_{SET} and V_{COND} voltages cannot be calculated analytically but require the use of physics-based circuit simulations. Thus, simulations for increasing array size were performed on the equivalent circuit in Fig. 3.6b where the RRAM devices that are not used in the simulations are removed since their analog buffers are in Hi-Z and using an $r_{PAR} = 1.908\Omega$ that was estimated for a realistic 50 nm feature size in [138]. Three corner cases were simulated by appropriately adjusting the parameters j, k, R_k, R_j, n, V_1, and V_2 (see Fig.3.6b,c), which determine the device position and initial resistance in the array, the array size, and the voltages applied to

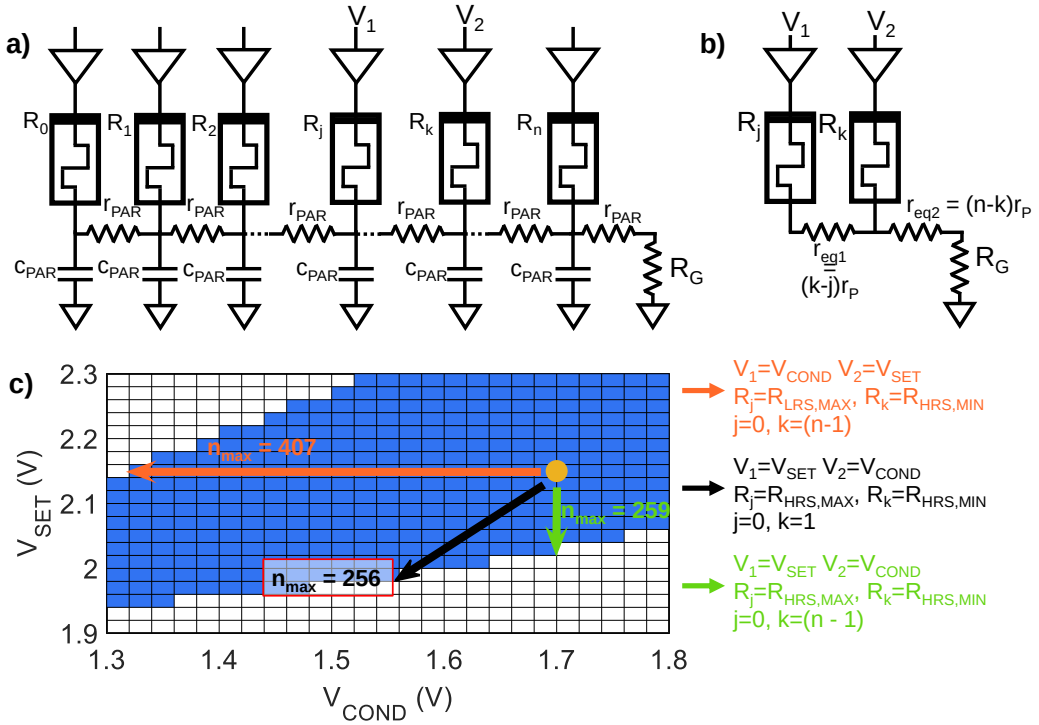


Figure 3.6: a) Linear RRAM array where the line parasitic resistances (r_{PAR}), node capacitances (c_{PAR}) are included. The control voltages for a generic $IMPLY(J,K)$ are reported. b) Equivalent circuit of the circuit in a), used to analyze the effect of the line parasitic resistances on the IMPLY gate functionality. c) Simulations results for three corner cases described in the legend. $R_{LRS,MAX}$, $R_{HRS,MAX}$, $R_{HRS,MIN}$, are the extreme values of the LRS and HRS resistance distributions, respectively. The maximum size of the array is determined as the minimum number of devices shifting the nominal operating point outside the region of correct operation. Circuit simulations were performed using Technology 1.

devices in position j and k, respectively. The total line parasitic resistance (i.e., $n \cdot r_{PAR}$) is split into two pieces, an equivalent resistance (r_{eq_1}) between R_j and R_k, and an equivalent resistance (r_{eq_2}) between R_k and R_G. As shown in Fig. 3.6c, by changing these parameters the initial operating point is shifted in different directions and a worst-case condition (i.e, black arrow) is determined therefore setting a boundary for the maximum array size (i.e., $n_{max} = 256$). This analysis could be repeated for different nominal operating points (i.e., the nominal pair of V_{SET} and V_{COND}) to determine the maximum achievable RRAM array size for a specific RRAM technology and R_G value. Indeed, a higher V_{SET} would increase n_{max} in the black and green arrow directions but reduce it in the orange arrow direction (see Fig.3.6c), thus highlighting the existence of an upper boundary for the array size. Also, the effect of LSD, energy consumption, and tolerance to skews between the driving signals change for different operating points. For instance, higher V_{SET} would lead to higher energy consumption, lower tolerance to LSD, and driving voltage skews, thus determining a complex optimization problem with complex trade-offs between maximum array size, energy efficiency, and circuit reliability, which requires the iterative use of physics-based simulations.

3.2 Smart IMPLY architecture

3.2.1 Working principle

To address the design complexity and reliability issues in conventional IMPLY-based architectures we propose the SIMPLY LiM architecture. As it can be observed from the IMPLY operation truth table in Fig. 3.1b, the state of the device Q changes during an $IMPLY(P, Q)$ only when the inputs are in the $P = Q = 0$ configuration, while the initial state of Q is preserved in all the

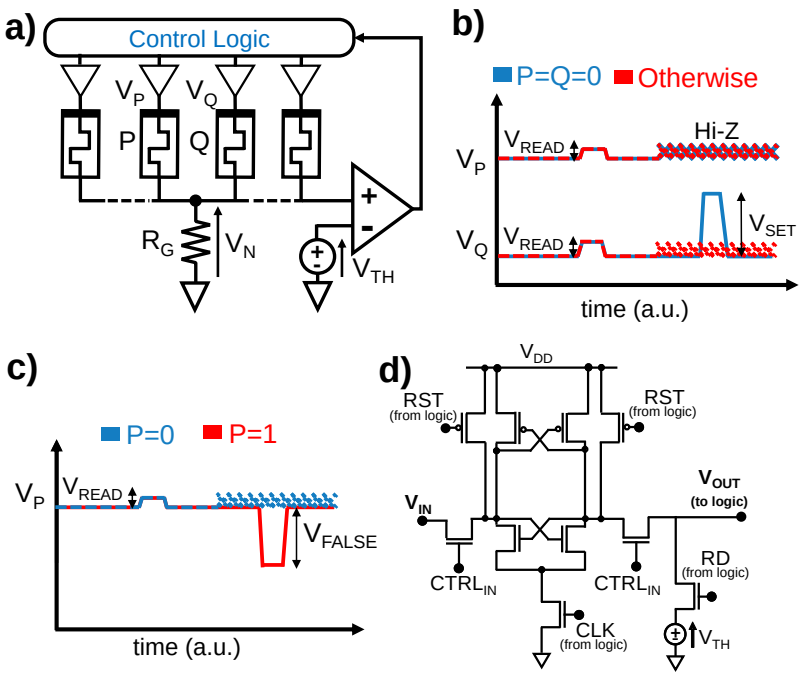


Figure 3.7: a) SIMPLY architecture implemented on a linear RRAM array. The control logic, the analog tri-state buffers, and the comparator are reported. b) Voltage pulses delivered to the TEs of devices P and Q to execute the $IMPLY(P,Q)$ operation in the SIMPLY framework. The comparator detects when $P = Q = 0$ (blue lines) and delivers V_{SET} to Q. In all other cases (dashed red lines), the analog buffers are kept in high impedance (Hi-Z). c) Voltage pulses delivered to the TEs of P to execute the $FALSE(P)$ operation in the SIMPLY framework. d) Simulated voltage sense amplifier circuit implementing the comparator.

other cases. In SIMPLY, the $P = Q = 0$ input combination is distinguished from all the others using a comparator with an appropriate threshold voltage (i.e., V_{TH} in Fig.3.7a) and splitting the operation into two steps. First small ($\approx 100mV$) read voltage pulses with amplitude V_{READ} are delivered in parallel to the devices P and Q and the voltage across R_G (i.e., V_N in Fig.3.7a) is compared with V_{TH}. The output of the comparator is fed to the control logic which then pulses V_{SET} on Q only when $P = Q = 0$, while in all the other cases the tri-state analog buffers are kept in Hi-Z, as shown in Fig.3.7b. In fact, when both inputs are in HRS the V_N voltage, resulting from the voltage divider with R_G, is lower than in all the other cases when at least a device is in LRS, therefore providing a read margin (RM) at the input of the comparator which can thus correctly detect the $P = Q = 0$ input configuration. To ensure high energy efficiency a low-power comparator design should be employed. Thus, we designed and simulated the sense amplifier (SA) circuit shown in Fig. 3.7d, using a $45nm$ CMOS technology from [139]. Such design occupies a small area on the chip (all FETs have a $W = 90nm$ and $L = 50nm$) and dissipates very small energy (i.e., just $8fJ$ per comparison when T is in the range 0°C to 85°C and V_{DD} is $2V$).

Thus by using SIMPLY, several improvements are achieved with respect to conventional IMPLY architectures. Specifically, SIMPLY breaks the trade-off between V_{SET} and V_{COND} since V_{COND} is not needed and simplifies the circuit design by setting precise performance vs reliability trade-offs. Specifically, higher circuit reliability translates to higher RMs which can be therefore achieved by moderately increasing V_{READ} or by increasing $|V_{FALSE}|$ and consequently R_{HRS}, however resulting in higher energy dissipation during the read step and FALSE operation. Additionally, SIMPLY results in a much lower energy per IMPLY operation since the large V_{SET} voltage pulse is ap-

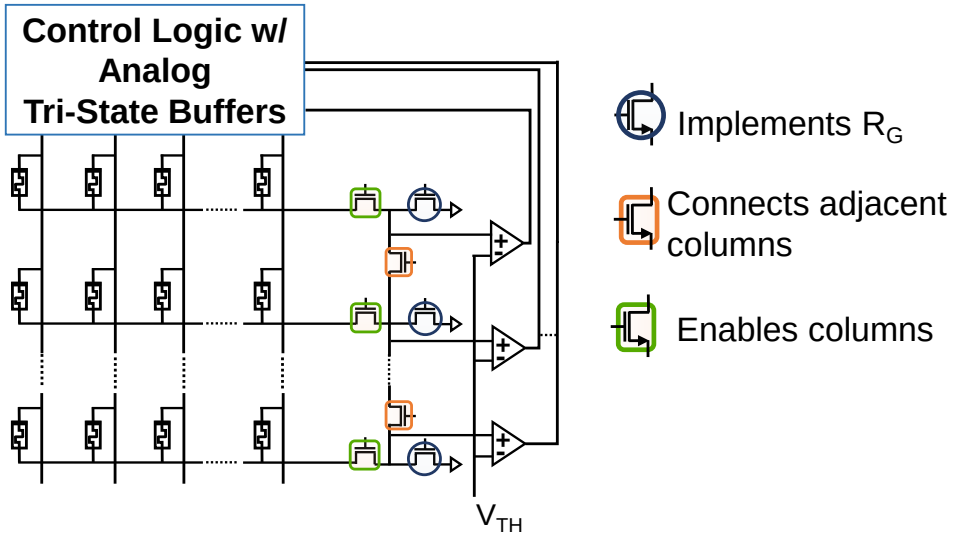


Figure 3.8: SIMPLY implementation on an RRAM crossbar array. By using more than one comparator and the FET devices enclosed in the blue circle, and orange and green rectangles the computation parallelism can be increased.

plied only in one case of the truth table while only small read voltage pulses are used in all the other cases, resulting in lower energy dissipation. All these improvements come at a moderate cost of a negligible increase in the complexity of the control logic and of a lower computing speed, which is well justified by the achieved considerable improvements.

In SIMPLY also the FALSE operation can be optimized by following an approach similar to the one used for the IMPLY operation. Indeed, delivering V_{FALSE} pulses to a device already in HRS would result in energy wastes. Using the same comparator, voltage threshold, and read voltage, the state of the device can be read, and consequently, the V_{FALSE} is delivered to the device only when a LRS is detected. Although the energy improvement may be limited for technologies with very high HRS, avoiding the application of very large V_{FALSE} pulses when a device is in HRS reduces the times that an RRAM is exposed to high voltage stresses, thus improving the circuit endurance. As the conventional IMPLY architecture, also the SIMPLY ar-

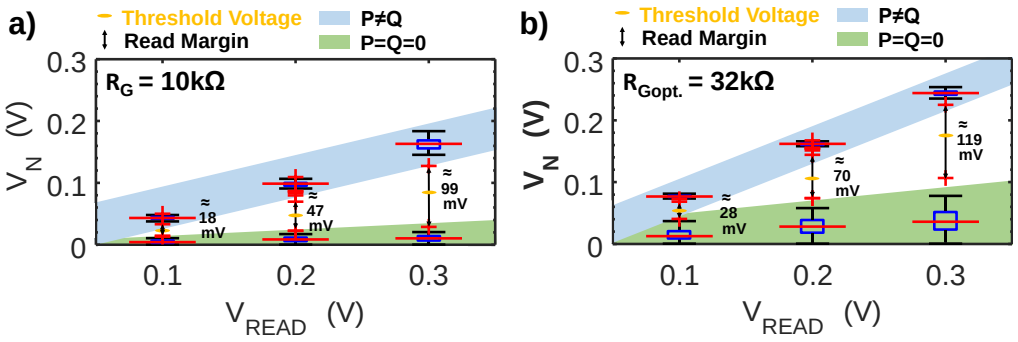


Figure 3.9: Box plots of the distributions of the V_N voltage during the read step of the IMPLY operation in SIMPLY, when $P = Q = 0$ (green band) and $P \neq Q$ (blue band), and for different R_G values. Specifically, the results for a sub-optimal and the optimal R_G are reported in a) and b), respectively. Using $R_{G_{opt.}}$ increases the read margin (RM - black arrow). The simulations were performed on Technology 3. The blue boxes, and black whiskers, indicate the $25^{th} - 75^{th}$ percentile range and the extreme values of the distributions, respectively. Outliers are indicated with red crosses.

chitecture can be implemented on linear and crossbar arrays. As shown in Fig.3.8, in the crossbar implementation FET devices are used to implement R_G (FET enclosed in the blue circles), to enable a specific column (FET enclosed in the green rectangles), and to connect adjacent columns (FET enclosed in the orange rectangles). Multiple SAs are introduced in the array periphery to enable the parallel execution of IMPLY and FALSE operations between devices on the same columns but different rows, therefore resulting in a SIMD architecture.

3.2.2 Reliability analysis

The circuit reliability of the SIMPLY architecture can be quantitatively estimated using physics-based simulations enabled by the UniMORE compact model. As already mentioned in the previous section, to ensure the correct circuit operation a sufficient RM needs to be provided at the input of the

comparator. Thus, the RM represents the main circuit reliability metric. To estimate the RM, the V_N voltage distributions when $P = Q = 0$ and $P \neq Q$ are estimated performing circuit simulations that include both HRS and LRS variability and the effect of multilevel RTN. As shown in Fig.3.9 for Technology 3, even when including the effect of RTN and variability a sufficient RM exists, and the use of higher V_{READ} values leads to larger RMs. Furthermore, the choice of R_G influences the RM, and its optimal value (i.e., $R_{G_{opt.}}$) can be computed using Equation (3.1), where $R_{HRS,MAX}$, $R_{HRS,MIN}$, and $R_{LRS,MAX}$ are the extreme values of the HRS and LRS resistance distributions, respectively. As shown in Fig. 3.9b, the use of $R_{G_{opt.}}$ results in the largest possible RMs that can be achieved with the specific technology, current compliance, and reset voltage.

$$R_{G_{opt}} = \sqrt{\frac{1}{\frac{1}{R_{HRS,MAX}} + \frac{1}{R_{LRS,MAX}}} \cdot \frac{R_{HRS,MIN}}{2}} \tag{3.1}$$

Compared to the conventional IMPLY architecture, the use of the SIMPLY architecture virtually solves the problem of the LSD. Indeed, the use of low read voltages slows down LSD, and as shown in Fig.3.10, no relevant degradation is observed up to $4.5 \cdot 10^6$ IMPLY operations executions even when the worst case is considered.

3.2.3 Multi-input IMPLY operation

Compared to traditional CMOS logic gates, where the implementation of more complex operations translates to an increased number of transistors, in IMPLY-based implementations more complex logic operations require longer sequences of IMPLY and FALSE computing steps. Although some applications benefit from the increased parallelism offered by IMPLY-based ar-

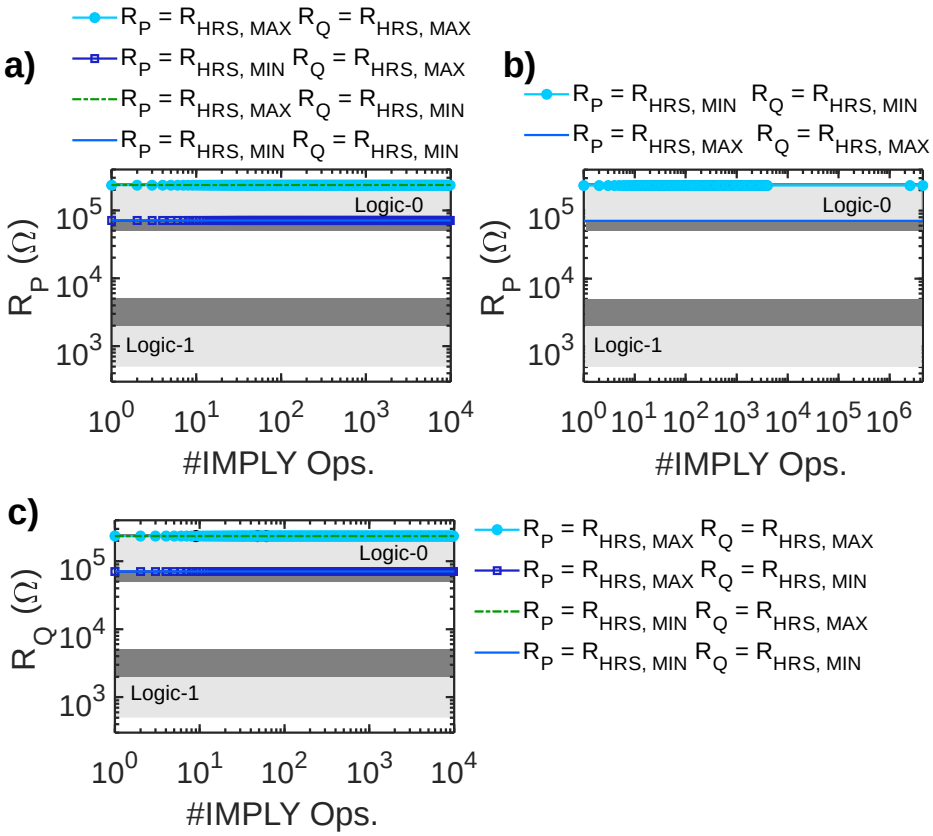


Figure 3.10: a) Resistance of device P when repeatedly used in an $IMPLY(P, Q)$ operation when $P = Q = 0$ in the SIMPLY framework. For all four corner cases, only negligible degradation of the resistive state is observable up to 10^4 IMPLY operations. b) Considering the two worst cases for degradation due to variability, degradation is negligible at least up to $4.5 \cdot 10^6$ cycles. c) Resistance of device Q when repeatedly used in an $IMPLY(P, Q)$ operation when $P = 1\, and\, Q = 0$ in the SIMPLY framework. Also in this case, for all four corner cases, only negligible degradation of the resistive state is observable up to 10^4 IMPLY operations. Circuit simulations were performed using Technology 1.

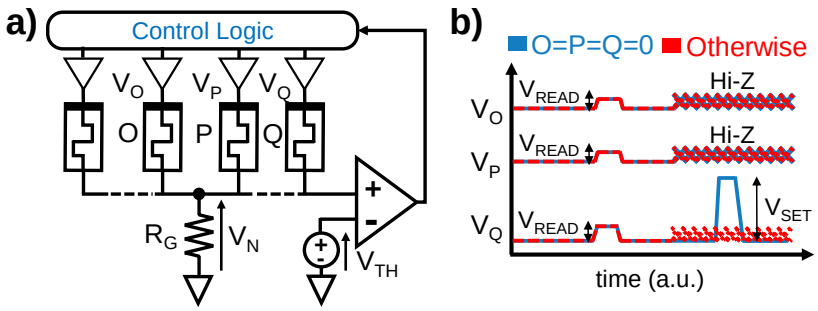


Figure 3.11: a) n-SIMPLY (with n=3) operation performed on the SIMPLY architecture implemented on a linear RRAM array. b) Voltage pulses delivered to the TEs of devices O, P, and Q to execute the $3-SIMPLY(O,P,Q)$ operation in the SIMPLY framework. The comparator detects when $O=P=Q=0$ (blue lines) and delivers V_{SET} to Q. In all other cases (dashed red lines), the analog buffers are kept in high impedance (Hi-Z).

chitectures, reducing the overall number of computing steps remains critical for improving efficiency. A promising solution would be to increase the IMPLY operation fan-in to more than two inputs [136], [140], implementing the n-IMPLY operation, where V_{SET} is applied to a single device while V_{COND} is simultaneously applied to $n-1$ devices. Siemon et al. [141] recently demonstrated the feasibility of the n-IMPLY operation with n=3 (i.e., $IMPLY(O,P,Q) \equiv OR(Q,NOR(O,P))$) by performing physics-based simulations. However, increasing the number of inputs of an IMPLY operation worsen the reliability issues (e.g., LSD) affecting the conventional IMPLY architectures, preventing the implementation of the n-IMPLY operation with real devices.

Following the same approach used for the two inputs (i.e., 2-IMPLY) operation, the n-IMPLY operation can be implemented also on the SIMPLY architecture, see Fig.3.11. In fact, also in the $IMPLY(...,O,P,Q)$ the state of device Q changes only when all the inputs are in HRS, which can be detected by applying V_{READ} to all the input devices in parallel and comparing the voltage at node N with the same comparator and threshold voltage used

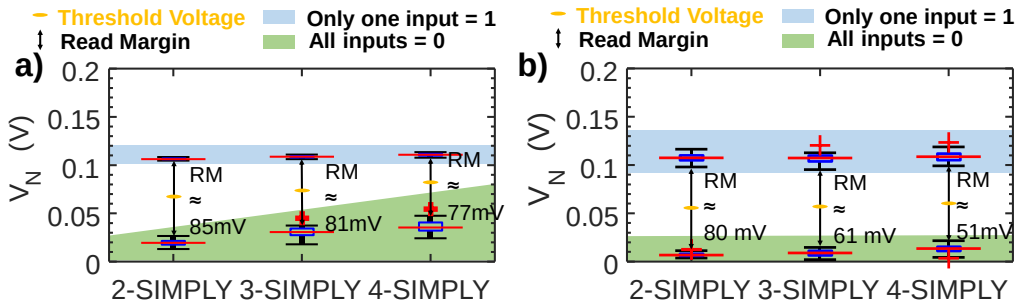


Figure 3.12: Box plots of the distributions of the V_N voltage during the read step of the n-SIMPLY operation (with n=2, 3, 4) in the SIMPLY framework, when $O = P = Q = 0$ (green band) and $O = 1P = Q = 0$ (blue band). a), b) show the V_N distributions experimentally evaluated on Technology 4 in a), and obtained by performing circuit simulation on Technology 3 in b). When increasing the number of devices read in parallel, the read margin (RM - black arrow) decreases. The blue boxes, and black whiskers, indicate the $25^{th} - 75^{th}$ percentile range and the extreme values of the distributions, respectively. Outliers are indicated with red crosses.

for the 2-IMPLY operation. Thus, no changes to the memory architecture are needed. To verify its correct functionality an analysis of the available RM for increasing n needs to be performed. The RM for $n = 2, 3, 4$ was experimentally evaluated and by performing circuit simulations on Technology 4, and 3 respectively. Despite the increase of the fan-in, a sufficient RM is still available even for $n = 4$, as shown in Fig.3.12a, and b, confirming the feasibility of the n-IMPLY operation implementation on the SIMPLY architecture. The high circuit reliability is further demonstrated by the result of the LSD analysis (see Fig.3.13) which shows that no relevant degradation is expected up to 10^8 n-IMPLY operation execution (with n=3,4) even when considering the worst-case for LSD.

Furthermore we highlight that the implementation of the n-IMPLY operation enabled by the SIMPLY architecture, simplifies the synthesis of complex logic functions by exploiting the canonical sum of products (SoP) representa-

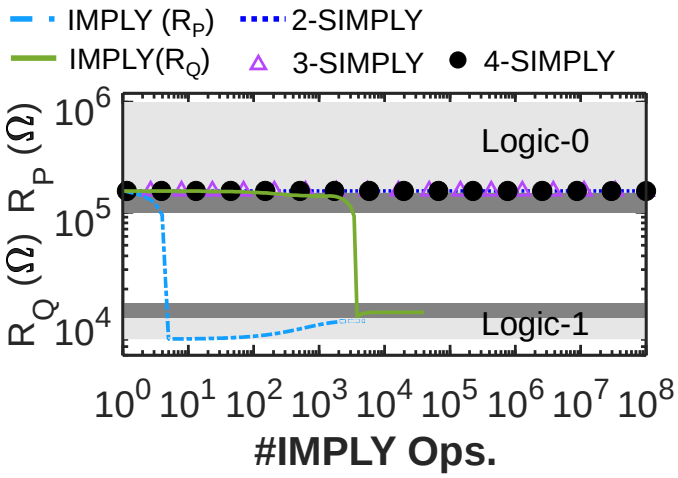


Figure 3.13: Results of the analysis of LSD of R_P (dashed blue line) and R_Q (solid green line) due to the repeated execution of the IMPLY and n-SIMPLY (with n up to 4) operations with Technology 3. The values stored in devices P and Q are corrupted after a limited number of IMPLY operations executed on the conventional IMPLY architecture when considering the two worst cases for degradation due to variability. When considering the n-SIMPLY operation, no noticeable degradation is observed up to at least 10^8 cycles.

tion. Although different optimization strategies have been proposed for the IMPLY logic [81], [142]–[144], these are commonly less intuitive and often require the use of more additional RRAM devices to compute intermediate results than the SoP implementation. As previously mentioned, the n-IMPLY operation can be represented as OR and NOR operations, with the latter being equivalent to an AND operation with inputs the complemented values of the inputs of the NOR operation (i.e., $NOR(A, B, C) \equiv AND(\overline{A}, \overline{B}, \overline{C})$). Therefore, any logic function can be synthesized in two steps, first, the complement of all the inputs to the AND gates is computed, then implement all AND gates with the equivalent NOR gates. Thus, to compute any logic function a number of (n+1)-IMPLY operations equivalent to the number of minterms, plus a number of FALSE and 2-IMPLY operations equivalent to the number of inputs that need to be complemented is required in all

cases. Also, no additional RRAM device to those storing the inputs and their complements and the output is needed, since all the partial results are computed directly on the output RRAM device. Also, RRAM devices storing the inputs are never overwritten during computations and therefore can be re-used. Furthermore, optimized sequences in which the number of minterms is minimized can be easily implemented by exploiting common optimization algorithms such as heuristic algorithms, Karnaugh Maps, and Binary Decision Diagrams [145]–[147].

3.2.4 Performance Benchmarking

To benchmark the performance of IMPLY-based architectures, we simulated using the UniMORE physics-based compact model calibrated on the four RRAM technologies the execution of different operations and tasks. Since HfO_x-based RRAMs technologies are known to switch in less than $1ns$ [148], in all the simulations $1ns$ voltage pulses are used to program or read the state of RRAM devices. Therefore, appropriate amplitudes for the set and FALSE pulses were determined using the physics-based compact model to ensure that the worst-case memory window (i.e., R_{HRS}/R_{LRS}) when including the effect of variability is larger than 10. The resulting circuit parameters used to simulate IMPLY and FALSE operations with the conventional IMPLY and SIMPLY architectures are reported in Table 3.1 for each technology [149]. FET devices from a $45nm$ technology [139] were used to implement R_G and the current compliance during a device SET.

Simple operations

In this section, the energy efficiency of the conventional IMPLY and SIMPLY architectures when implementing 1- and 2-inputs logic operations are

Table 3.1: Circuit parameters of IMPLY and SIMPLY architectures used in circuit simulations

Parameter	**Tech. 1**	**Tech. 2**	**Tech. 3**
$V_{COND}(IMPLY)$	1.65 V	1.1 V	2 V
$V_{SET}(IMPLY)$	2 V	1.7 V	3.2 V
$V_{SET}(SIMPLY)$	1.9 V	1.7 V	3 V
$V_{READ}(SIMPLY)$	0.2 V	0.2 V	0.2 V
V_{FALSE}	-1.55 V	-2V	-2.8V
R_G	2 $k\Omega$	6 $k\Omega$	10 $k\Omega$

discussed and reported. Specifically, when considering the core operations of IMPLY-based architectures, the use of SIMPLY results in considerable energy savings. When computing a 2-IMPLY operation in the SIMPLY framework, the high V_{SET} voltage pulse is delivered to the RRAM devices only when both the inputs are in HRS while in the remaining three possible input combinations only the small V_{READ} pulses are delivered to both devices, thus reducing the average energy per operation. The same approach can be used also for the FALSE operation, thus avoiding the use of the high V_{FALSE} pulse when a device is already in HRS. Also, since SIMPLY breaks the trade-off between V_{COND} and V_{SET}, the latter can be optimized for achieving higher energy efficiency (see Table 3.1). The performance improvement offered by SIMPLY is confirmed by the results of circuit simulation performed with the RRAM Technology 3 that are reported in Table 3.2. When equally likely input combinations are considered, SIMPLY provides an average energy saving of x33 and x12 for the 2-IMPLY and FALSE operations, respectively. The performance of IMPLY-based architectures when implementing other 2-inputs logic operations strictly depends on the sequence of IMPLY and FALSE operations that are employed. Thus, some logic operations can

Table 3.2: Performance comparison of IMPLY and FALSE operations executed on the IMPLY and SIMPLY architectures simulated using Technology 3

	IMPLY Arc.		**SIMPLY Arc.**	
Input Comb.	**2-IMPLY (min-avg.-max)**	**FALSE (min-avg.-max)**	**2-IMPLY (min-avg.-max)**	**FALSE (min-avg.-max)**
0 0	533-626-669 fJ	90-198-363 fJ	498-532-557 fJ	8.70-8.77-9 fJ
0 1	657- 672-691 fJ	217-357-492 fJ	11.7-11.8-11.9 fJ	213-341-492 fJ
1 0	251-266-287 fJ	NA	11.7-11.8-11.9 fJ	NA
1 1	660-678-699 fJ	NA	12.6-12.6-12.6 fJ	NA

be more effectively implemented than others when using the IMPLY logic. Specifically, the 2-NAND operation can be efficiently implemented by using just 3 computing steps [81], [136], which correspond to a FALSE of the RRAM storing the output and two IMPLY operations between each input and the output RRAMs. Also, SIMPLY is most effective in reducing the energy per 2-NAND operations since the output device switches at most once, meaning that V_{SET} cannot be delivered more than once per 2-NAND operations while the energy-efficient small read voltage pulses are used in all the other cases. On the contrary, the exclusive NOR and OR operations (i.e., 2-XNOR and 2-XNOR, respectively) require more computing steps (i.e., 11 and 13 steps for the 2-XNOR and 2-XOR, respectively. The lists of computing steps can be found in our works [117], [133]) and more support devices which are often switched when computing intermediate results. The best- and worst-case performance for a FALSE, a 2-IMPLY, a 2-NAND, and 2-XNOR operations were estimated for the three RRAM technologies from the literature (i.e., Technology 1, Technology 2, and Technology 3), and the results are reported in Table 3.3. Although for some technologies the worst-case energy per 2-IMPLY and FALSE operations can be slightly higher when

Table 3.3: Corner case performance comparison of 2- and 1-inputs operations with the IMPLY and SIMPLY architectures

	IMPLY Arc.				SIMPLY Arc. (2-SIMPLY)			
RRAM Tech.	2-IMPLY (min-max)	FALSE (min-max)	2-NAND (min-max)	2-XNOR (min-max)	2-IMPLY (min-max)	FALSE (min-max)	2-NAND (min-max)	2-XNOR (min-max)
Tech. 1	1.60-2.89	294-953 fJ	4.6-6.1 pJ	6.1-6.5 pJ	29-2140 fJ	17-932 fJ	0.7-2.8 pJ	10.7-12.4 pJ
Tech. 2	143-344fJ	160-1162 fJ	716-1640 fJ	4.0-5.8 pJ	13-370 fJ	10-1224 fJ	348-986 fJ	1.4-2.5 pJ
Tech. 3	197-710fJ	90-492 fJ	612-1481 fJ	3.9-5.0 pJ	12-571 fJ	9-492 fJ	0.8-1.2 pJ	2.0-4.0 pJ

using the SIMPLY architecture, the use of SIMPLY considerably reduces the energy for both the 2-NAND and 2-XNOR operations with the efficiency gain depending both on the specific RRAM technology and current compliance used. Indeed, the largest and lowest efficiency improvement is achieved when considering Technology 1, and 3, respectively which are programmed using the highest and lowest current compliance among the 3 technologies (i.e., $I_{C_{Tech.\,1}} = 1\,mA$, $I_{C_{Tech.\,3}} = 100\,\mu A$).

Large fan-in operations

Differently from conventional digital logic circuits, where increasing the fan-in of a logic operation consists in increasing the number of transistors, in IMPLY-based architectures larger fan-ins require the use of more IMPLY and FALSE computing steps thus requiring more energy and longer computing times which depend on the specific implemented logic operations. Indeed, as mentioned in the previous section, the IMPLY-based implementation of the NAND operation remains extremely efficient also for fan-ins larger than two (i.e., n-NAND), especially when using the SIMPLY architecture. Each additional input translates to just an additional IMPLY between the input and the output device (i.e., n+1 computing steps are needed for an n-NAND operation). Also, when using SIMPLY V_{SET} is delivered at most once to the output device, reducing the overall energy consumption and solving the

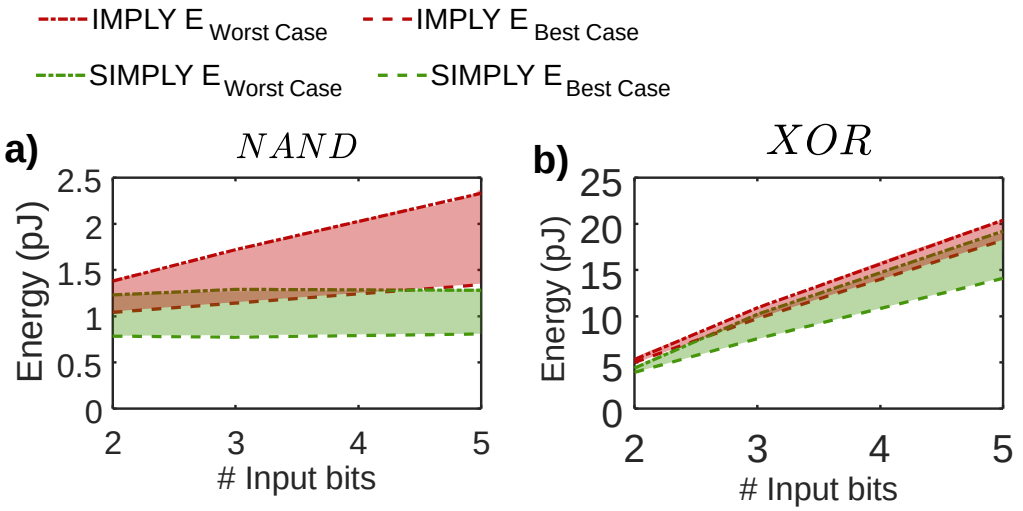


Figure 3.14: Comparison of the energy dissipated when computing the NAND a) and XOR b) operations in the IMPLY (red area) and SIMPLY (green area) frameworks and for an increasing number of input bits. The results of the simulations performed using Technology 2, for the best- and worst-case input combinations are reported.

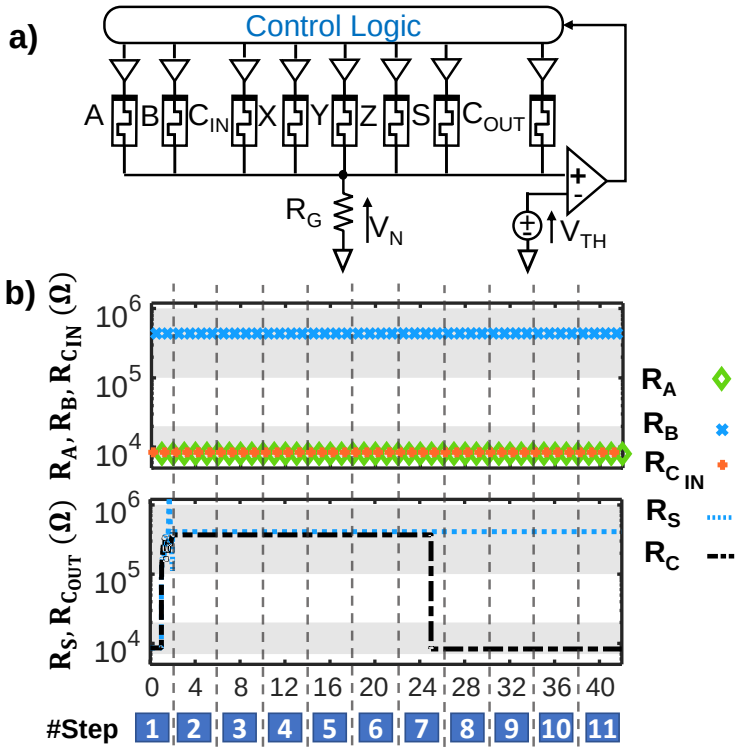
problem of LSD that would otherwise influence the circuit reliability when using the conventional IMPLY architecture, since many IMPLY operations are performed on the output device. These observations are confirmed by the result of circuit simulations of the best case (i.e., all inputs are 1) and the worst case (i.e., the first input is 0 and all the others are 1) input combinations shown in Fig. 3.14a. Although in the conventional IMPLY architecture the energy per operation rises linearly with the number of inputs, in the SIMPLY architecture it remains almost constant, leading to considerable energy efficiency improvement.

When considering other logic operations, the gain in efficiency offered by SIMPLY could also be lower. For instance, in the n-XOR operations, only a few set voltage pulses are skipped by using SIMPLY, since it is implemented as a sequence of 2-XOR operations using each time a new input and the partial result from the previous 2-XOR operation. As shown in 3.14b, also

when using SIMPLY the energy per operation grows linearly with the number of inputs both when considering the best- (all inputs are 1) and worst-case (all inputs are 0) input combinations. Still the use of SIMPLY results in a 24% and 6% energy efficiency improvement when considering the best- and worst-case input combinations, respectively. Also, by projecting the trends shown in Fig. 3.14, the energy estimates at fan-in larger than 5-bits can be calculated.

Complex logic operations

In this section, we benchmark the performance of the SIMPLY architecture on more complex logic operations, specifically the 1-bit full addition, the pop counting (i.e., binary accumulation), the comparator, and the hard max operations. These require more complex sequences of IMPLY and FALSE operations, which also depend on the number of support devices that are used, and retention or loss of the input data through the computation. For instance in Table 3.4 the characteristics of different implementations from the literature of a 1-bit full adder (FA) are reported [10], [132], [136], [141], [150], [151], together with the one we developed (i.e., Zanotti et al. [132], [150]). The performance of the 1-bit FA was estimated considering the linear array implementation shown in Fig. 3.15a, which is composed of 8 RRAM devices that store the inputs (i.e., A, B, C_{IN}), the outputs (i.e., S, C_{OUT}), and the results of intermediate steps (i.e., X, Y, Z). In the rest of this chapter, the term 2-IMPLY refers to a two inputs IMPLY operation executed on the conventional IMPLY architecture, while the terms 2-SIMPLY, and n-SIMPLY refer to a 2-IMPLY and n-IMPLY operations executed on the SIMPLY architecture, respectively. Differently from other works, only sequences that preserve the state of the inputs (i.e., the resistive state of



c)

	Energy FA (Avg.)	Delay	EDP
2-IMPLY*	51.8 pJ	56 ns	2.9×10^{-18}
2-SIMPLY*	17.2 pJ	92 ns	1.6×10^{-18}
n-SIMPLY	5.11 pJ	42 ns	2.1×10^{-19}

Figure 3.15: a) SIMPLY-based 1-bit FA implemented on a linear RRAM array where A, B, C_{IN} are the input bits, S and C_{OUT} are the output bits, and X, Y, Z are support devices used in intermediate steps. b) Resistance of devices A, B, C_{IN}, S, and C_{OUT} resulting from the simulation of the computing steps implementing the 1-bit FA operation performed using n-SIMPLY operations when $A = 1$, $B = 0$, and $C_{IN} = 1$. c) Comparison of the performance of 3 different IMPLY-based implementations simulated using Technology 1. * The delay and energy estimates from [150] are scaled to 1ns pulses. The use of n-SIMPLY results in a considerable EDP improvement.

the input devices does not change through the computation) were considered to evaluate the performance of 2-IMPLY, 2-SIMPLY, and n-SIMPLY implementations. Thus, in all the implementations reported in Table 3.4 which do not preserve the state of the inputs, six additional computing steps would be required to copy the state of the inputs and should be added to the number of computing steps for a fairer comparison. In this work, the performance of the 1-bit FA when using the 2-IMPLY operations are simulated using the optimized sequence, which includes the additional steps required to initialize the output and support devices and consists of a total of 28 computing steps. The complete sequence is available in [150]. As shown in Table 3.4, SIMPLY requires on average a third of the energy for each 1-bit full addition compared to the conventional IMPLY (i.e., 7^{th} and 8^{th} rows in Table 3.4), however increasing the time for each operation, therefore limiting the achievable Energy Delay Product (EDP) improvement, see Fig. 3.15c. Still, SIMPLY also dramatically improves the device endurance before a refresh is required, thus further reducing the energy consumption. To reduce the computing time the n-SIMPLY operation should be employed. In fact, by using the n-IMPLY operation with n up to 4, we devised a sequence which requires only 11 computing steps (the list of computing steps is available in [132]). An example of the state of the RRAM devices through the simulated FA operation is shown in Fig. 3.15b. By more than halving the number of computing steps, almost one order of magnitude improvement of the EDP with respect to the 2-SIMPLY implementation is achieved, as shown in Fig. 3.15c. Also, the n-SIMPLY implementations consume the lowest energy among other IMPLY-based 1-bit FA implementations from the literature, see Table 3.4. Only a hybrid FET-RRAM LiM design [151], shows better performance, however, such estimates were calculated using an extremely

Table 3.4: Detailed comparison among the proposed and existing FAs in the literature. SIMPLY, IMPLY, and Hybrid-CMOS LiM solutions (both experimental and simulation works)

Author(s)	LiM type (exp./sim.)	Physics-Based Model	Number of devices	Viable in crossbar	Energy (reported/ estimated)	Delay (reported/ estimated)	Number of steps	Preserves inputs	Endurance before refresh[$$]
Lehtonen et al. [136]	IMPLY (sim.)	✗	8 RRAM	✓	-	-	136	✓	?
Kvatinsky et al. [10]	IMPLY (sim.)	✗	9 RRAM	✓	-	9.1 μs (estimated)	23	✗	≈ 300 up to 10^5 (trades with energy)
Kvatinsky et al. [10]	IMPLY (sim.)	✗	6 RRAM	✓	-	11.5 μs (estimated)	29	✗	≈ 300 up to 10^5 (trades with energy)
Cheng et al. [27]	IMPLY (exp.)	NA	8 RRAM	✓	19.5 pJ (reported)	54 μs (reported)	27	✗	?
Siemon et al. [141]	n-IMPLY (sim.)	✓	8 RRAM	✓	202 pJ (estimated)	3.61 μs (estimated)	19	✗	?
Zanotti et al. [150]	IMPLY (sim.)	✓	9 RRAM	✓	6.4 nJ (reported)	345 ns (reported)	43	✓	67
Zanotti et al. [150]	IMPLY (sim.)	✓	8 RRAM	✓	518 pJ	560 ns	28	✓	≈ 30
Zanotti et al. [150]	SIMPLY (sim.)	✓	8 RRAM	✓	172 pJ	920 ns (reported at 0.05GHz)	28	✓	$> 4.5 \cdot 10^6$ (no energy trade-off)
Zanotti et al. [132]	n-SIMPLY (sim.)	✓	8 RRAM	✓	2.4 pJ	42 ns	11	✓	$> 4.5 \cdot 10^6$ (no energy trade-off)
Junsangsri et al. [151]	CMOS LiM (sim.)	✓ FET 7 RRAM	41 FET+ 4 RRAM	✗	2.2 fJ (reported-no RRAM energy)	52 ps (reported-no RRAM delay)	-	✓	? (Limited by FET Reliability)

Table 3.5: 1-bit FA performance comparison for the 4 RRAM technologies [132]

	Energy FA Min	**Energy FA Max**
Tech. 1	6.3 pJ	9.9 pJ
Tech. 2	2.5 pJ	5.8 pJ
Tech. 3	2.2 pJ	4.2 pJ
Tech. 4	5.94 pJ	7.28 pJ

simplified RRAM model. Also, the hybrid FET-RRAM LiM solution cannot be implemented on RRAM crossbars limiting its area efficiency. Additionally, the worst- and best-case performances were estimated and compared using all four RRAM technologies considered in this thesis. As expected, the use of lower current compliance results in lower energy consumption. Thus, Technology 3 which uses the lowest current compliance (i.e., $I_C = 100\mu A$) among the considered RRAM technologies, consumes the lowest energy, see Table 3.5. The worst-case performance of the 1-bit FA implemented with n-SIMPLY and Technology 3 are used to estimate the energy for the parallel computation of 512 32-bit full addition operations considering a simple ripple carry architecture. By using the worst-case energy consumption, the overall energy is overestimated thus leaving enough room to consider the additional energy dissipated in circuit components, such as the analog tri-state buffers, that are not considered in the simulations. Also, the use of 512 32-bit full additions corresponds to 4kB of data which commonly corresponds to a memory page size [152]. The performance of the n-SIMPLY implementations were compared to different standard CMOS one [153]–[155] with and without including the penalty introduced by the data movement (i.e., VNB) on a flash memory technology [152] and the results are reported in Table 3.6. Although the CMOS implementations without including the VNB achieve

Table 3.6: Comparison between the proposed FA and a CMOS FA when executing 512 parallel 32-bit FA operations (4kB data)

	Number of Computing Devices	Energy	Delay	EDP	EDP Improvement Normalized to CMOS w/ VNB
CMOS w/ VNB $^{+}$	163840-458752 FET	$\approx 85.5\mu J$	$\approx 2.6ms$	$\approx 2.2 \cdot 10^{-7} Js$	1
CMOS w/o VNB $^{++}$	163840-458752 FET	$\approx 8.8 \cdot 10^{-7} - 107nJ$	$\approx 0.2 - 1.2 \cdot 10^{5} ns$	$\approx 2.5 \cdot 10^{-25} - 1.4 \cdot 10^{-11} Js$	$1.6 \cdot 10^{4} - 8.9 \cdot 10^{17}$
n-SIMPLY Zanotti et al.	18944 RRAM	$\approx 52.8nJ$	$\approx 1.3\mu s$	$\approx 6.8 \cdot 10^{-14} Js$	$3.2 \cdot 10^{6}$
n-SIMPLY Projections $I_C = 10nA^{+++}$ $f = 1GHz^{+++}$	18944 RRAM	$\approx 5.28pJ$	$\approx 0.65\mu s$	$\approx 3.4 \cdot 10^{-18} Js$	$6.5 \cdot 10^{10}$

$^{+}$, $^{++}$, estimates with (w/) and without (w/o) considering the energy and delay overhead introduced by the VNB for reading and writing 4kB data [152]. $^{++}$ Performances of CMOS FA implementations estimated by projecting the time and energies for different 1-bit FA schemes and CMOS technologies (i.e., 0.18 μm, 45 nm, and 10 nm) from the literature [153]–[155], that were combined in a ripple carry configuration. $^{+++}$ Estimated projections with optimized devices [148], [156].

the highest energy efficiency, the n-SIMPLY implementations strongly outperform (i.e., $> 10^{6}$ improvement in EDP) the CMOS when including the energy penalty of the VNB. Additional device-circuit co-optimization would potentially result in additional performance improvements. For instance, the energy projections at lower current compliance [156] and shorter pulses [148] show that the n-SIMPLY implementation could potentially reach the CMOS gate performance without the VNB.

The accumulation or pop counting operation is commonly used to count the number of bits that are "1" in a sequence of bits and is commonly adopted in BNNs. We devised an IMPLY-based implementation which enables to

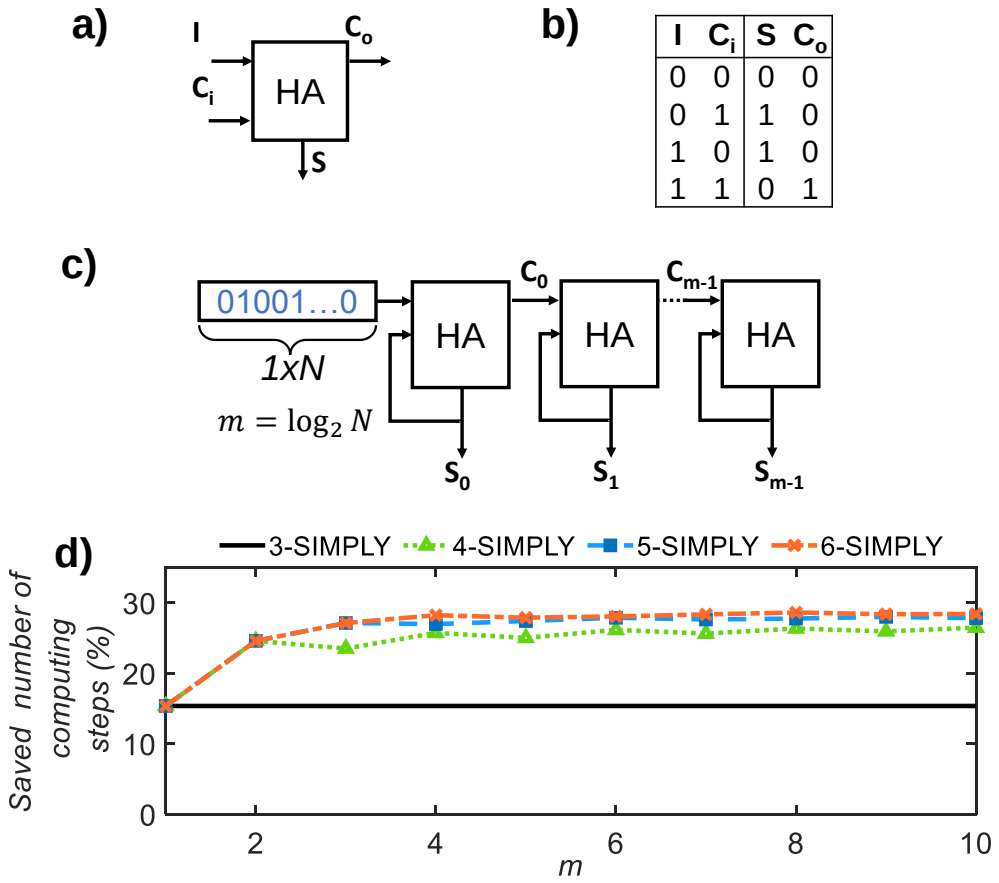


Figure 3.16: a) 1-bit Half-Adder (HA) logic block. b) 1-bit HA truth table. c) Chained 1-bit HAs used to implement the pop count operation. d) Improvement as the number of saved computing steps when using the n-SIMPLY (with n=3,4,5,6) operation with respect to the 2-SIMPLY implementation. Most of the improvement is achieved with $n = 4$.

compute the pop counting operation, an is implemented with a sequence of 1-bit half adders (HAs) (see Fig.3.16a, and b). The developed sequences of computing steps are available in [117], [132]. To accumulate N bits a chain of $\lceil log_2(N) \rceil$ HAs is used, with each HA receiving as inputs its output from the previous step and the output carry (or the input bit) of the previous HA in the chain, as shown in Fig. 3.16c. In IMPLY-based architectures, computing steps are executed in sequential order, thus when accumulating each new input bit first the HA corresponding to the least significant bit (LSB) is executed. The results following HAs are then computed one after the other only if 2^i bits have been summed, where i is the HA index. For instance, the HA in position 2 is activated only after the 4^{th} input bit is being summed since no output carry can be generated by the HA at position 1 before that. Thus, the length of the sequence of computing steps increases exponentially as the number of input bits increases. For instance, $\sum_{i=1}^{\lceil log_2(N) \rceil} 13i \cdot 2^{i-1}$ steps are required when using only the FALSE and 2-SIMPLY operations [117]. Appropriate strategies that exploit the multi-input IMPLY operation can be used to reduce the number of computing steps. Specifically, just by using up to the 3-SIMPLY operation the number of computing steps required for each HA drops to 11. Also, by further increasing the fan-in of the IMPLY operation, a more effective strategy requires optimizing the sequence of computing steps used to calculate the results of groups of serially connected HAs. For instance, by using the 4-SIMPLY operation the result of two serially connected HAs is computed, saving the number of steps required to compute the output carry between the two HAs. The use of the n-SIMPLY reduces the number of computing steps required for computing a series of 2, 3, 4, HAs from 26, 39, and 52, when using only the 2-SIMPLY to 19, 28, and 37 steps. The corresponding sequences are available in [132]. As shown in

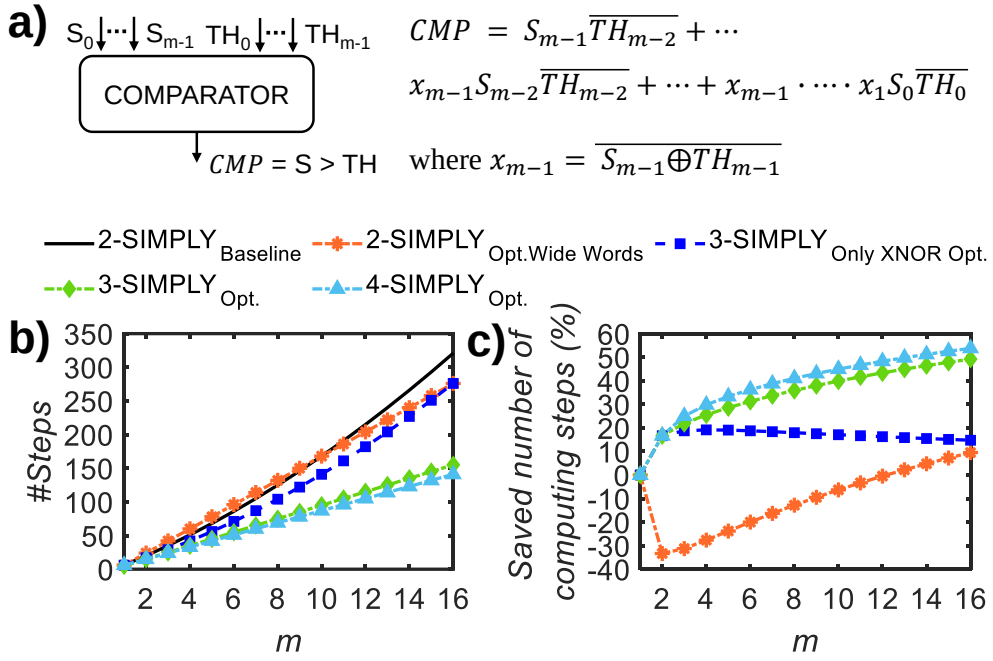


Figure 3.17: a) Comparator logic function computing the result of $S > TH$. b), c) Number of computing steps and percentage of the saved number of computing steps, for an increasing number of compared bits (m) and different implementation strategies based on n-SIMPLY. The use of the optimized sequence of operations exploiting the 3-SIMPLY (3-SIMPLYOpt.) provides most of the step reduction compared to the baseline 2-SIMPLY implementation. Still, additional improvements are achieved by using the 4-SIMPLY operation.

Fig. 3.16d, when considering an increasing number of output bits of the complete accumulator the use of the 3-SIMPLY leads to a 15% reduction of the total number of computing steps. Further increasing the fan-in of the IMPLY operation enables to achieve a >25% step reduction, but only minor improvements are obtained when fan-ins larger than 4 are considered, suggesting that most of the improvements are achieved without compromising the circuit reliability, since increasing the fan-in produces smaller RMs.

Another logic operation, that is commonly used in different applications, (e.g., BNNs) is the comparison between two vectors. Considering the spe-

cific BNNs use case, this operation is used as the neuron activation function and outputs a "1" logic each time its input is larger than a threshold which is a network parameter that is optimized during training and used at inference time. We propose several SIMPLY-based implementations [117], [132], all optimizing the logic function reported in Fig. 3.17a. Specifically, we propose two different 2-SIMPLY implementations, in which the number of computing steps grows exponentially (i.e., $2-SIMPLY_{Baseline}$) and linearly (i.e., $2-SIMPLY_{Opt.Wide\,Words}$) with the number of compared bits, respectively. Thus the former is best suited for comparing a few bits while the latter uses fewer steps when more bits are compared (see dashed orange lines in Fig. 3.17a, b). The details regarding the optimized sequence of computing steps are available in [132]. Also, in this case, increasing the fan-in of the IMPLY operation leads to a reduction of the required number of computing steps. Thus, we evaluated the performance of three different implementations. First, we use the 3-SIMPLY operation to optimize the $2-SIMPLY_{Baseline}$ by reducing the number of computing steps required to compute the XNOR terms only (i.e., x_i in Fig. 3.17a), achieving a $> 10\%$ step reduction (see $3-SIMPLY_{OnlyXNOROpt.}$ in Fig. 3.17). Still, by exploiting all the potentialities offered by the 3-SIMPLY and 4-SIMPLY operations, it is possible to further optimize the number of computing steps. Specifically, two implementations in which the 3-SIMPLY or 4-SIMPLY operations are used wherever possible (i.e., $3-SIMPLY_{Opt.}$ and $4-SIMPLY_{Opt.}$), were developed. As shown in Fig. 3.17, these two implementations achieve significant improvements. The lists of computing steps for each comparator implementation are available in [132]. Also in this case, further increasing the fan-in of the n-SIMPLY would not provide significant additional steps savings.

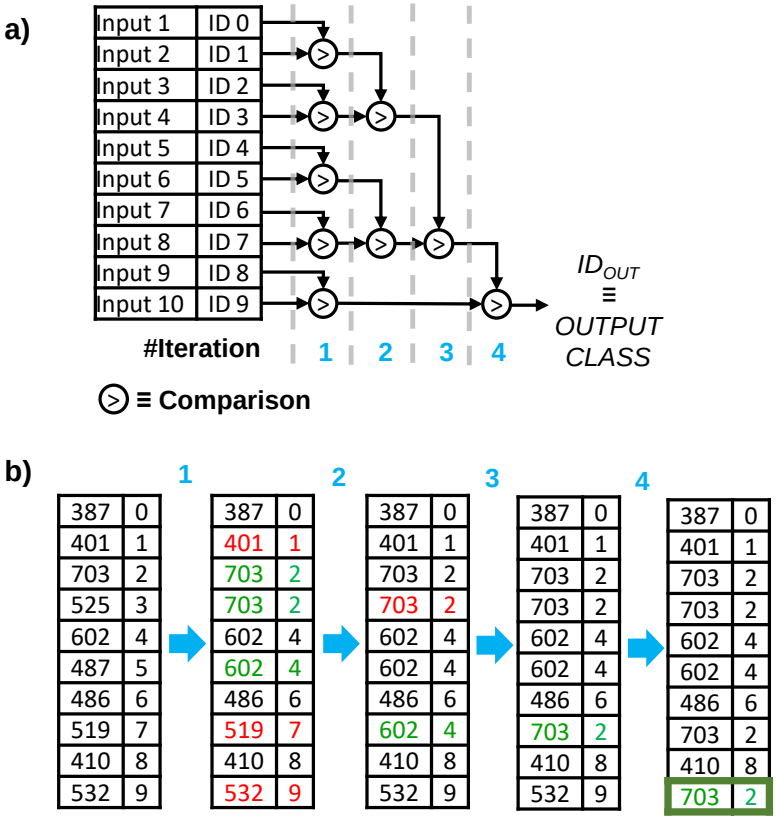


Figure 3.18: a) Sketch of the hard max function used to compute the maximum among 10 input values stored inside the memory. IDn represents the input label. $\lceil log_2(10) \rceil$ iterations are needed. b) Example of the results of each iteration for a specific set of inputs. Each time the first operand is higher than the second its value and ID are moved to the memory location previously storing the second operand (green numbers), otherwise, the stored values are preserved (red numbers). The final result is stored in the last row of the memory array.

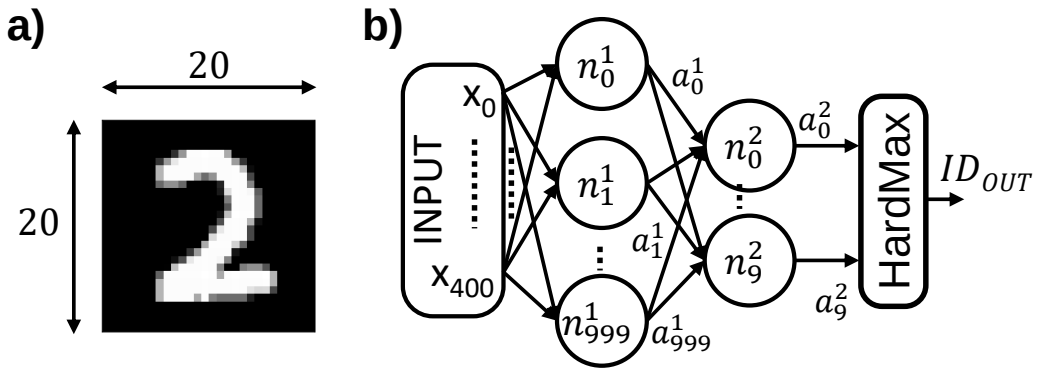


Figure 3.19: a) Sample image from the MNIST handwritten digits dataset [157]. b) Sketch of a multilayer perceptron (MLP) Neural Network considered in this work.

The last studied complex logic function implemented on the SIMPLY architecture is the hard max function which determines the maximum value among a group of elements. Such a function is also commonly employed in BNNs to determine the predicted output class during inference. In Fig. 3.18a sketch representation of the required operations when applying the hard max function to a group of 10 elements is shown. These consist of $\lceil log_2(10) \rceil$ iterations, each comprising the comparison between pairs of elements, and the subsequent copy of the maximum value and corresponding identifier in the lower position in the array among the two compared elements. Therefore only the copy and the comparison operations previously described are used. An example is shown in Fig. 3.18b. Thus, at the end of the $\lceil log_2(10) \rceil$ iterations the output is stored at the bottom of the array.

IMPLY-based Binarized Neural Network inference

The rising popularity of artificial neural networks has promoted the design of new in-memory computing solutions based on RRAM devices. Although the use of RRAM devices to implement analog hardware accelerators of the VMM operation that is extensively used in DNNs is a very promising so-

lution, due to the current limitations showed by real devices, such as the high C2C variations and RTN, its design complexity is still very high due to the difficulty in storing enough bits in a single RRAM device [158]. Thus, other approaches, such as BNNs, which limit the bit precision of the network weights and activations at inference time are currently being studied in the literature [57], [85]–[87], [159]. It was demonstrated that for some classification tasks, the use of binary weights and activations results in a limited accuracy drop despite the considerable reduction in the computational complexity [85]. Also, the use of binary weights and activations converts the multiply and accumulate (MAC), and the activation function operations to simple logic operations. Specifically, the VMM becomes a bitwise XNOR operation between the inputs and each neuron weights, the accumulation translates to a pop counting operation, the batch normalization to a full addition, while the activation function into a comparison with a threshold. As discussed in the previous section these can be realized on the SIMPLY architecture exploiting its high degree of reconfigurability and computation parallelism realizing an energy-efficient hardware solution for edge computing applications.

Thus, to benchmark the performance of the SIMPLY architecture on a BNN inference task, we implemented and trained the neural network shown in Fig. 3.19b, which consists of a single hidden layer with 1000 neurons and ten output neurons. Specifically, such network was trained using the DoReFa-Net algorithm [160] on the MNIST handwritten digits dataset [157] which consist of 20x20 pixels images that were converted to black and white such as the one reported in Fig. 3.19. The dataset was divided into groups composed of 9500, 2500, and 2000 images for training, validation, and testing, respectively. After training the network achieves a 91.4% accuracy, which is

comparable to other results reported in the literature [159].

The trained network parameters were mapped to the corresponding resistive states of RRAM devices that are divided into two crossbars, one for each network layer, which include also auxiliary RRAM devices. Then to evaluate the energy and delay of different SIMPLY-based implementations the complete set of operations required to classify an input image was emulated considering for IMPLY or FALSE operations the worst-case energy estimates for the specific input combination, and considering the data from Technology 3. Since the energy per operation depends on the specific input image, the energy was evaluated on the complete test set. Also, in this case, the energy is overestimated to account, as a first-order approximation, also for any additional peripheral circuit component that is not included in the circuit simulations. The inference latency was evaluated considering a 0.5 GHz clock frequency and two extreme cases, when all the operations in each layer are executed in parallel (i.e., referred to as "parallel" in this thesis) and when each operation is executed one after the other (i.e., referred to as "serial" in this thesis). All the proposed SIMPLY implementations considerably improve the energy efficiency and reliability compared to the corresponding conventional IMPLY-based implementations. In fact, due to the effect of LSD the network parameters that are commonly written once after training and only used at inference time, would quickly lead to accuracy drop or require periodic memory refresh cycles which would otherwise affect the network efficiency.

The performance of the SIMPLY-based solutions was benchmarked against a conventional low-power embedded system implementation of the same 1-hidden layer multilayer-perceptron BNN from the literature [159], which represents the state-of-the-art of low-power devices for edge computing. As

Table 3.7: Benchmark of the performance of SIMPLY architectures on an inference task performed on a multi-layer perceptron neural network

Implementation	Number of computing steps	Energy (min-avg.-max)	Latency	avg. EDP	EDP improvement with respect to embedded system
Embedded system [159]	-	$5.37mJ$	$17.35ms$	$9.3 \cdot 10^{-5} Js$	1
2-SIMPLY Serial* [149]	46122570	$7.2 - 7.6 - 8.0\mu J$	$184.5ms$	$1.3 \cdot 10^{-6} Js$	71.5
2-SIMPLY Parallel* [149]	174933	$7.2 - 7.6 - 8.0\mu J$	$0.700ms$	$5.0 \cdot 10^{-9} Js$	$1.86 \cdot 10^{4}$
n-SIMPLY Serial	34016840	$4.79 - 5.65 - 6.5\mu J$	$136ms$	$7.7 \cdot 10^{-7} Js$	121
n-SIMPLY Parallel	129095	$4.79 - 5.65 - 6.5\mu J$	$0.516ms$	$2.9 \cdot 10^{-9} Js$	$3.2 \cdot 10^{4}$
n-SIMPLY Parallel Proj. Ic=10nA [156]	129095	$1.04 - 1.14 - 1.29nJ$	$0.516ms$	$7.6 \cdot 10^{-13} Js$	$1.2 \cdot 10^{8}$

* The estimate from [149] where corrected by considering the delay and energy of the input mapping on the array.

shown in Table 3.7, while the 2-SIMPLY parallel implementation improves both the energy efficiency and latency, the serial implementation requires a longer inference time than the benchmark solution. Still, especially in ultra-low power applications, the energy is the most important constraint, while a minor increase in the computation latency could be tolerated. Thus, a more appropriate metric to evaluate the system performance is the EDP, and all SIMPLY solutions improve the EDP with respect to the benchmark solution. Further improvements are achieved when using the n-SIMPLY considering n up to 4. As shown in Table 3.7, the use of n-SIMPLY reduces the computing latency by a remarkable 26%. Also, by reducing the number of IMPLY and FALSE operations, the use of n-SIMPLY leads to an additional 25% reduction of the dissipated energy, therefore leading to a considerable improvement of the EDP of both the parallel and serial implementations. The above results, together with projections considering a device with lower current compliance ($I_C = 10nA$ [156])), underline the remarkable performance of the SIMPLY

architecture, that together with its intrinsic possibility to reconfigure the logic operations that are computed in-memory, is a very promising candidate for ultra-low-power hardware accelerators.

3.3 Low-bit precision neural networks

Due to the increasing demand for energy-efficient NNs hardware accelerators, also hardware-specific (i.e., lacking the possibility to reconfigure the kind of operation accelerated in-memory) accelerators are being investigated. An alternative solution to BNNs that can be reliably implemented with RRAM devices are LBPNNs [87], [134], [161], which use only a few bits (i.e., one or more) to represent the NN weights and activations, and compute the results of the VMMs in the analog domain. The use of more than one bit to represent the activations helps in achieving higher accuracy than BNNs [160], still limiting the requirements imposed on the memory elements. In fact, compared to full-precision (32 bit) NNs implementations, LBPNN are more robust to RRAM devices nonidealities (e.g., RTN, variability, drifts) [57], [134], [162]. Still, the design of LBNN is non-trivial, and the effects of RRAM devices' nonidealities can influence the classification accuracy if not appropriately considered during the design phase. In this section, we leverage the UniMORE RRAM compact model to determine by means of full circuit simulations clear performance and reliability trade-offs for an LPBNN implementation, devising an appropriate device-circuit co-optimization strategy.

The analysis is performed on a LBPNN implementing a simple handwritten digits recognition task considering a scaled version of the MNIST dataset where only the 3 and 8 digits are classified and the images are scaled down to 8x8 pixels. This approximation helps in reducing the size of the NN, lim-

iting the computational burden introduced by full-circuit simulations. Still, by considering digits that are similar, the classification task remains difficult because even a single bit flip could potentially lead to a misclassification, and thus represents a worst-case condition for the network. The network is a multilayer perceptron composed of 2 hidden layers with 40 and 10 neurons, respectively, and one output neuron, and was trained using the DoReFa-net algorithm considering a single bit for the weights and 4 bits for the activations. The use of 4-bits for the activations leads to high classification accuracy while limiting the circuit complexity [160]. The network achieves 94.7% accuracy in software.

The circuit simulations were performed using the compact model calibrated on Technology 3, and considering the nominal $50ns$ programming voltage pulses from [129]. Since in the designed network, a single bit is used to represent each weight, a single set or reset voltage pulse is used to program each device either in LRS or HRS, respectively. The nominal LRS resistance is $\approx 9k\Omega$, while different HRS resistances distributions, and thus memory windows, were obtained by using different reset voltages (i.e., -2.3V, 2.6V, and -3V).

Each layer of the LBPNN was implemented on a 1T1R array using the architecture shown in Fig. 3.20a, where two 1T1R devices placed in adjacent columns in complementary states are used to represent a single network weight. As shown Fig. 3.20b, when a device pair is in the (LRS, HRS) configuration it represents a +1 while a -1 when in the opposite configuration. During inference a column decoder, biases each pair of crossbar columns representing the weights of a single neuron with $+V_{READ}$ and $-V_{READ}$ (i.e., $V_{READ} = 200mV$ in this work). A row decoder drives the bitlines with the input activations, thus turning on the transistors only when the input acti-

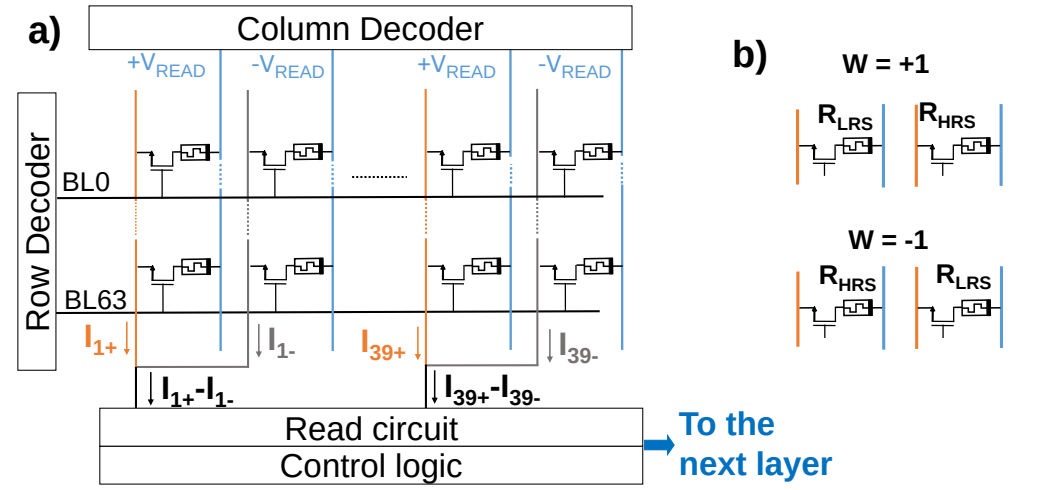


Figure 3.20: a) Single layer of a LBPNN implemented on a 1T1R devices crossbar array. Peripheral circuits are used to deliver the input activation to the crossbar rows, while the columns are driven with either $+V_{READ}$ or $-V_{READ}$. The result of the VMM is encoded in the difference between the current flowing into adjacent columns which are digitally converted by an appropriate read circuitry. b) Weight encoding into RRAM devices resistances. two devices in the opposite resistive state are used to represent a single bit.

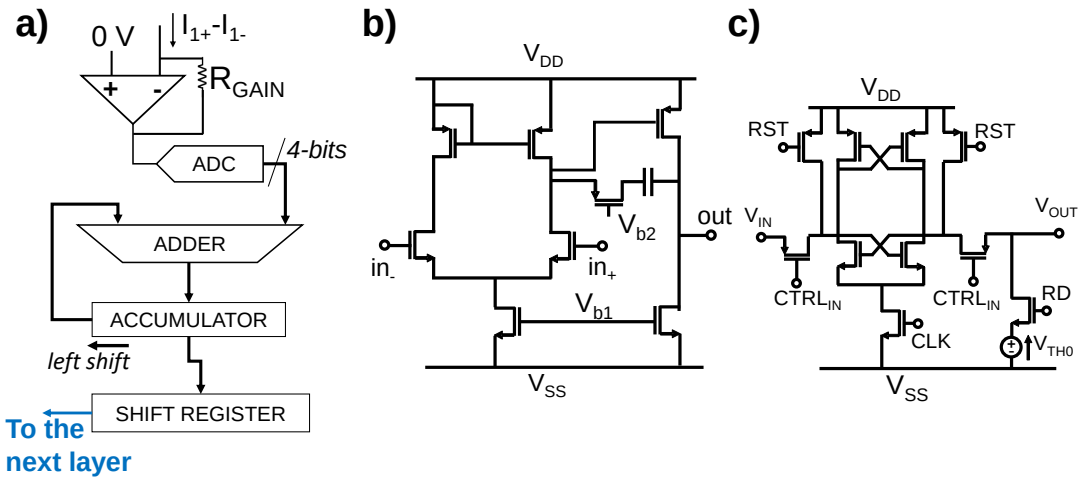


Figure 3.21: a) Functional diagram of the neuron circuit. b) Operational amplifier used for each neuron's transimpedance amplifier. c) SA circuit used as a comparator in the 4-bit Flash ADC implementation.

vation is 1. Thus, when the input activation is 0, the transistor is off and only a small leakage current flows in both the vertical lines connected to each transistor source terminal (orange and gray columns), while when the input activation is 1, a higher I_{LRS} current is introduced either in the orange or gray column depending on the specific synapse value, e.g, if the synapse is $+1$, I_{LRS} and I_{HRS} are added to the orange and blue columns, respectively. Thus, by driving in parallel all the bitlines the VMM is computed in one step, and the sum of each positive product $((+1) \cdot (+1) = +1)$ and the sum of each negative product $((+1) \cdot (-1) = -1)$ of each neuron are encoded in the current flowing in each orange and gray lines, respectively. The orange and gray columns are short-circuited and connected to a virtual ground terminal before being connected to a read circuit, thus effectively computing the difference between the sum of the positive and the sum of the negative products. The read circuit is depicted in Fig. 3.21a, and is composed of a transimpedance amplifier that converts the current to a voltage which is sampled and converted by a 4-bits ADC. Since the activations are encoded into 4-bits, the input split method is employed [161], [163], which consists of separately and sequentially computing the results of the VMM for each bit, starting from the most significant bit (MSB). So, in the read circuit, the intermediate results are accumulated in a register and between the VMM computation of each new input bit the result is left-shifted to perform a x2 multiplication. After the four bits of the activations have been multiplied, the result is truncated to 4-bits in the hidden layers or to only the MSB in the output layer. The transimpedence amplifier was implemented with a 2-stage operational amplifier designed in a 45 nm technology [139] (see Fig. 3.21b). Also, the ADC is implemented as a Flash converter using 15 energy-efficient SA (see Fig. 3.21c) as comparators. Such circuits were included in the circuit

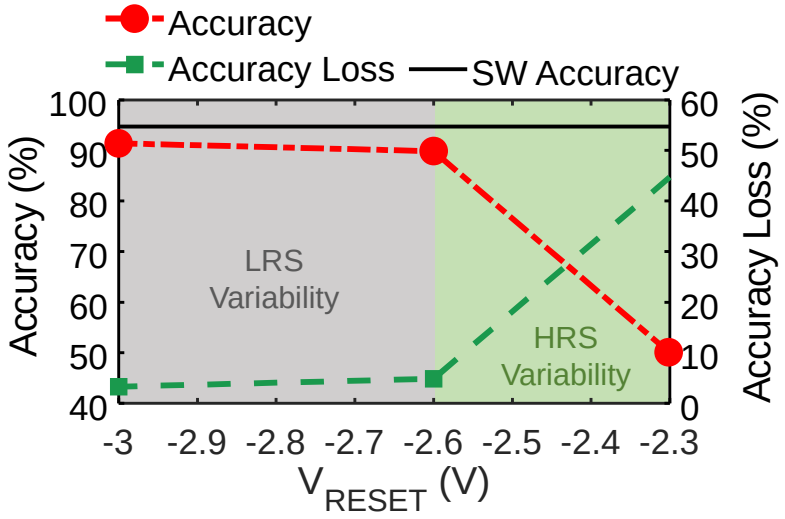


Figure 3.22: Accuracy (red circles, left y-axis) and accuracy loss (green squares, right y-axis) as a function of V_{RESET}. The baseline software accuracy (black line) is reported. Below and above $-2.6V$ the accuracy loss is dominated by HRS and LRS variability respectively.

simulations and their impact was accounted for in the performance estimates.

Using full-circuit simulations the effect of variability, RTN, and different memory windows on the classification accuracy and energy efficiency was analyzed on a batch of 128 images (i.e., 64 "3" and "8" digits, respectively). The inference and weight mapping phases were both analyzed. During the weight mapping phase, all RRAMs are first reset and then V_{SET} is delivered to half of the devices, due to the used weight encoding. The effect of variability and RTN results in noise to be added to the current that is input to the read circuit (see Equations (3.2), (3.3), and (3.4), where $Noise_{VAR.}$ and $Noise_{RTN}$ are the equivalent noise sources due to variability and RTN, respectively), potentially introducing errors in the output of each neuron, even a sign flip [134], eventually resulting in misclassifications. As expected, the tolerance of the network implementation to such noise sources is strictly connected to the memory window and thus the employed reset voltage. As shown in Fig. 3.22,

for sufficiently large memory windows (gray area) the network retains almost the software level accuracy (black line), and the accuracy losses are mainly due to the LRS variability since $I_{HRS} << I_{LRS}$. On the contrary, when smaller memory widows are considered (green area), the network is more sensitive to HRS variations and to time-varying RTN fluctuations, leading to considerable accuracy losses, due to read errors. Thus, in this condition, the main culprit of the accuracy losses is the HRS variability.

$$I_{i+} = \sum_{p} I_{LRS} + \sum_{n} I_{HRS} + Noise_{VAR.} + Noise_{RTN} \tag{3.2}$$

$$I_{i-} = \sum_{p} I_{LRS} + \sum_{n} I_{HRS} + Noise_{VAR.} + Noise_{RTN} \tag{3.3}$$

$$I_i = I_{i+} - I_{i-} \tag{3.4}$$

Adopting more negative reset voltages also reduces the power dissipation both during the weight mapping and at inference time, as clearly shown in Fig. 3.23a. As shown in Fig. 3.23b, during a reset pulse the resistance of the device changes in just a few ns to an HRS, with R_{HRS} increasing exponentially with more negative reset voltages. Thus, during most of the pulse duration, the device is already in HRS, resulting in much lower energy dissipation. At inference time larger V_{RESET} translates to higher R_{HRS}, which leads to lower I_{HRS} values and thus lower energy dissipated in the network synapses, as shown in Fig. 3.23c (green dashed lines). However, also the energy associated with the neurons adds up to the total inference power. As shown in Fig. 3.23c, the neuron contribution does not scale with V_{RESET}, and strongly dominates (i.e., $\approx 83\%$ of the total power when $V_{RESET} = -3V$) the total inference power. This is caused by the used transimpedence amplifier design, which uses an operational amplifier with a class A output. Thus, the output stage is biased considering the maximum

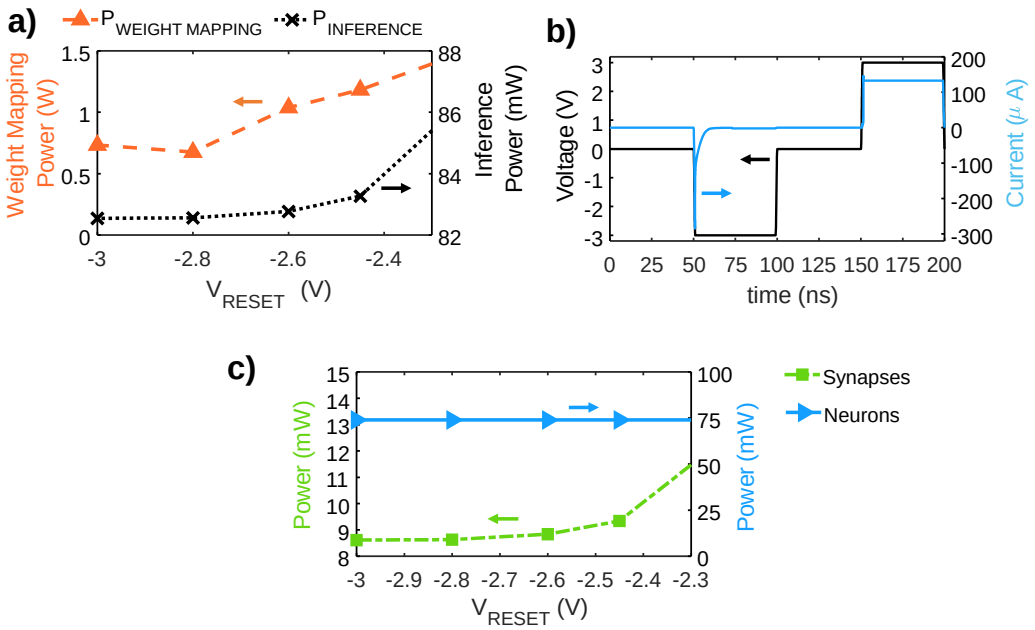


Figure 3.23: a) Power dissipation during inference (black crosses, right y-axis) and weight mapping (orange triangles, left y-axis) as a function of V_{RESET}. b) Voltage (black curve, left y-axis) and current (blue curve, right y-axis) in a RRAM device during programming. b) Dissipated power in synapses (green squares, left y-axis) and in the neurons (blue triangles, right y-axis) as a function of V_{RESET}.

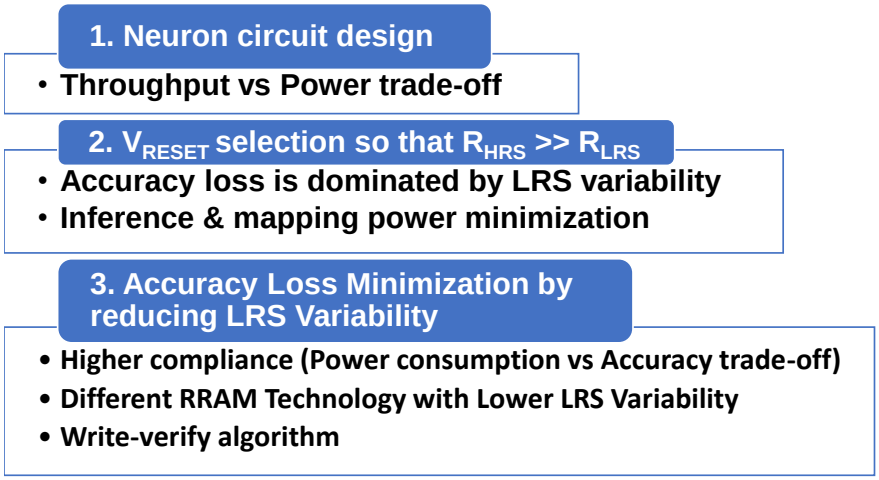
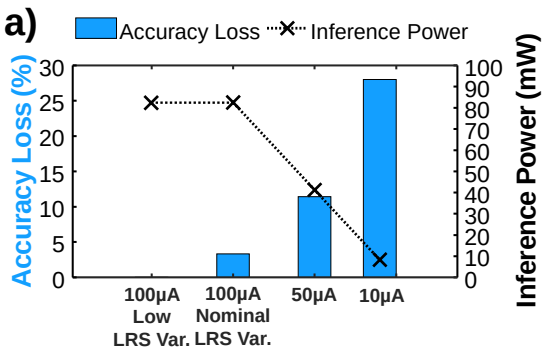


Figure 3.24: Flow-chart describing the proposed design strategy for LBPNNs.

and minimum possible currents flowing. Therefore, the use of more energy-efficient neuron designs would be extremely beneficial in reducing the total inference power. A few examples are class AB operational amplifiers with low bias currents or current conveyor II circuits [164] which could indeed result in significant energy savings, however highlighting the existence of a trade-off between the chip area, neuron speed (i.e., the maximum number of inferences per second), and energy efficiency.

Thus, such trade-offs can be used to determine appropriate design strategies used to optimize the energy efficiency, accuracy, and throughput (i.e., the number of inferences per unit time) of the hardware accelerator. As sketched in the flow-chart in Fig. 3.24, the design procedure can be divided into three steps. Since the throughput of the network is mainly limited by the speed of the circuit implementing the neuron, the first step should focus on designing an energy-efficient operational amplifier that meets the desired throughput requirement. Thus, the slew rate of the operational amplifier must be sufficiently high thus highlighting an efficiency throughput trade-



b)

Accuracy	91.4%
Inference Energy (nJ)	29.7
Weight Mapping Energy (nJ)	36.6
Max. Throughput (Inference/s)	$2.8 \cdot 10^6$
Efficiency (Inference/s/W)	$3.37 \cdot 10^7$

Figure 3.25: a) Accuracy loss (blue bars) and inference power (black crosses) as a function of the current compliance. Inference power linearly increases with the current compliance, while the accuracy loss is considerably higher for lower current compliance values. Two implementation at $I_C = 100\mu A$ with nominal $\sigma R/R$ (0.041) and a lower one (0.01), are reported.b) Performance of the low bit-precision neural network considered in this study, when using Technology 3 and $V_{RESET} = -3V$ and $I_C = 100\mu A$.

off. In the second step V_{RESET} is determined to guarantee a large enough memory window, also accounting for variability, so that the accuracy losses are mainly caused by the LRS variability. Finally, the third step minimizes the accuracy losses either by:

- Using higher current compliance. As shown in Fig. 3.25a, due to the higher LRS variability commonly showed by RRAM devices at lower I_C, the accuracy losses are inversely proportional to the current compliance. However, the use of a larger current compliance value also results in higher mapping and inference power.

- Using a different RRAM technology which shows less variability at the same I_C (see $100\,\mu A\,Low.\,LRS\,Var.$ in Fig. 3.25a, which illustrates the results that are achieved when considering a lower LRS variability).

- Employing write-verify algorithms [72], [73] which limit the standard deviation of the LRS distribution, however at the cost of an increased

chip area and power dissipation.

By employing the devised strategy, a network targeting a throughput of at least a $10^6 inference/s$ was designed, and the estimated performances are reported in Fig.3.25b. Specifically, the network was evaluated considering $V_{RESET} = -3\,V$, the nominal $I_C = 100\mu A$, $V_{READ} = 200\,mV$. The read step requires 40 ns for each bit of the input activations. The network satisfies the design constraint reaching a throughput of $2.8 \cdot 10^6 inference/s$ and accuracy of 91.4%, while consuming $\approx 30\,nJ$ per inference, thus showing high energy efficiency and robustness features.

3.4 Merging multiple computing paradigms on the same memory array

Although the design of operation-specific hardware accelerators, for instance the LBPNN accelerator described in this thesis, can provide very high performance when executing a specific task, still, for resource-constrained and low-cost devices for edge computing, adding the possibility of reconfiguring the type of operations that are accelerated in-memory, would lead to better use of the scarce resources [165]. For instance, the SIMPLY architecture provides the possibility of reconfiguring the operations that are computed in memory just by changing the programmed sequence of IMPLY and FALSE computing steps, however at the cost of a lower efficiency compared to solutions that are optimized only for a single task. Even though a BNN inference task can be executed on the SIMPLY architecture, when properly designed, optimized hardware accelerators based on resistive memory technologies [86], [161], [166]–[168] computing in analog the MAC operation can provide a better performance, however at the cost of reduced reconfigurability. Thus, in

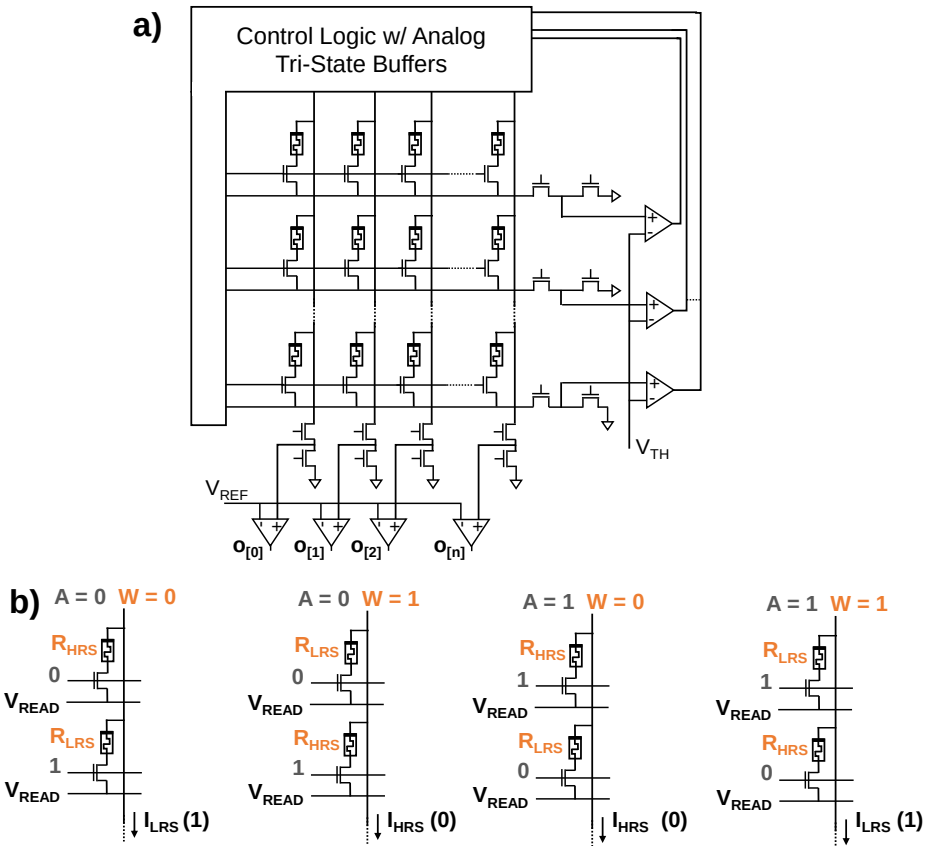


Figure 3.26: a) In-memory computing architecture merging the SIMPLY and analog vector matrix multiplication frameworks on a single 1T1R array. b) Sketch showing how the XNOR operation (i.e., activation and weight multiplication in BNNs) is computed in analog on the 1T1R crossbar using two RRAM devices to encode the value of a single weight.

this section, we propose a solution for ultra-low-power edge computing hardware by enabling the coexistence of both SIMPLY and the analog VMM for BNNs on the same crossbar array, and describe its design trade-offs.

Differently from the SIMPLY crossbar implementation discussed in Section 3.2, the proposed architecture is based on 1T1R arrays, and require the SAs to be connected both at the crossbar rows and columns, and a sketch representation is available in Fig. 3.26a. Although this increases the circuit

area, it improves the performance of SIMPLY by increasing the computation parallelism also when computing operations between devices that reside in the same crossbar columns, thus reducing the time required to copy data between rows. To execute an IMPLY between devices on the same column, the selection transistors of the specific rows are turned on, and V_{READ} is delivered to the crossbar rows and the output voltage is compared at the column where the devices are located. Although the selector transistor is subject to different source-bulk biasing when executing operations on the rows and columns of the array, the body-effect can be limited by using a small V_{READ} and controlling the gates of the transistors with sufficiently high voltages.

The specific advantage of the architecture in Fig. 3.26a, is that it enables the computation of the VMM operation for BNN in the analog domain, by mapping each weight of the BNN to RRAM devices located on the same column and adjacent rows, with the pair of devices programmed in opposite resistive states, as shown in Fig. 3.26b. The input activations are delivered to the gates of the selector transistors using a pair of complementary signals (see Fig. 3.26b), while V_{READ} is delivered to the source gates of the selector transistors. By doing so, each product between the input activation and the neuron weight, which corresponds to an XNOR operation in BNNs, is computed and a current proportional to the result flows in the crossbar column. However, while commonly in NN accelerators a virtual ground terminal is available at the end of each column by using an operational amplifier, to limit the neuron area and improve the energy efficiency the computation is performed with a simple operational amplifier and a pull-down resistor (R_{PD}) implemented with the same FET used to implement R_G in SIMPLY. Also current-mode sense amplifiers could be employed, however leading to lower efficiency [169], [170]. When computing the product between multiple neuron

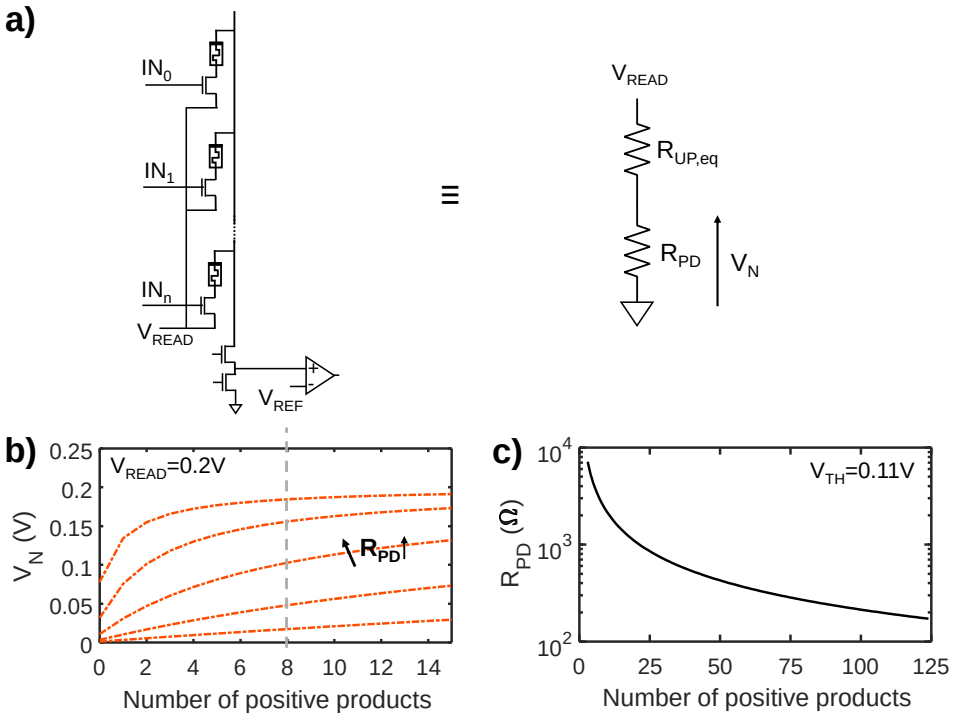


Figure 3.27: a) BNN vector-vector multiplication between the input activations and a single neuron's weights (i.e., single crossbar column). The equivalent circuit is shown on the right. b) Trends of V_N for different values of R_{PD} values and an increasing number of +1 products results. When the sum of product is higher than half the number of input activations (15), the comparator switches to +1. Thus, different R_{PD} values result in different voltage thresholds. c) Optimal value for R_{PD} when the number of input devices that are read in parallel is increased and a fixed threshold (V_{TH}) is considered.

weights and activations, a voltage divider is formed between the equivalent parallel resistance of the active 1T1R devices and the resistor R_{PD}, as shown in Fig. 3.27a. Thus, the relation between the number of positive products and the voltage at the input of the SA is not linear. Although this introduces some limitations, He et al. [161] demonstrated that high inference accuracy can be achieved with this method even when considering the effect of variability. Nonetheless, the nonlinear relation introduces a limitation on the number

of products that can be computed in parallel, and this number depends on R_{PD}, R_{HRS}, R_{LRS}, and the threshold voltage of the SA. Considering as an example the case where 15 products are computed in parallel, increasing the value of R_{PD} increases the voltage at the input of the SA (V_N) for the same result of the products, see Fig. 3.27b. Also, higher R_{PD} values worsen the linearity and quickly saturate to V_{READ} however too low values considerably reduce the dynamic range at the input of the comparator, resulting in higher error rates. The result of more products could be computed in parallel when using low R_{PD} values when considering a fixed threshold voltage, see Fig. 3.27c, however increasing the required FET size and making the circuit more susceptible to noise and line parasitic resistances. Therefore, all the products of the VMM are divided between multiple computing steps, using the method described in [161], [163]. In each network layer, before applying the activation function the partial results of the products are accumulated. By avoiding the need of programming RRAM devices and by reducing the number of computing steps during each VMM, this approach results in higher energy efficiency compared to the SIMPLY implementation. In fact, in SIMPLY the MAC operation requires a very high number of computing steps, which increases exponentially with the number of inputs of a network layer. The activation (i.e., the comparison with a threshold) and the hard max functions on the contrary require a much smaller number of computing steps.

Thus, by using multiple crossbars such as the one in Fig. 3.26a, an entire BNN can be efficiently implemented on a single chip, by computing the result of the VMM in analog while implementing the partial results and the activation functions using SIMPLY. We estimated the performance of the proposed solution on the same BNN inference task considered in Section 3.2.4 for the SIMPLY implementations at different fan-in. The estimated performance

Table 3.8: Comparison of the performance of the SIMPLY and the SIMPLY with analog VMM acceleration on a BNN inference task

Implementation	Average Energy	Latency	avg. EDP
2-SIMPLY Parallel	$7.6\mu J$	$700\mu s$	$5.0 \cdot 10^{-9} Js$
n-SIMPLY Parallel	$5.65\mu J$	$516\mu s$	$2.9 \cdot 10^{-9} Js$
SIMPLY with Analog VMM	$231nJ$	$31.6\mu s$	$7.6 \cdot 10^{-13} Js$

reported in Table 3.8, highlight the remarkable performance improvements achieved by the proposed architecture merging the two computing paradigms [171]. Compared to the 2-SIMPLY and n-SIMPLY fully parallel implementations, the addition of the analog computation of the VMM operation leads to a $> x10$ energy and latency reduction, leading to more than two orders of magnitudes EDP improvement, proposing the developed in-memory computing architecture as a promising reconfigurable hardware accelerator solution for edge computing.

3.5 Related published works

The results presented in this Chapter were published during the PhD program in the following journals [82], [117], [132], [149], [150], [171]–[173], international conferences [133], [134], [174]–[178]. Also, the work on SIMPLY resulted in a patent application [179].

Chapter 4

Study of RTN-based TRNGs

The diffusion of smart devices has increased the amount of sensitive information that is transferred over the internet and wireless networks, or that is stored in digital memory chips. To protect these data from malicious attacks, new low-cost and energy-efficient, highly random, hardware security primitives such as TRNGs [55], need to be developed and investigated. Designing devices where high-quality TRNGs circuits reside directly on the same chip results in improved security to external attacks [95], [180], such as side-channel and physical attacks. Therefore, while the intrinsic stochasticity presented by RRAM devices is commonly a nuisance for in-memory computing architectures, it can be exploited as a high-quality source of entropy. A promising entropy source is the RTN signal shown by RRAM devices. In fact, the capture and the emission times of defects in the RRAMs are intrinsically random. Such defects include oxygen vacancies or metallic atoms from the electrodes penetrating into the dielectric and are commonly introduced as a side-effect of fabrication processes or as the result of a soft breakdown of the dielectric. Also, RTN signals manifest as noise on the device current, which is read by biasing the device with small read voltages (i.e., $\approx 100mV$).

Since there is no need to switch the RRAM device during the RTN signal collection a high energy efficiency can be achieved. Still, the characteristics of RTN signals are hard to control. For instance, a single defect produces a two-level RTN signal in which the current switches between two levels, while multiple defects result in a multi-level RTN, but the number of defects introduced in the RRAM MIM structure is hard to control. Also, due to the effect of trapped charges and their electrostatic interactions with ions, the defects sustaining the RTN signal may remain stuck in a specific state, resulting in the disappearance of the RTN signal for periods that are much longer than the capture and emission times, compromising its stability. These effects, together with other characteristics, are directly linked to the stack of materials building the MIM structure, and the used fabrication processes. Although on the one side it is important to introduce appropriate solutions during the TRNG circuit design to compensate for possible limitations of the RTN signal shown by a specific technology, on the other side it is important to identify which combination of materials and fabrication processes is more prone to produce better quality RTN signals [181].

Therefore, in this Chapter, the main characteristics of low-cost and low-power TRNGs circuits exploiting RTN of RRAM devices are introduced in Section 4.1, and device-circuit co-optimization strategies guided by the experimentally measured RTN signals from different RRAM technologies are discussed in Section 4.2.

4.1 RTN-based TRNGs circuits

Ideally, a TRNG circuit should provide high randomness, energy-efficiency, throughput (i.e., number of bit/s of the generated bitstream), and security

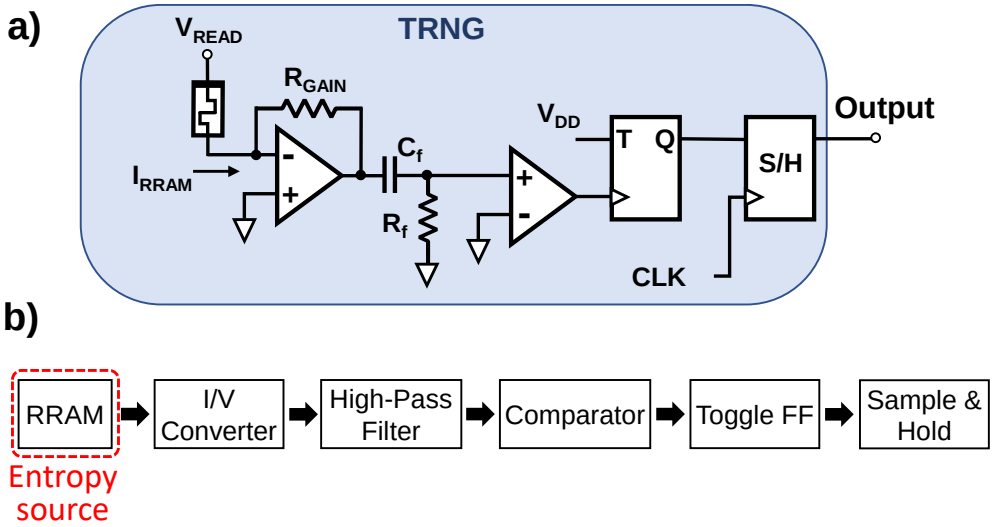


Figure 4.1: RTN-based TRNG circuit exploiting an RRAM device as the entropy source. A read voltage (V_{READ}) is delivered to the RRAM device. The resulting current is converted to a voltage using a transimpedence amplifier with an appropriate gain (R_{GAIN}). Such current is low-pass filtered to isolate the RTN component. The resulting signal is digitized with a comparator and de-biased with T-FF. Finally, the signal is appropriately sampled with a sample and hold circuit. b) Functional blocks composing the TRNG circuit in a).

against attacks, that are aimed to break the randomness (e.g., RF and temperature attacks), or spoof the generated sequence (e.g., reading the current absorbed on V_{DD}). RTN signals are a high-quality entropy source, that enables to design highly random, energy-efficient, TRNGs circuits with high security against attacks depending on the circuit design that is employed [96]. An example of a simple circuit [74], [180]–[182] for RTN-based TRNGs and its core building blocks are shown in Fig. 4.1a, and b, respectively. A small V_{READ} voltage is applied to the TE of the RRAM device while its BE is connected to a virtual ground node. Thus, a current including the RTN fluctuations is input to a transimpedance amplifier, which performs the I/V conversion and amplifies the signal. RTN fluctuations are superimposed to a DC component which can also drift over time depending on the specific de-

fect properties, their spatial disposition, and environmental conditions [183]. Thus, a high-pass filter is used to remove the DC component and to isolate the RTN noise contribution. A comparator converts the noise signal to digital values. Since a disparity in the defects' capture and emission times would result in a disparity in the number of bits that are zero or one in the output bitstream, this bias needs to be removed. A solution is to consider that the sum of the capture and emission time is random ($\tau_c + \tau_e$) to determine whether the output bit should flip state [94]. Thus, the output of the comparator is used as the clock of a T-FF with its input T always at "1" logic. Finally, a sample and hold (S/H) circuit, is used to sample the output random bit with a clock (CLK) period that must be lower than the average sum of the capture and emission times. Using a higher clock frequency would produce longer intervals where the output bitstream remains constants either to zero or one, compromising the randomness.

Thus, the main limitation of RTN-based TRNGs is their low throughput, which is directly related to the type of defects and their specific capture and emission times that are commonly in the order of the tens or hundreds of ms. Although the throughput could be increased by using more TRNG in parallel, we propose an alternative solution in which the RTN-based TRNG is combined with a pseudo-random number generator (PRNG) [182]. A PRNG is a deterministic algorithm that is able to generate a sequence with good randomness characteristics. However, the sequence is completely determined by the initial condition (i.e., seed), and after an entire period, which length depends on the employed algorithm, the sequence starts to repeat itself, therefore, losing its randomness properties. Two examples of PRNG are the linear feedback shift register (LFSR) [184] and the nonlinear feedback shift register (NLFSR) [185]. Both solutions are similar and comprise a shift regis-

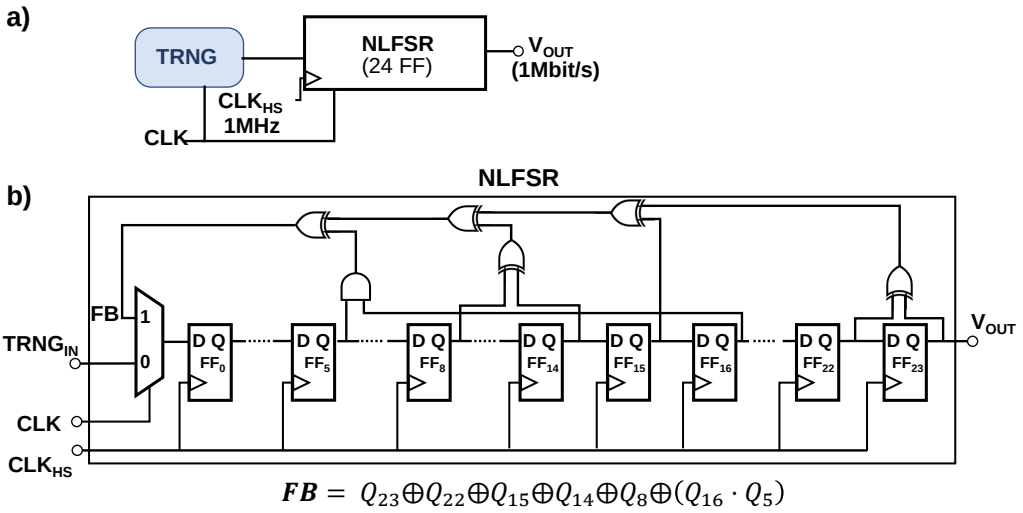


Figure 4.2: a) High-throughput TRNG where a NLFSR is employed to increase the output bit rate of the RTN-based TRNG, which is used to periodically reseed the state of the NLFR. b) Considered implementation of a NLFSR with 24 D-FFs. when a new bit from the RTN-based TRNG is available it is fed to $D - FF_0$ in place of the feedback bit (FB).

ter with the input of the first Flip-Flop that is a logic function of the current internal state of the register (i.e., feedback function). However, the NLFSR achieves better randomness performance by introducing a nonlinearity in the feedback function. The throughput of the NLFSR is adjusted by changing its input clock frequency. For instance, by using a 1 MHz clock a 1 Mbit/s bitstream is generated. The period of the NLFSR depends on the number of Flip-flops (n_{FF}) composing the shift register and the specific feedback function. By using the optimized feedback function [186], the longest possible period can be achieved, and is equivalent to $(2^{n_{FF}} - 1)$ bits.

Thus, a solution combining the RTN-based TRNG and a 24-Flip-Flops NLFSR was devised in [182]. Such circuit, sketched in Fig. 4.2a, combines the high randomness of the RTN-based TRNG and the flexibility and higher throughput of the NLFSR, to realize a high throughput, low-cost, and energy-

efficient TRNG. The output of the RTN-based TRNG is used to periodically change the internal state (i.e., the seed) of a modified version of the NLFSR, shown in Fig. 4.2b, in which a multiplexer selects the output of the feedback function or of the RTN-based TRNG which is injected in the shift register each time a new bit is generated (i.e., at each rising edge of CLK signal). As a result, the proposed solution overcomes the limitation of the finite period of the NLFSR, thus realizing a high-throughput TRNG [181], [182].

Despite its simplicity, the circuit is intrinsically robust to some common attacks such as noise injection attacks. As long as the noise is random, the output of the TRNG would still be random. Also, other precautions can be adopted while designing the circuit to improve its resistance to other physical attacks. For instance, RF noise attacks that commonly affect most TRNG solutions can be dealt with by shielding the circuit with metal enclosures.

Still, the overall randomness performance of this high throughput TRNG is strictly correlated to the quality of the randomness of the RTN-based TRNG, and consequently on the characteristic of the RTN signal used as an entropy source. This needs to be assessed both in terms of electrical characteristics (e.g., stability, current levels, and $\tau_c + \tau_e$) and the resulting randomness of the produced bitstream. The former requires collecting experimental data, while the latter can be measured with the randomness test suite provided by the National Institute of Standards and Technology (NIST) [187], which consists of 15 statistical tests.

4.2 Performance evaluation

4.2.1 Experimental data

To evaluate the quality of different MIM structures for generating RTN signals for TRNGs applications, we fabricated and characterized different MIM structures [181], [182]. Specifically, different metal oxide-based MIM devices were fabricated depositing the oxide layer either by magnetron sputtering or by Atomic Layer Deposition (ALD) [181]. The effects of the electrodes (i.e., Ni, Ti, and Au) and insulator (i.e., TiO_2, HfO_2, and Al_2O_3) materials, oxide thickness (i.e., 5 nm, 10 nm, and 12 nm), and device area (i.e., $5\mu m$ x $5\mu m$, and $50\mu m$ x $50\mu m$) were also investigated. Additionally, crosspoint MIM devices were fabricated depositing multilayer (i.e., from 3 to 18 layers) hexagonal boron nitride (h-BN) between Ni and Au electrodes using chemical vapor deposition (CVD) [182].

The different tested MIM structures and the results of the statistical and qualitative analysis are reported in Table 4.1. The quality of the generated RTN signal is evaluated in terms of energy efficiency (i.e., current levels), stability, and capture and emission times. To be considered an excellent RTN signal generator, a device need to show high stability, dissipate very low-power (i.e., $\approx 1 - 10\,nW$), have short capture and emission times (i.e., $\approx 10 - 100\,ms$), and have a sufficiently high $I_{MAX} - I_{MIN}$ difference so that the noise can be easily isolated with simple analog circuits.

Different oxides and electrodes configurations can easily lead to different RTN characteristics or no RTN at all. Specifically, we fabricated four different MIM structures, depositing a 5 nm thick oxide layer with ALD while keeping the device lateral size to 5 x 5 μm x μm. $Ni/Al_2O_3/Au$ devices did not show any RTN signal under different biases and the application of

Table 4.1: Analysis of the RTN current signals from different MIM devices. Data from [181], [182]

Sample structure	Oxide thickness (nm)	Device size (μm x μm)	Deposition method	Type of defects	V_{BIAS} (V)	I_{MAX}/I_{MIN} (nA)	Power (nW)	τ_c/τ_e (ms)	Stability	RTN signal quality
$Ni/Al_2O_3/Au$	5	5 x 5	ALD	Native	-	-	-	-	-	No RTN
				Induced	-	-	-	-	-	No RTN
$Ni/HfO_2/Au$	5	5 x 5	ALD	Native	-	-	-	-	-	No RTN
				Induced	4.5	60 000/24 000	189 000	2990/7460	unstable	Bad
$Au/HfO_2/Au$	5	5 x 5	ALD	Native	0.5	0.85/0.66	0.38	850/580	unstable	Bad
$Ti/TiO_2/Au$	5	5 x 5	ALD	Native	-	-	-	-	-	No RTN
				Induced	-	-	-	-	-	No RTN
$Ti/TiO_2/Au$	10	5 x 5	ALD	Native	-	-	-	-	-	No RTN
				Induced	-	-	-	-	-	No RTN
$Ti/TiO_2/Au$	10	50 x 50	ALD	Native	0.05	56/52	2.70	63.3/36.1	Excellent	Good
				Native	0.1	116/108	11.20	62.8/48	Excellent	Very good
				Native	0.15	195/180	28.13	48.3/22.2	Outstanding	Excellent
				Native	0.2	270/252	52.20	40/17.6	up-trend	Good
				Native	0.25	370/340	88.75	24.1/23.1	up-trend	Good
$Ti/TiO_2/Au$	12	5 x 5	Sputtering	Native	-	-	-	-	-	No RTN
				Induced	0.01	2400/2000	22	230/420	Excellent	Very good
				Induced	-0.01	-1900/-2200	20.50	100/260	Excellent	Very good
$Ni/TiO_2/Au$	12	5 x 5	Sputtering	Native	0.02	11.5/6	0.18	29.6/27.2	Very good	Good
$Ni/h-BN/Au$	6	15 x 15	CVD	Native	0.1	200.7/203.8	20.2	17/61	Outstanding	Excellent

a ramp voltage stress only resulted in the dielectric breakdown and thus are not suitable for TRNG applications. Also, $Ni/HfO_2/Au$ devices do not show RTN in the pristine state. Still, after soft-breakdown of the dielectric, some defects are introduced in the insulating layer, resulting in RTN current fluctuations when a 4.5 V bias is applied to the device, as shown in Fig. 4.3. However, despite its stability, the produced RTN signal results in high power consumption and long capture and emission times, thus reducing its quality for TRNG applications. Changing the MIM structure with an Au top electrode (i.e., $Au/HfO_2/Au$ structure), further worsen the RTN characteristic, leading only to a slow and unstable RTN when the device is in the pristine state (see Fig. 4.4), thus suggesting that Ni electrodes paired with HfO_2 oxides are more prone to display RTN. Also, $Ti/TiO_2/Au$ devices exhibit a low quality and erratic RTN signal. Increasing the oxide thickness to 10 nm

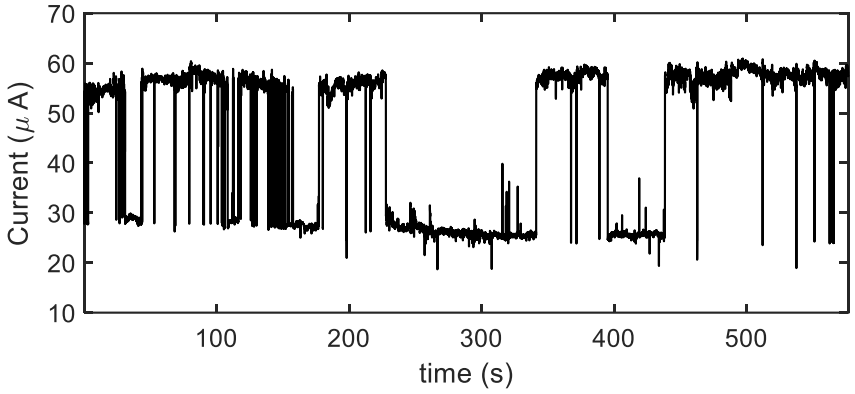


Figure 4.3: RTN current fluctuations measured on a $Ni/5\,nm - ALD\,HfO_2/Au$ $5\mu m \times 5\mu m$ device after a soft breakdown when applying a bias voltage of 4.5 V to its terminals.

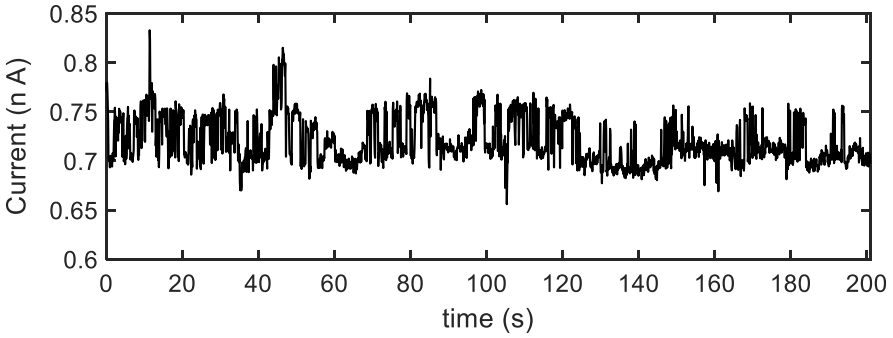


Figure 4.4: RTN current fluctuations measured on a $Au/5\,nm - ALD\,HfO_2/Au$ $5\mu m \times 5\mu m$ device under a 0.5 V bias.

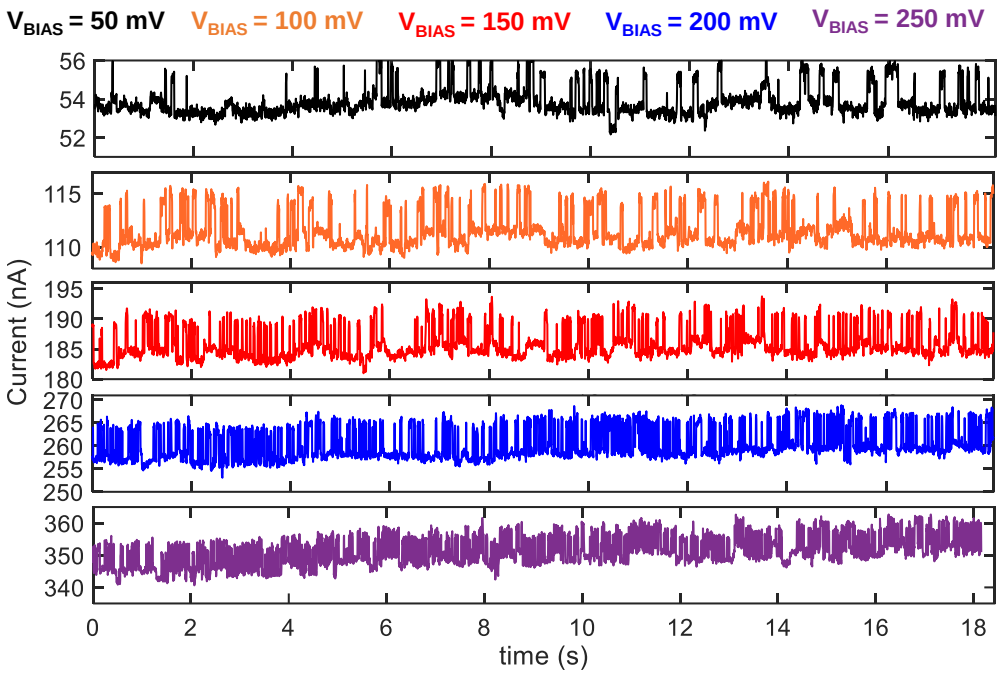


Figure 4.5: RTN current fluctuations measured on a $Ti/10\,nm - ALD\,TiO_2/Au$ $50\mu m$ $\times$ $50\mu m$ device for different bias voltages (i.e., 50 mV, 100 mV, 150 mV, 200 mV, and 250 mV).

does not result in any RTN signal improvement. On the contrary, increasing the device lateral size to $50\mu m$ x $50\mu m$ results in a much better RTN quality. RTN signals at low current levels and short average capture and emission time were consistently reproduced at different constant bias voltages from 50 mV to 250 mV, see Fig. 4.5. This is probably due to the increased probability of finding defects in a device with a larger area. Also, increasing the bias voltage increases the value of the $I_{MAX} - I_{MIN}$, thus simplifying the analog circuit design of the TRNG. However, when the bias voltage is higher than 0.2 V the device's current progressively increases, thus highlighting a progressive degradation of the dielectric. This highlights the existence of a bias voltage versus RTN signal stability trade-off. Devices fabricated by depositing the oxide layer with the magnetron sputtering technique were also

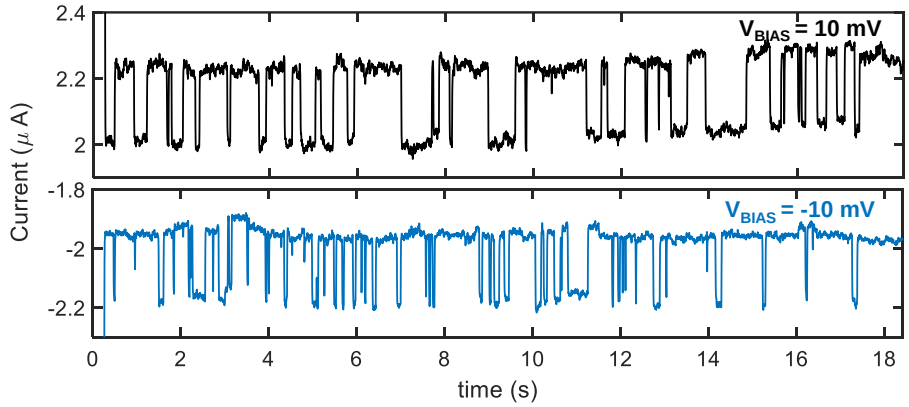


Figure 4.6: RTN current fluctuations measured on a $Ti/12\,nm - Sputtering\,TiO_2/Au$ $5\mu m \times 5\mu m$ device when applying a bias on +10 mV (black line) and -10 mV (blue line).

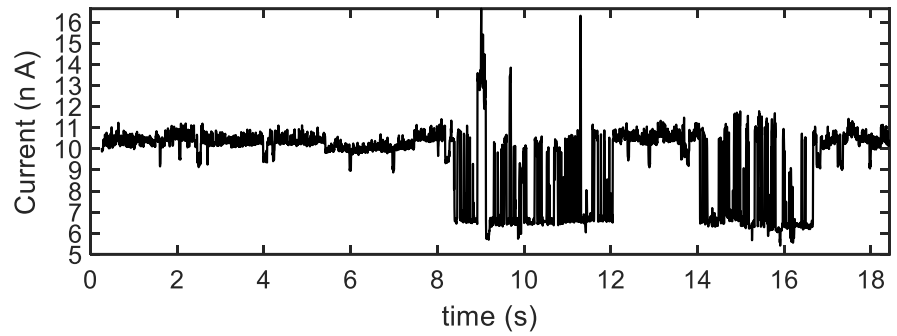


Figure 4.7: RTN current fluctuations measured on a $Ni/12\,nm - Sputtering\,TiO_2/Au$ $5\mu m \times 5\mu m$ device under a 20 mV bias.

studied. Specifically, $Ti/TiO_2/Au$ with 12 nm thick oxide layer and $5\mu m$ x $5\mu m$ lateral size were fabricated. Such devices exhibit a stable RTN signal for low positive and negative biases after a soft dielectric breakdown, as shown in Fig. 4.6. The peculiar RTN signal features a larger $I_{MAX} - I_{MIN}$, while still providing high energy efficiency and sufficiently fast capture and emission times. Magnetron sputtered $Ni/TiO_2/Au$ devices with $5\mu m$ x $5\mu m$ lateral size was shown to easily display RTN in their pristine state, with good characteristics. However, such RTN signal was shown to change characteristics over time (see Fig. 4.7) despite the same constant voltage bias probably due

to the presence of multiple defects sustaining the RTN current conduction.

Also, MIM devices in which approximately 3 and 18 h-BN layers were deposited with the CVD technique were fabricated and their RTN signal characteristics were analyzed [182]. In Table 4.1, are reported the results for $Ni/h{-}BN/Au$ devices with a lateral size of $15\mu m \times 15\mu m$. The experimental data suggest that such devices exhibit an exceptionally stable RTN signal, thanks to the characteristics of the defects in the h-BN layer (i.e., regions where the B and N atoms are disordered and nearly amorphous) [182]. By an analysis performed with conductive atomic force microscopy (CAFM), such defects are confined in a very small radius (i.e.,$\approx 10nm$), even after stress.

4.2.2 Randomness tests results

The experimental RTN current signals from $Ti/10nm - ALD\ TiO_2/Au$, $Ti/12nm{-}Sputtering TiO_2/Au$, $Ni/12nm{-}Sputtering TiO_2/Au$, and $Ni/6nm{-}CVD\ h-BN/Au$ devices (i.e., which are the four fabricated devices showing the best RTN quality) were used as inputs of TRNG circuit in Fig. 4.2 that was simulated with the Cadence Virtuoso® software. To evaluate the quality of the generated RTN signals as entropy sources and the achieved randomness of the complete high-throughput TRNG, we executed the set of randomness tests from the NIST test suite both on the output of the low-throughput (i.e., the output of the S/H circuit), and high-throughput (i.e., the output of the modified NLFSR) TRNGs, respectively. Some of the NIST tests require that the tested sequences are at least 10^5 bits long, thus only 11 out of 15 tests could be run on the output of the low-throughput TRNG due to the limited length of the experimental RTN signals. Nevertheless, all the tests were executed on the output bitstream of the high-throughput of the NLFSR.

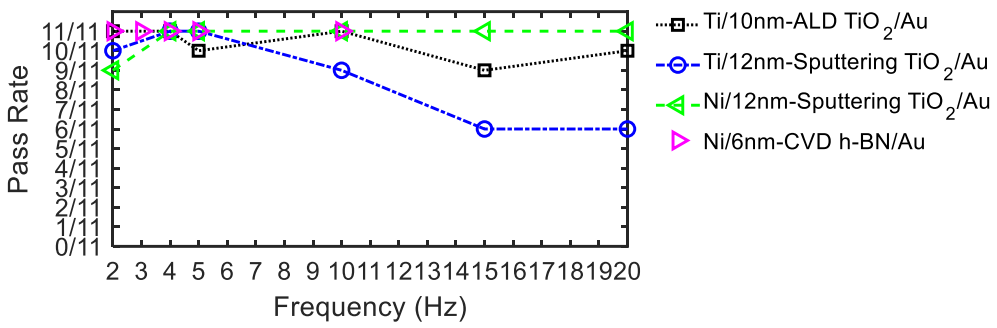


Figure 4.8: NIST tests pass rate computed on the output of the S/H circuit (RTN-based TRNG) for different sample clock frequencies and the four MIM structures displaying the highest quality TRNG. Data from [181], [182].

The throughput of the RTN-based TRNG is controlled by the frequency of the CLK signal and is directly linked to the capture and emission times of the defected present in the MIM structure of the devices. As shown by the result of the NIST tests for different clock frequencies that are reported in Fig. 4.8, the output of the TRNG starts failing some tests when input the RTN signal from the $Ti/12nm - Sputtering\ TiO_2/Au$ device and the clock frequency is $\geqslant 10\ Hz$, while passes all the tests up to a clock frequency of $20Hz$ when input the RTN signals from the other devices. This is due to the longer capture and emission times of the defects in the $Ti/12nm - Sputtering\ TiO_2/Au$ (see Table 4.1). In fact, as previously mentioned the sampling period of the S/H block must be around or lower than the sum of the average capture and emission times. Thus, for the $Ti/10nm - ALD\ TiO_2/Au$, $Ti/12nm - Sputtering\ TiO_2/Au$, $Ni/12nm - Sputtering\ TiO_2/Au$, and $Ni/6nm - CVD\ h-BN/Au$ devices, clock frequencies of 10 Hz, 5 Hz, 20 Hz, and 10 Hz are used, respectively.

a)

Test	p-Value	Result
Frequency Test	0.30	PASSED
Frequency Test within a Block	0.11	PASSED
Run Test	0.34	PASSED
Longest Run of Ones in a Block	0.84	PASSED
Binary Matrix Rank Test	0.08	PASSED
DFT Test	0.96	PASSED
Non-Overlapping Template Matching Test	0.19	PASSED
Overlapping Template Matching Test	0.06	PASSED
Maurer's Universal Statistical test	0.75	PASSED
Linear Complexity Test	>0.94	PASSED
Serial test	1.00	PASSED
Approximate Entropy Test	0.98	PASSED
Cumulative Sums (forward/backwards)	>0.27	PASSED
Random Excursions Test	>0.15	PASSED
Random Excursions Variant Test	>0.06	PASSED

b)

Test	p-Value	Result
Frequency Test	0.80	PASSED
Frequency Test within a Block	0.87	PASSED
Run Test	0.72	PASSED
Longest Run of Ones in a Block	0.63	PASSED
Binary Matrix Rank Test	0.37	PASSED
DFT Test	0.41	PASSED
Non-Overlapping Template Matching Test	0.66	PASSED
Overlapping Template Matching Test	0.88	PASSED
Maurer's Universal Statistical test	0.91	PASSED
Linear Complexity Test	>0.17	PASSED
Serial test	1.00	PASSED
Approximate Entropy Test	0.98	PASSED
Cumulative Sums (forward/backwards)	>0. 56	PASSED
Random Excursions Test	>0.28	PASSED
Random Excursions Variant Test	>0.32	PASSED

c)

Test	p-Value	Result
Frequency Test	0.15	PASSED
Frequency Test within a Block	0.96	PASSED
Run Test	0.54	PASSED
Longest Run of Ones in a Block	0.76	PASSED
Binary Matrix Rank Test	0.66	PASSED
DFT Test	0.52	PASSED
Non-Overlapping Template Matching Test	0.80	PASSED
Overlapping Template Matching Test	0.83	PASSED
Maurer's Universal Statistical test	0.59	PASSED
Linear Complexity Test	>0.70	PASSED
Serial test	1.00	PASSED
Approximate Entropy Test	0.98	PASSED
Cumulative Sums (forward/backwards)	>0.27	PASSED
Random Excursions Test	>0.20	PASSED
Random Excursions Variant Test	>0.20	PASSED

d)

Test	p-Value	Result
Frequency Test	0.174	PASSED
Frequency Test within a Block	0.965	PASSED
Run Test	0.285	PASSED
Longest Run of Ones in a Block	0.269	PASSED
Binary Matrix Rank Test	0.722	PASSED
DFT Test	0.557	PASSED
Non-Overlapping Template Matching Test	0.880	PASSED
Overlapping Template Matching Test	0.159	PASSED
Maurer's Universal Statistical test	0.165	PASSED
Linear Complexity Test	0.365	PASSED
Serial test	1.000	PASSED
Approximate Entropy Test	0.9997	PASSED
Cumulative Sums (forward/backwards)	> 0.208	PASSED
Random Excursions Test	> 0.025	PASSED
Random Excursions Variant Test	> 0.027	PASSED

Figure 4.9: Results of the NIST tests computed on the output of the modified NLFSR when the TRNG is input the RTN signal from a) a $Ti/10nm - ALD\ TiO_2/Au$ $50\mu m$ x $50\mu m$ device, b) a $Ti/12nm - Sputtering\ TiO_2/Au$ $5\mu m$ x $5\mu m$, c) $Ni/12nm - Sputtering\ TiO_2/Au$ $5\mu m$ x $5\mu m$ device, and d) a $Ni/6nm - CVD\ h - BN/Au$ $15\mu m$ x $15\mu m$ device, respectively. Data from [181], [182]. All tests are passed (i.e., p-value $>$ 0.01).

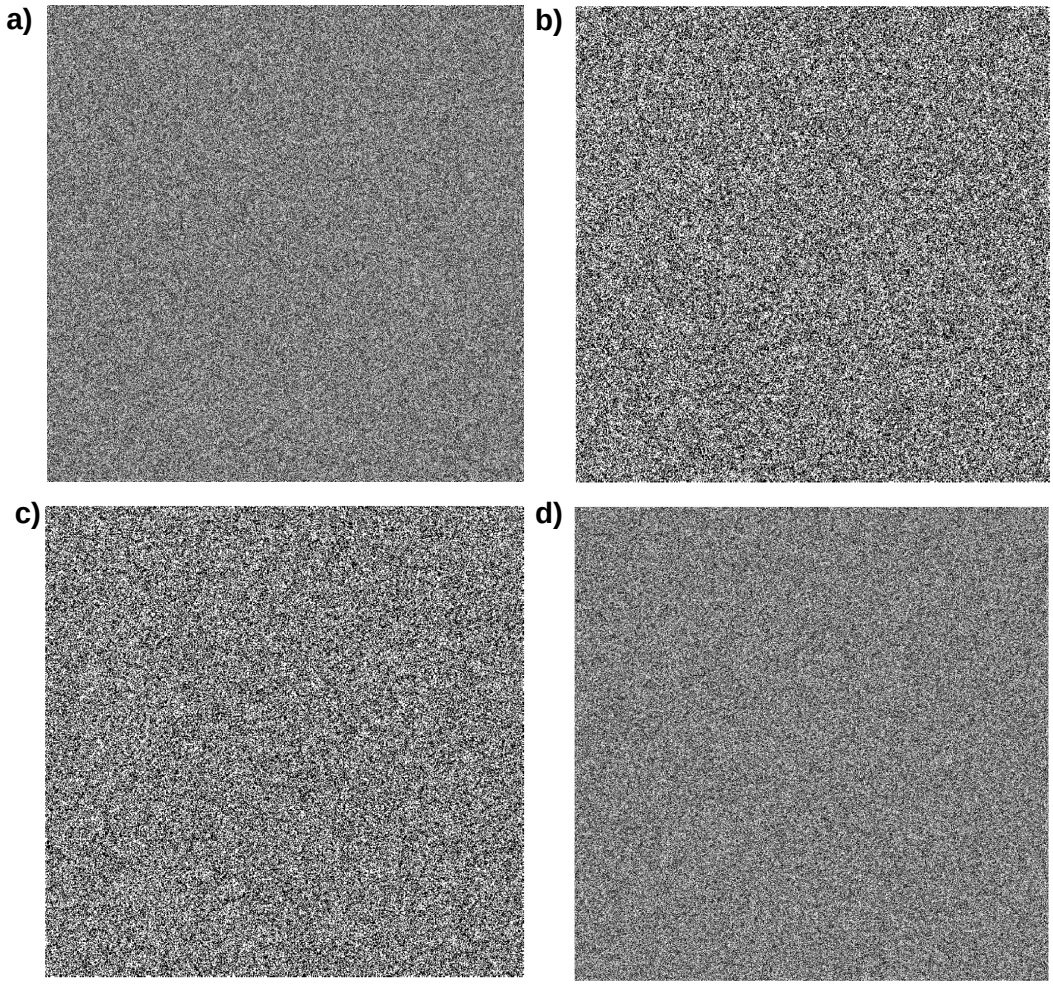


Figure 4.10: Bitmaps built from the output of the high-throughput TRNG when input the RTN signal from a) a $Ti/10nm-ALDTiO_2/Au$ 50μm x 50μm device, b) a $Ti/12nm-SputteringTiO_2/Au$ 5μm x 5μm, c) $Ni/12nm-SputteringTiO_2/Au$ 5μm x 5μm device, and d) a $Ni/6nm-CVDh-BN/Au$ 15μm x 15μm device, respectively. Data from [181], [182]. In all cases, no pattern can be distinguished despite more bits than the NLFSR period are displayed, confirming the randomness of the generated sequences.

For all four devices, the clock frequency of the modified NLFSR is 1 MHz thus resulting in a 10^6 bit/s output bitstream. A lower sampling frequency leads to longer time intervals between the update with a random bit of the internal state of the NLFSR. Increasing the number of Flip-Flops composing

the NLFSR would help in preventing the occurrence of too few seed updates, as shown in Fig. 4.9 the generated sequences pass all the 15 NIST tests. Also, no pattern is visible by eye inspection of the generated bitmaps (see Fig. 4.10), confirming the randomness of the generated sequences. The length of the generated sequences is longer than the period of the NLFSR, thus demonstrating the effectiveness of the periodic seed update strategy. These results confirm that RTN of MIM devices is a promising entropy source and enables the realization of low-cost, and energy-efficient TRNG that are ideal for low-power devices for the IoT.

4.3 Device-circuit co-optimization

When designing the high throughput TRNG circuit it is important to consider the specific characteristics of the RTN signal. For instance, as mentioned in the previous section the capture and emission times of the defects, and the desired output throughput, determine the value of the clock signals for the S/H and the NLFSR, and the number of bits of the latter. Also, the average value of the device current and the amplitude of the RTN signal influence the gain of the transimpedence amplifier and the number of amplification stages that are required.

Still, other possible side effects that could influence the circuit reliability should be considered. For instance, even if some of the measured devices showed stable and reproducible RTN signals, over the entire life span there is a high probability that in some periods RTN signal from a single device stops due to local field or temperature variations causing the defect to remain stuck in a specific state (i.e., filled or empty), as shown in Fig. 4.11b. Even though the circuit could potentially still function by exploiting the thermal

noise as entropy source, it would be less secure to attacks. Thus, some solutions should be introduced at the circuit level. For instance, increasing the redundancy of the entropy source by reading the state of multiple MIM devices in parallel and designing an arbitrator circuit capable of choosing the best entropy at a specific moment in time would potentially solve the issue, however at an increase of the circuit complexity and power consumption. Alternatively, another arbitrator circuit could be used to slightly change the bias voltage with the aim of restoring the defect behavior.

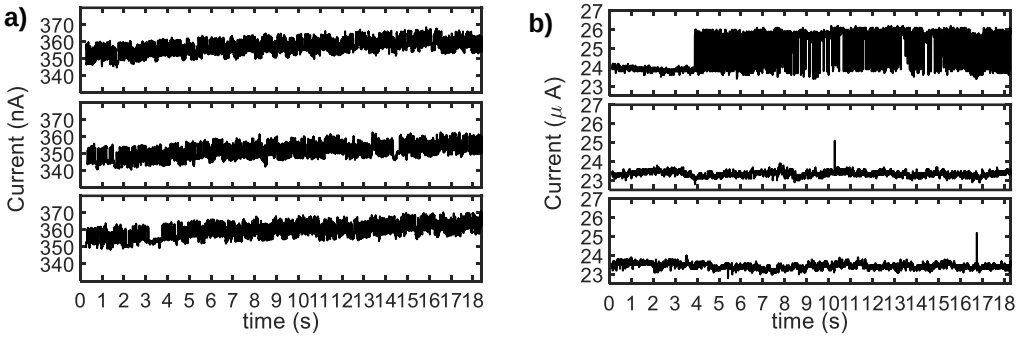


Figure 4.11: a) Drift (up-trend) of the average current observed on a $Ti/10nm - ALD\ TiO_2/Au$ 50μm x 50μm devices under a 250 mV bias. Data from [181] b) Different RTN signals measured at different times on a $Ti/12nm - Sputtering TiO_2/Au$ 5μm x 5μm under a 120 mV bias. In the upper I-t plot a clear RTN signal is visible after 4s, while in the other traces only a small noise is observed. Data from [181]

Also, potential slow drift of the device average current (see Fig. 4.11a) could affect the circuit operation. A well-designed high-pass filter would solve the problem of slowly varying drifts, it may require the use of very low cut-off frequencies and in turn large R_f and C_f values that are difficult to implement in integrated circuits. The use of external capacitors and resistors could be a solution, however potentially reducing the security of the circuit to physical attack or requiring the fabrication of custom enclosures.

While the $Ni/h - BN/Au$ thanks to their excellent stability may not

require specific compensations of the drift effect, they may require particular attention when designing the amplification stage. Low-noise and low-power amplifier design should be employed due to the small (i.e., few nA) available $I_{MAX} - I_{MIN}$ value showed by such devices. Thus, improvements both at a device and circuit level are still required before commercialization.

4.4 Related published works

The results presented in this Chapter were published during the PhD program in the following journals publications [181], [182].

Conclusions

Discussion

The results and solutions presented in this thesis represent an important advancement for the development of ultra-low-power and secure devices for the future of IoT and edge computing. Specifically, the UniMORE physics-based compact model, together with the developed automated parameter extraction procedure described in Chapter 2, represents a fundamental tool for enabling accurate and trustable circuit simulations. The possibility of calibrating the parameters of the model with few experimental data enables circuit designers to take advantage of the multitude of RRAM technologies that are being developed. For instance, a designer could select the best technology for a specific application, consider the impact of peripheral circuits, and thus implement appropriate device-circuit co-optimization strategies. This is confirmed by the performance and reliability analysis of the different in-memory computing architecture presented in Chapter 3.

Thanks to the UniMORE compact model, the reliability and performance limitations of conventional IMPLY architectures were estimated, providing clear design boundaries and trade-offs. Also, the effectiveness of the proposed SIMPLY architecture, and the multi-input IMPLY operation, was thoroughly evaluated with the model, highlighting its potential as a solution for imple-

menting ultra-low-power LiM hardware accelerators based on state-of-the-art RRAM devices for IoT applications. Compared to other solutions, SIMPLY provides enhanced flexibility, enabling reconfiguring the logic function computed in the accelerator. Such flexibility was further increased by proposing a novel architecture that enables the execution of both SIMPLY and analog VMM operations, enabling the energy-efficient implementation of BNN inference tasks, exploiting the advantages offered by the two approaches.

Also, the performance and reliability of LBPNN were studied, highlighting specific performance and reliability trade-offs: i) the power-throughput trade-off during inference is determined by the neuron circuits, ii) the reset voltage controls both accuracy and power efficiency, iii) accuracy losses are dominated by the LRS variability. Such trade-offs were used to devise appropriate design strategies, enabling the implementation of accurate and energy-efficient RRAM-based LBPNN on low-power devices.

Also, RTN-based TRNG were studied proposing device-circuit co- optimization strategies and determining the impact of different materials and fabrication techniques on the quality of the generated RTN that is used as entropy source of the TRNG circuit. Experimental results indicate for high-quality RTN generation, i) larger devices ($50\,\mu m$ x $50\,\mu m$) are better than smaller ones ($5\,\mu m$ x $5\,\mu m$), ii) electrodes built with Ni or Ti result in higher quality RTN signals than the ones built with Au, iii) TiO_2 oxide provides better RTN than Al_2O_3 and HfO_2, iv) deposition of the oxide layer through sputtering results in better quality RTN compared to ALD deposited layers and v) devices with 10 nm thick oxide layer provide better RTN than the ones with 5 nm thick oxide layer. Also, MIM structures with CVD deposited layers of 2D h-BN material result in exceptionally stable RTN signals, suggesting that such technology is a promising candidate for the development of

future hardware security primitives.

Overall, the results reported in this thesis underline the potential of RRAM technology as a candidate solution for future ultra-low-power devices for IoT and edge computing applications. Thanks to the UniMORE RRAM physics-based compact model, device-circuit co-optimization strategies aimed at high energy efficiency and circuit reliability have been demonstrated. Still, circuit solutions would benefit from technology improvements. Thus, efforts should also be directed to the circuits and devices' optimization.

Specifically, the peripheral circuit employed to compute the analog VMM multiplication and the SIMPLY operation can impact the performance and reliability achievable with different in-memory computing architectures. Indeed the sensing circuits play a major role in the successful execution of read operations. Although a simple sense amplifier design was used to demonstrate the feasibility of the SIMPLY operation, the fabrication of robust and reliable sense amplifier circuits is challenging. Other more robust sensing circuits could be employed, however likely introducing an area, energy efficiency, and speed trade-off, therefore demanding, as for LBPNNs, the development of custom solutions that will be the focus of future works. Thus, designing novel compact, robust, and energy-efficient sensing circuits remains a pressing and open challenge.

From a memory device perspective, although the results presented in this thesis are based on RRAM devices, the proposed circuit solutions could be implemented also with other eNVM technologies. Depending on their specific features and the considered circuit, each different eNVM technology could be more suited for specific applications. In fact, hardware accelerators for the SIMPLY and the analog VMM operations have different requirements that potentially could be better addressed by using different eNVM technologies.

Specifically, in the hardware accelerators for the analog VMM, endurance is not necessarily a limiting factor. Conversely, for SIMPLY-based architectures, endurance is an important discriminant for determining the set of applications that could be targeted.

For instance, an endurance $> 10^{14}$ should be available to target applications requiring intensive computations. Therefore, these applications could be targeted with STT-MTJ or SOT-MTJ technologies, which can provide $> 10^{15}$ endurance and high switching speed ($\approx$ ns), with the main challenge being the smaller available memory window. Nevertheless, recently we demonstrated the feasibility of the 2-SIMPLY operation with the STT-MTJ technology by performing circuit simulations enabled by a physics-based compact model including the effect of variability [188]. The reduced memory window could also affect the reliability of the analog VMM operation. Still, Gao et al. [189] demonstrated the feasibility of BNNs with analog VMM using STT-MTJ devices, however at the cost of introducing specific hardware calibration steps and introducing virtual ground nodes with operational amplifiers, thus leading to lower throughput, efficiency, and larger chip area.

Other applications, such as smart sensors, may require less frequent computations. For instance, a device may be in an idle state most of the time, while performing a sporadic burst of operations. These applications could be targeted with eNVM technologies characterized by lower endurance. Considering the case of the SIMPLY BNN implementation presented in Section 3.2.4 and a device endurance of 10^8, to ensure a reliable device operation over 10-years, a maximum of 20 inferences per minute should be performed. Many eNVM technologies, can achieve endurance $> 10^8$ and thus are suitable for these applications. Among these technologies, RRAMs represent a good compromise between low programming energy, fast switching speed, large

memory windows, and high scalability.

Also, other less mature technologies such as FTJ devices could provide interesting opportunities for the development of ultra-low-power in-memory computing architectures thanks to their very low programming energy and fast switching speed, but still, their endurance and retention characteristics must be improved [39], [42].

Future Outlook

In this thesis, we focused specifically on solutions that could be implemented with state-of-the-art devices and could be introduced in the market in the short- and medium-terms. By expanding the time horizon, one of the challenges of the future will be to bring the energy-efficient execution of learning paradigms at the edge of the network, on low-power devices. Thus, future research activity must be directed towards ultra-low-power learning at the edge of the network, making systems adaptable to changes to their operating environment and capable of automatically detecting anomalies. In this scenario, RRAM and eNVM technologies, in general, are considered as the key enabling technology. Technology improvement regarding the reliability of the multi-bit or analog programming and the endurance of eNVM devices would indeed enable the implementation of the online training (i.e., training the network on the same hardware used for inference) of DNNs. Still, other solutions could be adopted.

For instance, training a DNN on the same hardware that is later used to perform a classification task enables to reduce the accuracy loss that occurs when mapping the parameters of a network trained in software (i.e., offline training) [3], compensating for the hardware nonidealities. Thus, hy-

brid solutions could be employed, where the network is first trained offline on conventional hardware and a final parameter optimization step is performed in hardware (online) to improve the accuracy, while limiting the required device endurance. Also, a lot of research activity is directed towards the use of eNVM technologies as building blocks of biologically plausible learning rules implemented in hardware. In fact, neurons and synapses can perform classification, clustering, and motion control tasks despite their intrinsic stochastic characteristic, and thus could potentially best exploit the advantages provided by eNVM technologies.

Tackling these future challenges will require knowledge from different scientific disciplines, from material science to neuroscience, therefore highlighting the need for combined multidisciplinary efforts.

Appendix a

UniMORE RRAM physics-based compact model v2.0

I Compact Model

```
1  `include "constants.vams"
2  `include "disciplines.vams"
3  `include "UNIMORE_RRAM_model_additional_disciplines_1_0_0
       .va" // Additional natures definitions: the barrier
       thickness and the temperature require non-standard
       tolerances during the SET of the device.
4
5  module UNIMORE_RRAM_model_1_0_3 (TE, BE, oRTN_EN);
6
7    inout TE, BE;
8    input oRTN_EN;
9    //External nodes
10   electrical oRTN_EN,TE, BE;
11   //Internal nodes
```

```
12  electrical ME, ME2,N_noise_x, N_noise_s, N_noise_beta,
       N_noise_x2;
13  my_temperature n2, n3, n22, n33, gnd_t;
14  my_barrier n1, n11, gnd_b;
15
16  /********
17  * TE         - Top electrode
18  * BE         - Bottom electrode
19  * n1         - Barrier thickness internal node 1[V] = 1[
       nm]
20  * n11        - Support node used for barrier thickness
       initialization 1[V] = 1[nm]
21  * n2         - Conductive filament temperature internal
       node 1[V] = 1[nm]
22  * n22        - Support node used for conductive filament
        temperature initialization 1[V] = 1[K]
23  * n3         - Barrier temperature internal node 1[V] =
       1[nm]
24  * n33        - Support node used for barrier temperature
        initialization 1[V] = 1[K]
25  * gnd_t      - Internal ground reference for
       my_temperature discipline
26  * gnd_b      - Internal ground reference for my_barrier
       discipline
27  * N_noise_s - White_noise internal node for CF cross-
       section variability
28  * N_noise_x - White_noise internal node for barrier
       thickness variability
29  * N_dvdt     - Internal node for device voltage time
```

```
    derivative computation
30  ********/
31
32  (*desc = "Oxide material resistivity", units = "ohm*nm"
    *)              parameter real rho=3000 from (0:inf);
33  (*desc = "Oxide layer thickness", units = "nm"*)
                            parameter real t_ox=5 from (0:
    inf);
34  (*desc = "Initial conductive filament section", units =
     "nm^2"*)       parameter real S0=12.75 from (0:inf);
35  localparam real kb = 8.6e-5 from (0:inf); //Boltzmann
    constant eV/K
36  (*desc = "Activation Energy TAT", units = "eV"*)
                            parameter real Ea=0.0513 from [
    0:inf);
37  (*desc = "Ambient temperature", units = "K"*)
                               parameter real T0=303.15
    from [0:inf);
38  (*desc = "Typical tunneling length", units = "nm"*)
                          parameter real l=0.42 from (0:inf)
    ;
39  (*desc = "V0 HRS current non-linearity factor", units =
     "V"*)          parameter real V0_HRS=0.3326 from (0:
    inf);
40  (*desc = "V0 LRS current non-linearity factor", units =
     "V"*)          parameter real V0_LRS=2 from (0:inf);
41  (*desc = "Resistivity temperature coeff.", units = "1/K
    "*)             parameter real alpha=2.58e-4 from [0:
    inf);
```

```
42    (*desc = "Barrier␣resistance␣fitting␣parameter"*)
                              parameter real beta=0.199 from [
      0:inf);
43    (*desc = "Bond␣vibration␣frequency", units="Hz"*)
                              parameter real c0=1e13 from [0:
      inf);
44    (*desc = "Barrier␣thermal␣capacity", units = "J/K"*)
                            parameter real Cpb=1.1e-13 from [0:
      inf);
45    (*desc = "CF␣thermal␣capacity", units = "J/K"*)
                               parameter real Cpcf=5e-11 from
       [0:inf);
46    (*desc = "Barrier␣thermal␣conductivity", units = "W/K"
      *)                  parameter real kbar=1.0622e-06 from
      [0:inf);
47    (*desc = "CF␣thermal␣conductivity", units = "W/K"*)
                             parameter real kcf=2.136e-06 from
      [0:inf);
48    (*desc = "Barrier/CF␣mutual␣thermal␣conductivity",
      units = "W/K"*)     parameter real kex=1e-6 from [0:
      inf);
49    (*desc = "Diffusion␣activation␣energy␣of␣oxygen␣ions␣",
       units = "eV"*)parameter real Ead=1.8 from [0:inf);
50    (*desc = "Field␣enhancement␣factor␣for␣oxygen␣ions␣
      diffusion", units = "e␣*␣nm"*) parameter real g=5.1
      from [0:inf);
51    (*desc = "Field␣enhancement␣factor␣for␣bond␣breaking",
      units = "e␣*␣nm"*)     parameter real gg=1.7 from [0:
      inf);
```

```
52    (*desc = "Bond␣breaking␣activation␣energy", units = "eV
         "*)              parameter real Eag=1.5 from [0:inf);
53    (*desc = "Initial␣barrier␣thickness", units = "nm"*)
                         parameter real xinit=0 from [0:inf)
         ;
54    (*desc = "Initial␣temperature", units = "K"*)
                                 parameter real Tinit=303.15
         from [0:inf);
55    (*desc = "Temperature␣at␣which␣RLRS␣is␣measured␣",
         units = "K"*)        parameter real Tmeas=303.15 from
         [0:inf);
56    (*desc = "Minimum␣time␣step␣(for␣positive␣applied␣
         voltages)", units = "s"*) parameter real
         min_time_step_vpos = 100e-15 from [0:inf);
57    (*desc = "Minimum␣time␣step␣(for␣negative␣applied␣
         voltages)", units = "s"*) parameter real
         min_time_step_vneg = 100e-15 from [0:inf);
58    (*desc = "Adaptive␣time␣step␣parameter"*)
                                    parameter real
         tstep_param = 1e2;
59    (*desc = "RESET␣curve␣slope␣fitting␣parameter", units="
         e*nm"*)          parameter real a = 0.7 from [0:inf);
60    (*desc = "RESET␣curve␣curvature␣fitting␣parameter"*)
                         parameter real b = 4 from [0:inf);
61
62    // Variability Parameters
63    (*desc = "SET␣event␣detection␣threshold␣on␣the␣barrier␣
         derivative␣", units = "nm/s"*) parameter real th_set
          = 10;
```

```
64    (*desc = "Maximum variation allowed on the CF cross-
         section (=3sigma)", units = "nm^2"*)parameter real
         ds = 0.4748 from[0:inf);
65    (*desc = "CF cross-section at which the measured ds is
         obtained", units = "nm^2"*)       parameter real
         S0var = 12.75 from[0:inf);
66    (*desc = "Maximum variation allowed on the barrier
         thickness (=3sigma)", units = "nm"*) parameter real
         sigmaRoverR = 0.3374 from[0:inf);
67    (*desc = "Maximum frequency of the noise for transient
         noise analysis", units = "nm"*)  parameter real Fmax
          = 1000 from(0:inf);
68    parameter real Tmin = 5e-4;       // Transient noise
         analysis noise update time step
69    localparam real Wnoise_param_HRS = 1 /(9*Fmax) from[0:
         inf);                  //White_noise power
70    localparam real Wnoise_param_s = ds*ds * S0var*S0var
         /(9*Fmax) from[0:inf);  //CF cross-section
         white_noise power
71    localparam real max_s_noise_amplitude = ds * S0var from
         [0:inf);              //CF cross-section noise
         maximum amplitude
72    parameter real maxdxdt = 3e8;     // Barrier derivative
         saturation during SET
73    parameter real ddt_x_crit = 1e-3; // Threshold used to
         activate the injection of variability during RESET
74    parameter real smooth = 1e-12;    // Smoothing parameter
          used to switch the flag enabling the variability
         during RESET
```

```
75    parameter real smoothing_param = 1e-9; // Smoothing
         parameter used in the clipping of the barrier
         between 0nm  and tox
76    parameter real msigmaRoRvsdTbardt = 3.99e-10;
77    parameter real msigmaRoRvsdTcfdt = 6.81e-10;
78
79    // RTN Parameters
80    (*desc = "Initial␣random␣seed␣to␣account␣for␣
         variability"*)              parameter real
         rand_seed_ini = 0;
81    (*desc = "Parameter␣used␣to␣switch␣on␣the␣RTN␣module␣
         (0=OFF;␣1=ON)"*)    parameter integer RTN_ON = 0 from
          [0:1];
82    (*desc = "Capture␣and␣emission␣times␣constant", units =
          "J*m^3/s"*)      parameter real const0 = 4.19e-32;
83    (*desc = "Density␣of␣states␣at␣the␣bottom␣of␣the␣
         conduction␣band␣", units = "1/(J*m^3)"*)
         parameter real Nc = 2.42e45;
84    (*desc = "Energy␣barrier␣for␣injected␣electrons", units
          = "eV"*)         parameter real phi = 2.1;
85    (*desc = "Typical␣tunneling␣length␣(capture)", units =
         "eV"*)            parameter real lambda_c = 2e-10;
86    (*desc = "Typical␣tunneling␣length␣(emission)", units =
          "eV"*)           parameter real lambda_e = 2e-10;
87    (*desc = "Maximum␣number␣of␣defects␣that␣can␣be␣
         generated"*)             parameter integer
         maximum_number_defects = 100;
88    localparam real t_ox_rtn = 1e-9 * t_ox; // oxide
         thickness in (m)
```

```
89    (*desc = "Spread␣of␣the␣oxygen␣ions␣relaxation␣energy␣
         distribution", units = "eV"*)           parameter
         real Delta_Erel_O = 0.4;
90    (*desc = "Spread␣of␣the␣oxygen␣ions␣thermal␣ionization␣
         energy␣distribution␣", units = "eV"*) parameter real
          Delta_Et_O = 0.5;
91    (*desc = "Spread␣of␣the␣oxygen␣vacancies␣relaxation␣
         energy␣distribution", units = "eV"*)      parameter
         real Delta_Erel_V = 0.4;
92    (*desc = "Spread␣of␣the␣oxygen␣vacancies␣thermal␣
         ionization␣energy␣distribution␣", units = "eV"*)
                     parameter real Delta_Et_V = 0.5;
93    (*desc = "Mean␣of␣the␣normal␣distribution␣associated␣to
         ␣the␣logNormal␣distribution␣of␣the␣RHRS␣due␣to␣RTN"
         *) parameter real DeltaR_dist_HRS_mean =
         -0.693147180559945; // ln(0.5)
94    (*desc = "Standard␣deviation␣of␣the␣normal␣distribution
         ␣associated␣to␣the␣logNormal␣distribution␣of␣the␣
         RHRS␣due␣to␣RTN"*) parameter real
         DeltaR_dist_HRS_std = 0.6;
95    (*desc = "Standard␣deviation␣of␣the␣normal␣distribution
         ␣of␣the␣RLRS␣due␣to␣RTN.␣For␣further␣information␣see
         "*)              parameter real DeltaR_dist_LRS_std =
          0.3;
96    (*desc = "Length␣of␣the␣side␣of␣the␣cubic␣volume␣of␣
         influence␣of␣a␣defect", units = "m"*) parameter real
          rt = 1.7e-9;
97    (*desc = "Defects␣density␣in␣HRS", units = "m^-3"*)
                                  parameter real
```

```
    O_ions_density = 1e26;
98  (*desc = "Defects density in LRS", units = "m^-3"*)
                                parameter real V_density =
     2e27;
99  (*desc = "Nominal oxygen ions relaxation energy", units
     = "eV"*)                   parameter real Erel0_O =
    2.67;
100 (*desc = "Nominal oxygen ions thermal Ionization energy
    ", units = "eV"*)           parameter real Et0_O = 2.3;
101 (*desc = "Nominal oxygen vacancies relaxation energy ",
     units = "eV"*)             parameter real Erel0_V =
    1.19;
102 (*desc = "Nominal oxygen vacancies thermal ionization
    energy", units = "eV"*) parameter real Et0_V = 2.1;
103 (*desc = "Threshold on the barrier derivative to
    randomly re-assign defects positions", units = "m/s"
    *) parameter real dxdt_th = 1e-10 from [0:inf);
104 localparam real ln_beta = ln(beta); // oxide thickness
    in (m)
105
106 parameter real tcf_noise_fact = 1;
107
108
109 // Model variables
110 real Vtb, Vtbmin;               // Voltage across the
    Top and Bottom electrodes (V)
111 real R_lrs;             // Low Resistive State
    resistance variable (ohm)
112 real Sp;                // Negative voltage flag (1
```

```
        when Vtb<0, 0 elsewhere)
113     real Sn;                // Positive voltage flag (1
        when Vtb>=0, 0 elsewhere)
114     real barrier;           // Barrier thickness variable (
        nm)
115     real set_v;             // SET event detection variable
116     real Tcf;               // CF temperature variable (K)
117     real Tbar;              // Barrier temperature variable
         (K)
118     real ddt_Tbar;          // Barrier temperature time
        derivative (K/s)
119     real ddt_Tcf;           // CF temperature time
        derivative (K/s)
120     real ddt_x;             // Barrier thickness time
        derivative (nm/s)
121     real R;                 // Device total resistance
        variable (ohm)
122     real Rcf;               // CF resistance variable (ohm)
123     real Rbar;              // Barrier resistance variable
        (ohm)
124     real Vbar;              // Voltage across the barrier (
        V)
125     real Vcf;               // Voltage across the
        conductive filament (V)
126     real x_b;               // Support variable for barrier
         thickness time derivative (e*nm)
127     real S_var;             // CF cross-section variable (
        nm^2)
128     integer flag_set;       // Activation of variability
```

```
    during a SET event (1 during a SET event, 0
    elsewhere)
129 real tstep;              // Time step variable (s)
130 real sig_wn_dx, sig_wn_s;// RESET and SET noise
    variables for variability
131 real II;                 // Output current from RRAM
    model without RTN (A)
132 real std_x, std_x_Tcf;              // RESET noise power
     proportional to Rbar
133 real n_ddt_x_Rbar;       // RESET noise term on dx/dt
134 real tA, tB, tC;         // Barrier clipping terms added
     to dx/dt
135 real sig_wn_beta;
136 real I1,V1, V3, I4, V4;
137 real beta_var;
138 real sig_RoR;
139 real ddt_Tbar_avg, noise_on_x;
140 real n_ddt_x_Rbar2, noise_on_x2;
141 real s_beta;
142
143
144 // RTN model variables
145 real I_rtn;                // RTN Current contribution (
    A)
146 real x;                    // Barrier thickness value (m
    )
147 integer i;                 // for loop cycles variable
148 real N_O;                  // Average number of O ions
    in vol. x*S
```

```
149     real N_V;                   // Average number of O
            vacancies around the CF
150     real Erel_O;                // Oxygen ions relaxation
            energy (eV)
151     real Et_O;                  // Oxygen ions thermal
            ionization energy (eV)
152     real Erel_V;                // Oxygen vacancies
            relaxation energy (eV)
153     real Et_V;                  // Oxygen vacancies thermal
            ionization energy (eV)
154     integer n;                  // Number of defects
155     real Delta_R;               // Total Resistance Variation
             due to RTN (ohm)
156     real tau;                   // Defect transition time -
            either tau_e or tau_c, depending on State
157     real Srtn;                  // CF cross-section in m^2
158     real delta_R_ref;           // Delta R offset due to RTN
            updated after defect position re-initialization.
159     real delta_R_ref2;         // Delta R offset due to RTN
            updated after defect position re-initialization.
160     real f_correction;          // RTN defects DeltaR
            correction factor
161     real ddt_vtb;               // Applied voltage time
            derivative (V/s)
162     integer RTN_output_EN;
163
164     //Array Size is the maximum defect number (custom)
165     real pos[maximum_number_defects:1];              //
            Random defect location (distance from the cathode)
```

```
166     real delta_R[maximum_number_defects:1];           //
           Resistance variation due to RTN
167     real state[maximum_number_defects:1];             //
           Defect state - empty or filled
168     real last_time[maximum_number_defects:1];         //
           Defect last transition time
169     real p[maximum_number_defects:1];                 //
           Probability of Transition
170     integer type[maximum_number_defects:1];           //
           Defect type being generated - 0 (Oxygen Ions) 1 (
           Oxygen Vacancies)
171     real tau_c[maximum_number_defects:1];             //
           Defect capture time to bottom electrode
172     real tau_e[maximum_number_defects:1];             //
           Defect emission time to bottom electrode
173
174     //RTN random functions
175     integer rtn_rand_seed;
176     integer dr_rand_seed;
177     integer pos_rand_seed;
178     integer trans_rand_seed;
179     integer Erel_rand_seed;
180     integer Et_rand_seed;
181     integer beta_rand_seed;
182
183
184     real Delta_R_tot;
185
186     //Branches definition
```

```
187     branch (n1, gnd_b) b_a;
188     branch (n2, gnd_t) b_b;
189     branch (n3, gnd_t) b_c;
190     branch (TE,BE) b_TB;
191     branch (n1, n11) b_a2;
192     branch (n2, n22) b_b2;
193     branch (n3, n33) b_c2;
194     branch (TE, ME) b_TM;
195     branch (ME, BE) b_MB;
196
197     branch (ME, ME2) b_vcf;
198     branch (ME2, BE) b_vbar;
199
200     //Functions
201     //** Safeexp
202     analog function real safeexp;
203       input x, maxdxdt;
204       real x, maxdxdt;
205
206       safeexp = maxdxdt*tanh(exp(x)/maxdxdt);
207     endfunction
208     //** fsmoothing
209     analog function real fsmoothing;
210       input x, smoothing_param;
211       real x, smoothing_param;
212
213       fsmoothing = 0.5*(1+ x/sqrt(x*x+smoothing_param));
214     endfunction
215
```

```
216
217   analog begin
218     // Network node relationships definition
219     VB(b_a2)<+ 0;
220     VT(b_b2)<+ 0;
221     VT(b_c2)<+ 0;
222     V(b_TM)<+ 0;
223     VT(gnd_t)<+ 0; // voltage reference for temperature
          nature
224     VB(gnd_b)<+ 0; // voltage reference for barrier
          nature
225
226     //Set initial conditions
227     @(initial_step) begin
228       VB(n11, gnd_b)<+ xinit;     // initial barrier
            thickness
229       VT(n22, gnd_t)<+ Tinit;     // initial conductive
            filament temperature
230       VT(n33, gnd_t)<+ Tinit;     // initial barrier
            temperature
231       S_var = S0;                 // initial conductive
            filament section
232       Vtbmin = 0;
233       beta_var = beta;
234       ddt_x = 0;                  // Barrier time
            derivative initialization
235       Delta_R = 0;
236       delta_R_ref = 0;
237       delta_R_ref2 = 0;
```

```
238         flag_set = 0;
239
240         // Random seeds initializations
241         rtn_rand_seed   = rand_seed_ini;
242         dr_rand_seed    = rand_seed_ini;
243         pos_rand_seed   = rand_seed_ini;
244         trans_rand_seed = rand_seed_ini;
245         Erel_rand_seed  = rand_seed_ini;
246         Et_rand_seed    = rand_seed_ini;
247         beta_rand_seed  = rand_seed_ini;
248         // RTN defects initialization
249         RTN_output_EN = 0;
250
251         if(RTN_ON == 1) begin
252           Srtn = S0 * 1e-18;
253           x = xinit * 1e-9;
254           if (x > 0) begin     // RESET defect update
255             N_O = max(0.1,x * Srtn * O_ions_density);   //
                  Avg. Number of O Ions in vol. x*S . The max
                  function prevents errors when the barrier
                  becomes slightly negative.
256             n = $rdist_poisson(rtn_rand_seed, N_O);     //
                  Number of defects random generation
257             if (n > maximum_number_defects) begin       //
                  limiting the maximum number of defects
258               n = maximum_number_defects;
259             end
260
261             for(i = 1; i <= n ; i = i+1) begin
```

```
262             Erel_0 = Erel0_0 + $rdist_uniform(
                   Erel_rand_seed,-Delta_Erel_0,Delta_Erel_0)
                   ;
263             Et_0 = Et0_0 + $rdist_uniform(Et_rand_seed,-
                   Delta_Et_0,Delta_Et_0);
264             pos[i] = x*$rdist_uniform(pos_rand_seed,0,1);
265             delta_R[i] = Rbar*exp($rdist_normal(
                   dr_rand_seed, DeltaR_dist_HRS_mean,
                   DeltaR_dist_HRS_std));
266             tau_c[i] = 1/(const0 * Nc *exp(-pos[i]/
                   lambda_c)*exp(-(pow(Erel_0-(Et_0-phi+Vtb*
                   pos[i]/x),2)/(4*Erel_0*kb*Tcf))));
267             tau_e[i] = 1/(const0 * Nc *exp(-(x-pos[i])/
                   lambda_e)*exp(-Erel_0/(4*kb*Tcf)));
268             state[i] = $rdist_uniform(dr_rand_seed,0,1)
                   >0.5; //random initial state
269             last_time[i] = $abstime;
270             p[i] = 0;
271             type[i] = 0;
272             Delta_R = Delta_R + delta_R[i]*state[i];
273           end
274           for(i = n+1; i <= maximum_number_defects ;i = i
                 + 1) begin
275             pos[i] = 0;
276             delta_R[i] = 0;
277             tau_c[i] = 0;
278             tau_e[i] = 0;
279             state[i] = 0;
280             last_time[i] = 0;
```

```
281             p[i] = 0;
282             type[i] = 0;
283           end
284
285         end else if (xinit == 0) begin // SET defects
              update
286           N_V     = max(0.1,t_ox_rtn *`M_PI *(pow((rt),2)
                +2*rt*sqrt(Srtn/`M_PI))* V_density);//Avg.
                Number of O Vacancies in vol. t_ox*pi*(r_t
                ^2+2rcf*r_t)
287           n      = $rdist_poisson(rtn_rand_seed, N_V); //
                Number of defects random generation
288           if (n > maximum_number_defects) begin
289             n = maximum_number_defects;
290           end
291           for(i = 1; i <= n ; i = i + 1) begin
292             if (i % 2 == 0) begin
293               Erel_V          = Erel0_V + $rdist_uniform(
                    Erel_rand_seed,-Delta_Erel_V,
                    Delta_Erel_V);
294               Et_V          = Et0_V + $rdist_uniform(
                    Et_rand_seed,-Delta_Et_V,Delta_Et_V);
295               pos[i]          = t_ox_rtn*$rdist_uniform(
                    pos_rand_seed,0,1);
296               delta_R[i]      = R_lrs * exp($rdist_normal(
                    dr_rand_seed, ln(1/(2+(2*t_ox_rtn*Srtn)/
                    pow((rt),3))), DeltaR_dist_LRS_std));
297               tau_c[i] = 1/(const0 * Nc * exp(-pos[i]/
                    lambda_c)*exp(-(pow(Erel_V-(Et_V-phi+Vtb
```

```
            *pos[i]/t_ox_rtn),2)/(4*Erel_V*kb*Tcf)))
            );
298         tau_e[i] = 1/(const0 * Nc * exp(-(t_ox_rtn-
            pos[i])/lambda_e)*exp(-Erel_V/(4*kb*Tcf)
            ));
299         type[i]     = 1;
300         end else begin
301         Erel_0          = Erel0_0 + $rdist_uniform(
            Erel_rand_seed,-Delta_Erel_0,
            Delta_Erel_0);
302         Et_0          = Et0_0 + $rdist_uniform(
            Et_rand_seed,-Delta_Et_0,Delta_Et_0);
303         pos[i]          = t_ox_rtn*$rdist_uniform(
            pos_rand_seed,0,1);
304         delta_R[i]      = R_lrs * exp($rdist_normal(
            dr_rand_seed, ln(1/(2+(2*t_ox_rtn*Srtn)/
            pow((rt),3))), DeltaR_dist_LRS_std));
305         tau_c[i] = 1/(const0 * Nc * exp(-pos[i]/
            lambda_c)*exp(-(pow(Erel_0-(Et_0-phi+Vtb
            *pos[i]/t_ox_rtn),2)/(4*Erel_0*kb*Tcf)))
            );
306         tau_e[i] = 1/(const0 * Nc * exp(-(t_ox_rtn-
            pos[i])/lambda_e)*exp(-Erel_0/(4*kb*Tcf)
            ));
307         type[i]     = 0;
308       end
309       state[i] = $rdist_uniform(dr_rand_seed,0,1)
             >0.5;
310       Delta_R = Delta_R + delta_R[i]*state[i];
```

```
311             last_time[i] = $abstime;
312             p[i] = 0;
313           end
314           for(i = n+1; i <= maximum_number_defects ; i =
                 i + 1) begin
315             pos[i] = 0;
316             delta_R[i] = 0;
317             tau_c[i] = 0;
318             tau_e[i] = 0;
319             state[i] = 0;
320             last_time[i] = 0;
321             p[i] = 0;
322             type[i] = 0;
323           end
324         end
325       end
326     end
327
328     //Differential equations auxiliary functions
329     Sp=-ceil((floor(V(b_TB)/(abs(V(b_TB))+10e-25))-1)/2);
          // Sp = 1 when Vtb < 0 else Sp = 0
330     Sn = 1 - Sp; // Sn = 1 when Vtb >= 0 else Sn = 0
331
332     //Bound step computation
333     tstep = (0.1/(1e-1+abs(ddt_x*tstep_param)));
334     $bound_step(max(tstep, Sp * min_time_step_vneg + Sn *
          min_time_step_vpos)); //trick used to dynamically
          increase the time resolution on demand
335
```

```
336     // RESET variability
337     V(N_noise_x) <+ white_noise(1/Fmax);
338     noise_on_x = V(N_noise_x);
339     V(N_noise_x2) <+ white_noise(1/Fmax);
340     noise_on_x2 = V(N_noise_x2);
341
342     Vtb = V(b_TB); //Top-bottom voltage difference
343
344     sig_RoR = Sp*(ddt_Tbar + abs(ddt_Tbar))*
           msigmaRoRvsdTbardt; // sigma_R/R estimation
345     std_x = log((sig_RoR*(Rbar+Rcf))/(R_lrs*beta*exp(Ea/(
           kb*Tbar)))+1)*l; // Noise on x std estimation
346     sig_wn_dx = noise_on_x;
347     n_ddt_x_Rbar = sig_wn_dx *std_x*Fmax;
348     std_x_Tcf = Sp *((ddt_Tcf*msigmaRoRvsdTcfdt*Rcf*S_var
           )/(rho*(1+alpha*(Tcf-Tmeas))))*tcf_noise_fact;
349     n_ddt_x_Rbar2 = std_x_Tcf* noise_on_x2;
350
351     /** Vtb min computation **/
352     Vtbmin = min(Vtbmin,Vtb);
353
354     //Internal variables update
355     barrier = VB(b_a);
356     Tcf = VT(b_b);
357     Tbar = VT(b_c);
358     R_lrs = rho*(t_ox/S_var);
359     Rbar = max(0,(beta_var*R_lrs*(exp(barrier/l)-1)*exp(
           Ea/(kb*(Tbar+1e-15)))));
360     Rcf = (R_lrs*(t_ox-barrier)/t_ox)*(1+alpha*(Tcf-Tmeas
```

```
    ));
361     R = Rbar + Rcf;
362     Vbar = V(b_vbar);
363     Vcf = V(b_vcf);
364     II = I(b_TM);
365
366     // SET event detection
367     // The detection is caused by the crossing of a
        threshold (th_set) by the barrier time derivative
368     @(cross( -ddt_x - th_set,-1 )) flag_set=1;
369
370     // SET & beta Variability update
371     // At the SET event instant, the conductive filament
        cross section is varied randomly
372     V(N_noise_s) <+ white_noise(Wnoise_param_s);
373     sig_wn_s = Sn * V(N_noise_s);
374     sig_wn_s = abs(sig_wn_s)> max_s_noise_amplitude ?
        max_s_noise_amplitude*sig_wn_s/abs(sig_wn_s) :
        sig_wn_s;
375     V(N_noise_beta) <+ white_noise(1/Fmax);
376     sig_wn_beta = Sn * V(N_noise_beta);
377
378     if (flag_set == 1) begin
379       S_var = S0 + sig_wn_s/S0;
380       s_beta = -0.6966/(1+exp(17.02*(Vtbmin-(-1.084))))
          +0.9077;
381       beta_var = exp(ln_beta-ln(sqrt(exp(s_beta)))+
          s_beta*sig_wn_beta);
382       flag_set = 0;
```

```
383       end else begin
384         S_var = S_var;
385         beta_var = beta_var;
386       end
387
388
389       // Barrier time derivative
390       x_b = a*pow(barrier,b);
391       ddt_x = (Sp*(c0*(exp(-(Ead +(g-x_b)*Vtb/t_ox)/(kb*(
             Tcf+1e-15))) ))-Sn*maxdxdt*tanh(barrier*c0*(limexp
             (-(Eag-gg*Vbar/(barrier+1e-3))/(kb*(Tcf+1e-15))))/
             maxdxdt));
392
393       // Barrier clipping terms:
394       //       * tA set dx/dt to 0 when barrier<0 or barrier
             >tox
395       //       * tB increases x when barrier<0
396       //       * tC decreases x when barrier>tox
397       tA = - ddt_x*(fsmoothing(-barrier,smoothing_param)+
             fsmoothing(barrier-t_ox,smoothing_param));
398       tB = safeexp(1e4*(-barrier), maxdxdt);
399       tC = -safeexp(1e4*(barrier-t_ox), maxdxdt);
400
401       // ODE system of equations
402       IB(b_a)<+ ddt_x + tA + tB + tC + n_ddt_x_Rbar2 +
             n_ddt_x_Rbar;
403       IB(b_a)<+ ddt(-VB(b_a));
404
405       ddt_Tcf = (1/Cpcf)*(Vcf*II - kcf * (Tcf-T0)-kex*(Tcf-
```

```
        Tbar));
406     IT(b_b)<+ ddt_Tcf;
407     IT(b_b)<+ ddt(-Tcf);
408
409     ddt_Tbar = (1/Cpb)*(Vbar*II - kbar * (Tbar-T0)-kex*(
            Tbar-Tcf));
410     IT(b_c)<+ ddt_Tbar;
411     IT(b_c)<+ ddt(-Tbar);
412
413     // Output current update
414     I(b_vcf)<+ V0_LRS*sinh(Vcf/V0_LRS)/Rcf;
415     V(b_vbar)<+ V0_HRS*asinh(I(b_vcf)*Rbar/V0_HRS);
416
417     /*******************************
418     * RTN section
419     *******************************/
420     if ((RTN_ON == 1) && analysis("tran")) begin
421       // Scaling of the inputs from the RRAM model
422       x = barrier*1e-9;    // (nm) => (m)
423       Srtn = S_var*1e-18; // (nm^2) => (m^2)
424
425       // When the barrier thickness is changing, a random
              number of defects is generated.
426       //
427       // During a RESET event, the oxygen ions defects
             are randomly distributed along the barrier
             length. Each defect is assigned randomly
             distributed
428       // relaxation and thermal ionization energies (
```

```
            which are used to compute their respective
            capture and emission times) and a randomly
            distributed resistance
429         // variation caused by the defects.
430         //
431         // During a SET event, the same procedure used for
            the RESET event is followed although considering
             both oxygen ions and oxygen vacancy defects.
432
433         //RTN output enable based on oRTNEN input
434         @(cross( V(oRTN_EN) - 2.5,1 )) RTN_output_EN=1;
435         @(cross( V(oRTN_EN) - 2.5,-1 )) RTN_output_EN=0;
436
437
438         if (abs(ddt_x*1e-9) > dxdt_th) begin
439           delta_R_ref = 0;
440           Delta_R_tot = 0;
441           if (x > 1e-10) begin    // RESET defect update
442             N_O = max(0.1,x * Srtn * O_ions_density);// Avg
                  . Number of O Ions in vol. x*S . The max
                  function prevents errors when the barrier
                  becomes slightly negative.
443             n = $rdist_poisson(rtn_rand_seed, N_O); //
                  Number of defects random generation
444             if (n > maximum_number_defects) begin //
                  limiting the maximum number of defects
445               n = maximum_number_defects;
446             end
447
```

```
448             for(i = 1; i <= n ; i = i+1) begin
449               Erel_0 = Erel0_0 + $rdist_uniform(
                     Erel_rand_seed,-Delta_Erel_0,Delta_Erel_0)
                     ;
450               Et_0 = Et0_0 + $rdist_uniform(Et_rand_seed,-
                     Delta_Et_0,Delta_Et_0);
451               pos[i] = x*$rdist_uniform(pos_rand_seed,0,1);
452               delta_R[i] = Rbar*exp($rdist_normal(
                     dr_rand_seed, DeltaR_dist_HRS_mean,
                     DeltaR_dist_HRS_std));
453               tau_c[i] = 1/(const0 * Nc *exp(-pos[i]/
                     lambda_c)*exp(-(pow(Erel_0-(Et_0-phi+Vtb*
                     pos[i]/x),2)/(4*Erel_0*kb*Tcf))));
454               tau_e[i] = 1/(const0 * Nc *exp(-(x-pos[i])/
                     lambda_e)*exp(-Erel_0/(4*kb*Tcf)));
455               state[i] = $rdist_uniform(dr_rand_seed,0,1)
                     >0.5; //random initial state
456               last_time[i] = $abstime;
457               p[i] = 0;
458               type[i] = 0;
459               delta_R_ref = delta_R_ref + delta_R[i]*state[
                     i];
460               Delta_R_tot = Delta_R_tot + delta_R[i];
461             end
462
463             for(i = n+1; i <= maximum_number_defects ;i = i
                    + 1) begin
464               pos[i] = 0;
465               delta_R[i] = 0;
```

```
466             tau_c[i] = 0;
467             tau_e[i] = 0;
468             state[i] = 0;
469             last_time[i] = 0;
470             p[i] = 0;
471             type[i] = 0;
472           end
473           if(n>1)begin       // to prevent divide by 0
474             f_correction = min(R,Delta_R_tot)/Delta_R_tot
                  ;
475             delta_R_ref2 = delta_R_ref*f_correction;
476
477             for(i = 1; i <= n ; i = i+1) begin
478               delta_R[i] = delta_R[i]*f_correction;
479             end
480           end
481         end else if (x <= 1e-10) begin // SET defects
             update
482
483           N_V      = max(0.1,t_ox_rtn *`M_PI *(pow((rt),2)
              +2*rt*sqrt(Srtn/`M_PI))* V_density);//Avg.
              Number of O Vacancies in vol. t_ox*pi*(r_t
              ^2+2rcf*r_t)
484           n      = $rdist_poisson(rtn_rand_seed, N_V); //
              Number of defects random generation
485           if (n > maximum_number_defects) begin
486             n = maximum_number_defects;
487           end
488
```

```
489             for(i = 1; i <= n ; i = i + 1) begin
490               if (i % 2 == 0) begin
491                 Erel_V          = Erel0_V + $rdist_uniform(
                      Erel_rand_seed,-Delta_Erel_V,
                      Delta_Erel_V);
492                 Et_V          = Et0_V + $rdist_uniform(
                      Et_rand_seed,-Delta_Et_V,Delta_Et_V);
493                 pos[i]          = t_ox_rtn*$rdist_uniform(
                      pos_rand_seed,0,1);
494                 delta_R[i]      = R_lrs * exp($rdist_normal(
                      dr_rand_seed, ln(1/(2+(2*t_ox_rtn*Srtn)/
                      pow((rt),3))), DeltaR_dist_LRS_std));
495                 tau_c[i] = 1/(const0 * Nc * exp(-pos[i]/
                      lambda_c)*exp(-(pow(Erel_V-(Et_V-phi+Vtb
                      *pos[i]/t_ox_rtn),2)/(4*Erel_V*kb*Tcf)))
                      );
496                 tau_e[i] = 1/(const0 * Nc * exp(-(t_ox_rtn-
                      pos[i])/lambda_e)*exp(-Erel_V/(4*kb*Tcf)
                      ));
497                 type[i]     = 1;
498                 end else begin
499                 Erel_0          = Erel0_0 + $rdist_uniform(
                      Erel_rand_seed,-Delta_Erel_0,
                      Delta_Erel_0);
500                 Et_0          = Et0_0 + $rdist_uniform(
                      Et_rand_seed,-Delta_Et_0,Delta_Et_0);
501                 pos[i]          = t_ox_rtn*$rdist_uniform(
                      pos_rand_seed,0,1);
502                 delta_R[i]      = R_lrs * exp($rdist_normal(
```

```
                dr_rand_seed, ln(1/(2+(2*t_ox_rtn*Srtn)/
                pow((rt),3))), DeltaR_dist_LRS_std));
503           tau_c[i] = 1/(const0 * Nc * exp(-pos[i]/
                lambda_c)*exp(-(pow(Erel_0-(Et_0-phi+Vtb
                *pos[i]/t_ox_rtn),2)/(4*Erel_0*kb*Tcf)))
                );
504           tau_e[i] = 1/(const0 * Nc * exp(-(t_ox_rtn-
                pos[i])/lambda_e)*exp(-Erel_0/(4*kb*Tcf)
                ));
505           type[i]     = 0;
506         end
507         state[i] = $rdist_uniform(dr_rand_seed,0,1)
              >0.5;
508         delta_R_ref = delta_R_ref + delta_R[i]*state[
              i];
509         Delta_R_tot = Delta_R_tot + delta_R[i];
510
511         last_time[i] = $abstime;
512         p[i] = 0;
513       end
514
515       for(i = n+1; i <= maximum_number_defects ; i =
            i + 1) begin
516         pos[i] = 0;
517         delta_R[i] = 0;
518         tau_c[i] = 0;
519         tau_e[i] = 0;
520         state[i] = 0;
521         last_time[i] = 0;
```

```
522             p[i] = 0;
523             type[i] = 0;
524           end
525
526           if(n>1)begin  // to prevent divide by 0
527             f_correction = min(R,Delta_R_tot)/Delta_R_tot
                  ;
528             delta_R_ref2 = delta_R_ref*f_correction;
529
530             for(i = 1; i <= n ; i = i+1) begin
531               delta_R[i] = delta_R[i]*f_correction;
532             end
533           end
534         end
535         end else begin // Defects' capture and emission
              times update due to voltage and/or temperature
              variations
536
537           if (x > 1e-10) begin // Device in HRS
538             for(i = 1; i <= n ; i = i+1) begin
539               tau_c[i] = 1/(const0 * Nc *exp(-pos[i]/
                    lambda_c)*exp(-(pow(Erel_0-(Et_0-phi+Vtb
                    *pos[i]/x),2)/(4*Erel_0*kb*Tcf))));
540               tau_e[i] = 1/(const0 * Nc *exp(-(x-pos[i])/
                    lambda_e)*exp(-Erel_0/(4*kb*Tcf)));
541             end
542
543           end else begin  // Device in LRS
544             for(i = 1; i <= n ; i = i + 1) begin
```

```
545                 if (i % 2 == 0) begin
546                   tau_c[i] = 1/(const0 * Nc * exp(-pos[i]/
                         lambda_c)*exp(-(pow(Erel_V-(Et_V-phi+
                         Vtb*pos[i]/t_ox_rtn),2)/(4*Erel_V*kb*
                         Tcf))));
547                   tau_e[i] = 1/(const0 * Nc * exp(-(
                         t_ox_rtn-pos[i])/lambda_e)*exp(-Erel_V
                         /(4*kb*Tcf)));
548                 end else begin
549                   tau_c[i] = 1/(const0 * Nc * exp(-pos[i]/
                         lambda_c)*exp(-(pow(Erel_0-(Et_0-phi+
                         Vtb*pos[i]/t_ox_rtn),2)/(4*Erel_0*kb*
                         Tcf))));
550                   tau_e[i] = 1/(const0 * Nc * exp(-(
                         t_ox_rtn-pos[i])/lambda_e)*exp(-Erel_0
                         /(4*kb*Tcf)));
551               end
552             end
553           end
554         end
555
556         // RTN Waveform formation, Transition Management
557         // Each defect is randomly activated or de-
               activated in relation to its last transition
               time and capture and emission times.
558         // When a defect is active, it does not contribute
               to the current, thus its resistance
               contributions is accumulated in the variable
               Delta_R.
```

```
559
560         Delta_R = 0;
561         for(i = 1; i <= n; i = i + 1) begin
562           if (state[i] == 0) begin
563             tau = tau_c[i];
564             end else begin
565             tau=tau_e[i];
566           end
567
568           p[i] = 1 - exp(-($abstime - last_time[i])/tau);
                 // Transition probability
569
570           if (p[i] > $rdist_uniform(trans_rand_seed,0,1))
                 begin
571             last_time[i] = $abstime;
572             if (state[i] == 0) begin
573               state[i] = 1;
574             end else begin
575               state[i] = 0;
576             end
577           end
578           Delta_R = Delta_R + delta_R[i]*state[i]; //
                 Cumulative resistance variation due to defects
579         end
580
581         // RTN output current
582         if (RTN_output_EN==1) begin
583           I_rtn = (Delta_R-delta_R_ref2)*II/R;
584           I(b_TB)<+ I_rtn;//b_MB
```

```
585         end
586
587       end
588
589     end
590   endmodule
```

II Disciplines [112]

```
1     `ifdef ADDITIONAL_DISCIPLINES
2     `else
3     `define ADDITIONAL_DISCIPLINES 1
4
5       nature barrier_thickness
6         access = VB;
7         abstol = 0.001;
8         units = "nm";
9         blowup = 1e40;
10      endnature
11
12      nature barrier_variation
13        access = IB;
14        abstol = 1e40;
15        blowup = 1e40;
16        units = "nm/s";
17      endnature
18
19      discipline my_barrier
```

```
20      potential barrier_thickness;
21      flow barrier_variation;
22      enddiscipline
23
24
25      nature my_temperature_value
26        access = VT;
27        abstol = 1e3;
28        units = "K";
29      endnature
30
31      nature my_temperature_variation
32        access = IT;
33        abstol = 1e3;
34        units = "K/s";
35      endnature
36
37      discipline my_temperature
38      potential my_temperature_value;
39      flow my_temperature_variation;
40      enddiscipline
41
42    `endif
```

Bibliography

[1] "IoT devices 'to generate nearly 80 zettabytes of data' by 2025 - arrow," Arrow Business Communications. (Jun. 19, 2019), [Online]. Available: https://www.arrowcommunications.co.uk/iot-devices-to-generate-nearly-80-zettabytes-of-data-by-2025/ (visited on 01/15/2022).

[2] J. Backus, "Can programming be liberated from the von neumann style?: A functional style and its algebra of programs," *Commun. ACM*, vol. 21, no. 8, pp. 613–641, Aug. 1978, ISSN: 0001-0782. DOI: 10.1145/359576.359579. [Online]. Available: http://doi.acm.org/10.1145/359576.359579 (visited on 10/04/2019).

[3] Y. Xi, B. Gao, J. Tang, *et al.*, "In-memory learning with analog resistive switching memory: A review and perspective," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 14–42, Jan. 2021, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2020.3004543. [Online]. Available: https://ieeexplore.ieee.org/document/9138706/ (visited on 01/10/2022).

[4] D. V. Christensen, R. Dittmann, B. Linares-Barranco, *et al.*, "2021 roadmap on neuromorphic computing and engineering," p. 153,

[5] W. Zhang, B. Gao, J. Tang, *et al.*, "Neuro-inspired computing chips," *Nature Electronics*, vol. 3, no. 7, pp. 371–382, Jul. 2020, Number: 7 Publisher: Nature Publishing Group, ISSN: 2520-1131. DOI: 10.1038/s41928-020-0435-7. [Online]. Available: https://www.nature.com/articles/s41928-020-0435-7 (visited on 09/25/2020).

[6] I. Chakraborty, A. Jaiswal, A. K. Saha, S. K. Gupta, and K. Roy, "Pathways to efficient neuromorphic computing with non-volatile memory technologies," *Applied Physics Reviews*, vol. 7, no. 2, p. 021 308, Jun. 2020, ISSN: 1931-9401. DOI: 10.1063/1.5113536. [Online]. Available: http://aip.scitation.org/doi/10.1063/1.5113536 (visited on 01/10/2022).

[7] I. Chakraborty, M. Ali, A. Ankit, *et al.*, "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2276–2310, Dec. 2020, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2020.3003007. [Online]. Available: https://ieeexplore.ieee.org/document/9141337/ (visited on 01/10/2022).

[8] J. D. Kendall and S. Kumar, "The building blocks of a brain-inspired computer," *Applied Physics Reviews*, vol. 7, no. 1, p. 011 305, Mar. 2020, ISSN: 1931-9401. DOI: 10.1063/1.5129306. [Online]. Available: http://aip.scitation.org/doi/10.1063/1.5129306 (visited on 01/10/2022).

[9] P. Mannocci, G. Pedretti, E. Giannone, E. Melacarne, Z. Sun, and D. Ielmini, "A universal, analog, in-memory computing primitive for linear algebra using memristors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 12, pp. 4889–4899, Dec. 2021,

Conference Name: IEEE Transactions on Circuits and Systems I: Regular Papers, ISSN: 1558-0806. DOI: 10.1109/TCSI.2021.3122278.

[10] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2013.2282132.

[11] S. Kvatinsky, D. Belousov, S. Liman, *et al.*, "MAGIC—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014, Conference Name: IEEE Transactions on Circuits and Systems II: Express Briefs, ISSN: 1558-3791. DOI: 10.1109/TCSII.2014.2357292.

[12] J. Reuben, "Rediscovering majority logic in the post-CMOS era: A perspective from in-memory computing," *Journal of Low Power Electronics and Applications*, vol. 10, no. 3, p. 28, Sep. 2020, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/jlpea10030028. [Online]. Available: https://www.mdpi.com/2079-9268/10/3/28 (visited on 08/30/2021).

[13] B. Parhami, D. Abedi, and G. Jaberipur, "Majority-logic, its applications, and atomic-scale embodiments," *Computers & Electrical Engineering*, vol. 83, p. 106 562, May 1, 2020, ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2020.106562. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790619315290 (visited on 06/15/2020).

[14] R. Naous, A. Siemon, M. Schulten, *et al.*, "Theory and experimental verification of configurable computing with stochastic memristors,"

Scientific Reports, vol. 11, no. 1, p. 4218, Feb. 18, 2021, Bandiera_abtest: a Cc_license_type: cc_by Cg_type: Nature Research Journals Number: 1 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Electrical and electronic engineering;Electronic devices;Information technology Subject_term_id: electrical-and-electronic-engineering;electronic-devices;information-technology, ISSN: 2045-2322. DOI: 10.1038/s41598-021-83382-y. [Online]. Available: https://www.nature.com/articles/s41598-021-83382-y (visited on 01/11/2022).

[15] S. Raj, D. Chakraborty, and S. K. Jha, "In-memory flow-based stochastic computing on memristor crossbars using bit-vector stochastic streams," in *2017 IEEE 17th International Conference on Nanotechnology (IEEE-NANO)*, ISSN: 1944-9380, Jul. 2017, pp. 855–860. DOI: 10.1109/NANO.2017.8117440.

[16] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Applied Physics Reviews*, vol. 7, no. 3, p. 031 301, Sep. 2020, ISSN: 1931-9401. DOI: 10.1063/1.5143815. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.5143815 (visited on 01/10/2022).

[17] Z. Sun, E. Ambrosi, G. Pedretti, A. Bricalli, and D. Ielmini, "In-memory PageRank accelerator with a cross-point array of resistive memories," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1466–1470, Apr. 2020, Conference Name: IEEE Transactions on Electron Devices, ISSN: 1557-9646. DOI: 10.1109/TED.2020.2966908.

[18] S. Wang, Z. Sun, Y. Liu, *et al.*, "Optimization schemes for in-memory linear regression circuit with memristor arrays," *IEEE Transactions*

on Circuits and Systems I: Regular Papers, vol. 68, no. 12, pp. 4900–4909, Dec. 2021, Conference Name: IEEE Transactions on Circuits and Systems I: Regular Papers, ISSN: 1558-0806. DOI: 10.1109/TCSI.2021.3122327.

[19] A. Serb, A. Corna, R. George, *et al.*, "Memristive synapses connect brain and silicon spiking neurons," *Scientific Reports*, vol. 10, no. 1, p. 2590, Dec. 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-58831-9. [Online]. Available: http://www.nature.com/articles/s41598-020-58831-9 (visited on 01/10/2022).

[20] A. Kurenkov, S. Fukami, and H. Ohno, "Neuromorphic computing with antiferromagnetic spintronics," *Journal of Applied Physics*, vol. 128, no. 1, p. 010902, Jul. 7, 2020, ISSN: 0021-8979, 1089-7550. DOI: 10.1063/5.0009482. [Online]. Available: http://aip.scitation.org/doi/10.1063/5.0009482 (visited on 01/10/2022).

[21] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, "Deep in-memory architectures in SRAM: An analog approach to approximate computing," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2251–2275, Dec. 2020, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2020.3034117. [Online]. Available: https://ieeexplore.ieee.org/document/9252843/ (visited on 01/10/2022).

[22] H.-T. Lue, P.-K. Hsu, K.-C. Wang, and C.-Y. Lu, "Introduction of non-volatile computing in memory (nvCIM) by 3d NAND flash for inference accelerator of deep neural network (DNN) and the read disturb reliability evaluation : (invited paper)," in *2020 IEEE International Reliability Physics Symposium (IRPS)*, ISSN: 1938-1891, Apr. 2020, pp. 1–6. DOI: 10.1109/IRPS45951.2020.9128340.

[23] F. Gao, G. Tziantzioulis, and D. Wentzlaff, "ComputeDRAM: In-memory compute using off-the-shelf DRAMs," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52, New York, NY, USA: Association for Computing Machinery, Oct. 12, 2019, pp. 100–113, ISBN: 978-1-4503-6938-1. DOI: 10.1145/3352460.3358260. [Online]. Available: https://doi.org/10.1145/3352460.3358260 (visited on 01/13/2022).

[24] W. Shim, H. Jiang, X. Peng, and S. Yu, "Architectural design of 3d NAND flash based compute-in-memory for inference engine," in *The International Symposium on Memory Systems*, ser. MEMSYS 2020, New York, NY, USA: Association for Computing Machinery, Sep. 28, 2020, pp. 77–85, ISBN: 978-1-4503-8899-3. DOI: 10.1145/3422575.3422779. [Online]. Available: https://doi.org/10.1145/3422575.3422779 (visited on 01/13/2022).

[25] S. Mittal, G. Verma, B. Kaushik, and F. A. Khanday, "A survey of SRAM-based in-memory computing techniques and applications," *Journal of Systems Architecture*, vol. 119, p. 102 276, Oct. 1, 2021, ISSN: 1383-7621. DOI: 10.1016/j.sysarc.2021.102276. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001909 (visited on 01/14/2022).

[26] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, Apr. 2010, ISSN: 1476-4687. DOI: 10.1038/nature08940. [Online]. Available: https://www.nature.com/articles/nature08940 (visited on 08/26/2019).

[27] Q. Chen, X. Wang, H. Wan, and R. Yang, "A logic circuit design for perfecting memristor-based material implication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 2, pp. 279–284, Feb. 2017, ISSN: 0278-0070. DOI: 10.1109/TCAD.2016.2578881.

[28] S. Yu, Z. Li, P. Chen, *et al.*, "Binary neural network with 16 mb RRAM macro chip for classification and online training," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2016, pp. 16.2.1–16.2.4. DOI: 10.1109/IEDM.2016.7838429.

[29] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao, "Emerging NVM: A survey on architectural integration and research challenges," *ACM Transactions on Design Automation of Electronic Systems*, vol. 23, no. 2, 14:1–14:32, Nov. 14, 2017, ISSN: 1084-4309. DOI: 10.1145/3131848. [Online]. Available: https://doi.org/10.1145/3131848 (visited on 01/14/2022).

[30] R. F. Freitas and W. W. Wilcke, "Storage-class memory: The next storage system technology," *IBM Journal of Research and Development*, vol. 52, no. 4, pp. 439–447, Jul. 2008, ISSN: 0018-8646, 0018-8646. DOI: 10.1147/rd.524.0439. [Online]. Available: http://ieeexplore.ieee.org/document/5388608/ (visited on 01/14/2022).

[31] J. S. Meena, S. M. Sze, U. Chand, and T.-Y. Tseng, "Overview of emerging nonvolatile memory technologies," *Nanoscale Research Letters*, vol. 9, no. 1, p. 526, Sep. 25, 2014, ISSN: 1931-7573. DOI: 10.1186/1556-276X-9-526. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4182445/ (visited on 01/14/2022).

[32] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM Journal of Research and Development*, vol. 52, no. 4, pp. 449–464, Jul. 2008, ISSN: 0018-8646, 0018-8646. DOI: 10.1147/rd.524.0449. [Online]. Available: http://ieeexplore.ieee.org/document/5388605/ (visited on 01/14/2022).

[33] G. Pedretti and D. Ielmini, "In-memory computing with resistive memory circuits: Status and outlook," *Electronics*, vol. 10, no. 9, p. 1063, Jan. 2021, Number: 9 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/electronics10091063. [Online]. Available: https://www.mdpi.com/2079-9292/10/9/1063 (visited on 05/20/2021).

[34] H.-S. P. Wong, H.-Y. Lee, S. Yu, *et al.*, "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2012.2190369. [Online]. Available: http://ieeexplore.ieee.org/document/6193402/ (visited on 10/07/2019).

[35] D. Ielmini, "Resistive switching memories based on metal oxides: Mechanisms, reliability and scaling," *Semiconductor Science and Technology*, vol. 31, no. 6, p. 063 002, May 2016, Publisher: IOP Publishing, ISSN: 0268-1242. DOI: 10.1088/0268-1242/31/6/063002. [Online]. Available: https://doi.org/10.1088/0268-1242/31/6/063002 (visited on 01/14/2022).

[36] M. N. Kozicki and H. J. Barnaby, "Conductive bridging random access memory—materials, devices and applications," *Semiconductor Science and Technology*, vol. 31, no. 11, p. 113 001, Nov. 1, 2016, ISSN:

0268-1242, 1361-6641. DOI: 10.1088/0268-1242/31/11/113001. [Online]. Available: https://iopscience.iop.org/article/10.1088/0268-1242/31/11/113001 (visited on 01/14/2022).

[37] M.-C. Wu, T. Yi-Hsin, C. Jui-Yuan, and W. Wen-Wei, "Low power consumption nanofilamentary ECM and VCM cells in a single sidewall of high-density VRRAM arrays," *Adv. Sci.*, vol. 6, no. 24, p. 1 902 363, 2019. DOI: https://doi.org/10.1002/advs.201902363. (visited on 03/02/2020).

[38] S. W. Fong, C. M. Neumann, and H.-S. P. Wong, "Phase-change memory—towards a storage-class memory," *IEEE Transactions on Electron Devices*, vol. 64, no. 11, pp. 4374–4385, Nov. 2017, Conference Name: IEEE Transactions on Electron Devices, ISSN: 1557-9646. DOI: 10.1109/TED.2017.2746342.

[39] S. Slesazeck and T. Mikolajick, "Nanoscale resistive switching memory devices: A review," *Nanotechnology*, vol. 30, no. 35, p. 352 003, Aug. 30, 2019, ISSN: 0957-4484, 1361-6528. DOI: 10.1088/1361-6528/ab2084. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6528/ab2084 (visited on 02/27/2020).

[40] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333–343, Jun. 2018, ISSN: 2520-1131. DOI: 10.1038/s41928-018-0092-2. [Online]. Available: http://www.nature.com/articles/s41928-018-0092-2 (visited on 03/03/2020).

[41] H. Mulaosmanovic, E. T. Breyer, S. Dünkel, S. Beyer, T. Mikolajick, and S. Slesazeck, "Ferroelectric field-effect transistors based on HfO2: A review," vol. 32, no. 50, p. 502 002, Sep. 2021, Publisher: IOP Pub-

lishing, ISSN: 0957-4484. DOI: 10.1088/1361-6528/ac189f. [Online]. Available: https://doi.org/10.1088/1361-6528/ac189f (visited on 10/26/2021).

[42] A. Chen, "A review of emerging non-volatile memory (NVM) technologies and applications," *Solid-State Electronics*, vol. 125, pp. 25 – 38, 2016, ISSN: 0038-1101. DOI: https://doi.org/10.1016/j.sse.2016.07.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038110116300867.

[43] N. Maciel, E. Marques, L. Naviner, Y. Zhou, and H. Cai, "Magnetic tunnel junction applications," *Sensors*, vol. 20, no. 1, p. 121, Dec. 24, 2019, ISSN: 1424-8220. DOI: 10.3390/s20010121. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/121 (visited on 01/14/2022).

[44] X. Han, X. Wang, C. Wan, G. Yu, and X. Lv, "Spin-orbit torques: Materials, physics, and devices," *Applied Physics Letters*, vol. 118, no. 12, p. 120 502, Mar. 22, 2021, ISSN: 0003-6951. DOI: 10.1063/5.0039147. [Online]. Available: https://aip.scitation.org/doi/full/10.1063/5.0039147 (visited on 02/21/2022).

[45] M. Natsui, A. Tamakoshi, H. Honjo, *et al.*, "Dual-port SOT-MRAM achieving 90-MHz read and 60-MHz write operations under field-assistance-free condition," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1116–1128, Apr. 2021, Conference Name: IEEE Journal of Solid-State Circuits, ISSN: 1558-173X. DOI: 10.1109/JSSC.2020.3039800.

[46] K. Garello, F. Yasin, and G. S. Kar, "Spin-orbit torque MRAM for ultrafast embedded memories: From fundamentals to large scale tech-

nology integration," in *2019 IEEE 11th International Memory Workshop (IMW)*, ISSN: 2573-7503, May 2019, pp. 1–4. DOI: 10.1109/IMW.2019.8739466.

[47] Y. Cao, G. Xing, H. Lin, N. Zhang, H. Zheng, and K. Wang, "Prospect of spin-orbitronic devices and their applications," *iScience*, vol. 23, no. 10, p. 101614, Oct. 23, 2020, ISSN: 2589-0042. DOI: 10.1016/j.isci.2020.101614. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2589004220308063 (visited on 02/22/2022).

[48] B. Hoberman and J.-P. Nozieres, "SOT-MRAM – third generation MRAM memory opens new opportunities : Hot chips conference august 2021," in *2021 IEEE Hot Chips 33 Symposium (HCS)*, ISSN: 2573-2048, Aug. 2021, pp. 1–10. DOI: 10.1109/HCS52781.2021.9567072.

[49] F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, "Resistive random access memory (RRAM): An overview of materials, switching mechanism, performance, multilevel cell (mlc) storage, modeling, and applications," *Nanoscale Research Letters*, vol. 15, no. 1, p. 90, Apr. 22, 2020, ISSN: 1556-276X. DOI: 10.1186/s11671-020-03299-9. [Online]. Available: https://doi.org/10.1186/s11671-020-03299-9 (visited on 01/12/2022).

[50] J. Song, H. Dixit, B. Behin-Aein, C. H. Kim, and W. Taylor, "Impact of process variability on write error rate and read disturbance in STT-MRAM devices," *IEEE Transactions on Magnetics*, vol. 56, no. 12, pp. 1–11, Dec. 2020, Conference Name: IEEE Transactions on Magnetics, ISSN: 1941-0069. DOI: 10.1109/TMAG.2020.3028045.

[51] H. Honjo, T. V. A. Nguyen, T. Watanabe, *et al.*, "First demonstration of field-free SOT-MRAM with 0.35 ns write speed and 70 thermal stability under 400°c thermal tolerance by canted SOT structure and its advanced patterning/SOT channel technology," in *2019 IEEE International Electron Devices Meeting (IEDM)*, ISSN: 2156-017X, Dec. 2019, pp. 28.5.1–28.5.4. DOI: 10.1109/IEDM19573.2019.8993443.

[52] N. Du, H. Schmidt, and I. Polian, "Low-power emerging memristive designs towards secure hardware systems for applications in internet of things," *Nano Materials Science*, Nano Energy Materials and Devices for Miniaturized Electronics and Smart Systems, vol. 3, no. 2, pp. 186–204, Jun. 1, 2021, ISSN: 2589-9651. DOI: 10.1016/j.nanoms.2021.01.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2589965121000015 (visited on 01/10/2022).

[53] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/JPROC.2014.2335155.

[54] J. Lee, S. Choi, D. Kim, Y. Choi, and W. Sun, "A novel hardware security architecture for IoT device: PD-CRP (PUF database and challenge–response pair) bloom filter on memristor-based PUF," *Applied Sciences*, vol. 10, no. 19, p. 6692, Jan. 2020, Number: 19 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/app10196692. [Online]. Available: https://www.mdpi.com/2076-3417/10/19/6692 (visited on 01/10/2022).

[55] Y. Pang, B. Gao, B. Lin, H. Qian, and H. Wu, "Memristors for hardware security applications," *Advanced Electronic Materials*, vol. 5, no. 9, p. 1 800 872, 2019, ISSN: 2199-160X. DOI: 10.1002/aelm.201800872. [Online]. Available: https://www.onlinelibrary.wiley.com/doi/abs/10.1002/aelm.201800872 (visited on 11/06/2020).

[56] Y. Kim, R. Daly, J. Kim, *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, Minneapolis, MN, USA: IEEE, Jun. 2014, pp. 361–372, ISBN: 978-1-4799-4394-4 978-1-4799-4396-8. DOI: 10.1109/ISCA.2014.6853210. [Online]. Available: http://ieeexplore.ieee.org/document/6853210/ (visited on 01/11/2022).

[57] Y. Du, L. Jing, H. Fang, *et al.*, "Exploring the impact of random telegraph noise-induced accuracy loss on resistive RAM-based deep neural network," *IEEE Transactions on Electron Devices*, vol. 67, no. 8, pp. 3335–3340, Aug. 2020, Conference Name: IEEE Transactions on Electron Devices, ISSN: 1557-9646. DOI: 10.1109/TED.2020.3002736.

[58] S. Lv, J. Liu, and Z. Geng, "Application of memristors in hardware security: A current state-of-the-art technology," *Advanced Intelligent Systems*, vol. 3, no. 1, p. 2 000 127, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.10 ISSN: 2640-4567. DOI: 10.1002/aisy.202000127. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202000127 (visited on 01/10/2022).

[59] Y. Gao, S. F. Al-Sarawi, and D. Abbott, "Physical unclonable functions," *Nature Electronics*, vol. 3, no. 2, pp. 81–91, Feb. 2020, Bandiera_abtest: a Cg_type: Nature Research Journals Number: 2 Primary_atype: Re-

views Publisher: Nature Publishing Group Subject_term: Electrical and electronic engineering;Electronics, photonics and device physics;Information technology Subject_term_id: electrical-and-electronic-engineering;electronics-photonics-and-device-physics;information-technology, ISSN: 2520-1131. DOI: 10.1038/s41928-020-0372-5. [Online]. Available: https://www.nature.com/articles/s41928-020-0372-5 (visited on 01/10/2022).

[60] K. S. Woo, J. Kim, J. Han, J. M. Choi, W. Kim, and C. S. Hwang, "A high-speed true random number generator based on a CuxTe1-x diffusive memristor," *Advanced Intelligent Systems*, vol. 3, no. 7, p. 2100062, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.2021 ISSN: 2640-4567. DOI: 10.1002/aisy.202100062. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202100062 (visited on 01/10/2022).

[61] L. Yang, L. Cheng, Y. Li, *et al.*, "Cryptographic key generation and in situ encryption in one-transistor-one-resistor memristors for hardware security," *Advanced Electronic Materials*, vol. 7, no. 5, p. 2001182, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aelm.202001182, ISSN: 2199-160X. DOI: 10.1002/aelm.202001182. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aelm.202001182 (visited on 01/10/2022).

[62] M. Uddin, A. S. Shanta, M. Badruddoja Majumder, M. S. Hasan, and G. S. Rose, "Memristor crossbar PUF based lightweight hardware security for IoT," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, ISSN: 2158-4001, Jan. 2019, pp. 1–4. DOI: 10.1109/ICCE.2019.8661912.

[63] J. F. Gibbons and W. E. Beadle, "Switching properties of thin nio films," *Solid-State Electronics*, vol. 7, no. 11, pp. 785–790, Nov. 1, 1964, ISSN: 0038-1101. DOI: 10.1016/0038-1101(64)90131-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0038110164901315 (visited on 01/14/2022).

[64] T. W. Hickmott, "Low-frequency negative resistance in thin anodic oxide films," *Journal of Applied Physics*, vol. 33, no. 9, pp. 2669–2682, Sep. 1, 1962, Publisher: American Institute of Physics, ISSN: 0021-8979. DOI: 10.1063/1.1702530. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.1702530 (visited on 01/14/2022).

[65] I. Baek, M. Lee, S. Seo, *et al.*, "Highly scalable nonvolatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses," in *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.*, Dec. 2004, pp. 587–590. DOI: 10.1109/IEDM.2004.1419228.

[66] A. Padovani, L. Larcher, F. M. Puglisi, and P. Pavan, "Multiscale modeling of defect-related phenomena in high-k based logic and memory devices," in *2017 IEEE 24th International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, Jul. 2017, pp. 1–6. DOI: 10.1109/IPFA.2017.8060063.

[67] G. Bersuker, D. C. Gilmer, D. Veksler, *et al.*, "Metal oxide RRAM switching mechanism based on conductive filament microscopic properties," in *2010 International Electron Devices Meeting*, ISSN: 2156-017X, Dec. 2010, pp. 19.6.1–19.6.4. DOI: 10.1109/IEDM.2010.5703394.

[68] S. Larentis, F. Nardi, S. Balatti, D. C. Gilmer, and D. Ielmini, "Resistive switching by voltage-driven ion migration in bipolar RRAM—part

II: Modeling," *IEEE Transactions on Electron Devices*, vol. 59, no. 9, pp. 2468–2475, Sep. 2012, Conference Name: IEEE Transactions on Electron Devices, ISSN: 1557-9646. DOI: 10.1109/TED.2012.2202320.

[69] F. M. Puglisi, L. Larcher, A. Padovani, and P. Pavan, "Bipolar resistive RAM based on HfO2: Physics, compact modeling, and variability control," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 171–184, Jun. 2016, Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems, ISSN: 2156-3365. DOI: 10.1109/JETCAS.2016.2547703.

[70] M. Lanza, H.-S. P. Wong, E. Pop, *et al.*, "Recommended methods to study resistive switching devices," *Advanced Electronic Materials*, vol. 5, no. 1, p. 1 800 143, 2019, ISSN: 2199-160X. DOI: 10.1002/aelm.201800143. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aelm.201800143 (visited on 02/05/2020).

[71] F. M. Puglisi, L. Larcher, A. Padovani, and P. Pavan, "A complete statistical investigation of RTN in HfO2-based RRAM in high resistive state," *IEEE Transactions on Electron Devices*, vol. 62, no. 8, pp. 2606–2613, Aug. 2015, Conference Name: IEEE Transactions on Electron Devices, ISSN: 1557-9646. DOI: 10.1109/TED.2015.2439812.

[72] E. Pérez, A. Grossi, C. Zambelli, P. Olivo, and C. Wenger, "Impact of the incremental programming algorithm on the filament conduction in HfO2-based RRAM arrays," *IEEE Journal of the Electron Devices Society*, vol. 5, no. 1, pp. 64–68, Jan. 2017, ISSN: 2168-6734. DOI: 10.1109/JEDS.2016.2618425.

[73] E. Pérez, A. Grossi, C. Zambelli, P. Olivo, R. Roelofs, and C. Wenger, "Reduction of the cell-to-cell variability in hf1-xAlxOyBased RRAM

arrays by using program algorithms," *IEEE Electron Device Letters*, vol. 38, no. 2, pp. 175–178, Feb. 2017, ISSN: 0741-3106, 1558-0563. DOI: 10.1109/LED.2016.2646758.

[74] F. M. Puglisi, N. Zagni, L. Larcher, and P. Pavan, "Random telegraph noise in resistive random access memories: Compact modeling and advanced circuit design," *IEEE Transactions on Electron Devices*, vol. 65, no. 7, pp. 2964–2972, Jul. 2018, ISSN: 0018-9383. DOI: 10.1109/TED.2018.2833208.

[75] H. Li, Z. Jiang, P. Huang, *et al.*, "Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2015, pp. 1425–1430. DOI: 10.7873/DATE.2015.0362.

[76] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, ISSN: 1530-1591, Mar. 2019, pp. 1769–1774. DOI: 10.23919/DATE.2019.8715178.

[77] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL — memristor ratioed logic," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, Turin, Italy: IEEE, Aug. 2012, pp. 1–6, ISBN: 978-1-4673-0289-0 978-1-4673-0287-6 978-1-4673-0288-3. DOI: 10.1109/CNNA.2012.6331426. [Online]. Available: http://ieeexplore.ieee.org/document/6331426/ (visited on 01/13/2022).

[78] J. Reuben, D. Fey, and C. Wenger, "A modeling methodology for resistive RAM based on stanford-PKU model with extended multilevel capability," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 647–

656, 2019, ISSN: 1536-125X, 1941-0085. DOI: 10.1109/TNANO.2019.2922838.

[79] J. Yu, H. A. Du Nguyen, M. Abu Lebdeh, M. Taouil, and S. Hamdioui, "Enhanced scouting logic: A robust memristive logic design scheme," in *2019 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, ISSN: 2327-8226, Jul. 2019, pp. 1–6. DOI: 10.1109/NANOARCH47378.2019.181296.

[80] L. Xie, H. Du Nguyen, J. Yu, *et al.*, "Scouting logic: A novel memristor-based logic design for resistive computing," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, ISSN: 2159-3477, Jul. 2017, pp. 176–181. DOI: 10.1109/ISVLSI.2017.39.

[81] E. Lehtonen, J. Poikonen, and M. Laiho, "Implication logic synthesis methods for memristors," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 2441–2444. DOI: 10.1109/ISCAS.2012.6271792.

[82] T. Zanotti, F. M. Puglisi, and P. Pavan, "Reliability-aware design strategies for stateful logic-in-memory architectures," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 278–285, Jun. 2020. DOI: 10.1109/TDMR.2020.2981205.

[83] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," *Journal of Physics D: Applied Physics*, vol. 51, no. 28, p. 283 001, Jun. 2018, Publisher: IOP Publishing, ISSN: 0022-3727. DOI: 10.1088/1361-6463/aac8a5. [Online]. Available: https://doi.org/10.1088%2F1361-6463%2Faac8a5 (visited on 03/19/2020).

[84] D. Ielmini and G. Pedretti, "Device and circuit architectures for in-memory computing," *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2 000 040, 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202000040, ISSN: 2640-4567. DOI: 10.1002/aisy.202000040. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202000040 (visited on 10/20/2021).

[85] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv:1602.02830 [cs]*, Mar. 17, 2016. arXiv: 1602.02830. [Online]. Available: http://arxiv.org/abs/1602.02830 (visited on 06/16/2020).

[86] X. Sun, X. Peng, P. Chen, R. Liu, J. Seo, and S. Yu, "Fully parallel RRAM synaptic array for implementing binary neural network with (+1, -1) weights and (+1, 0) neurons," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2018, pp. 574–579. DOI: 10.1109/ASPDAC.2018.8297384.

[87] O. Krestinskaya, O. Otaniyozov, and A. P. James, "Binarized neural network with stochastic memristors," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Mar. 2019, pp. 274–275. DOI: 10.1109/AICAS.2019.8771565.

[88] D. Tirfe and V. K. Anand, "A survey on trends of two-factor authentication," in *Contemporary Issues in Communication, Cloud and Big Data Analytics*, H. K. D. Sarma, V. E. Balas, B. Bhuyan, and N. Dutta, Eds., ser. Lecture Notes in Networks and Systems, Singapore: Springer, 2022, pp. 285–296, ISBN: 9789811642449. DOI: 10.1007/978-981-16-4244-9_23.

[89] A. Chen and M.-R. Lin, "Reset switching probability of resistive switching devices," *IEEE Electron Device Letters*, vol. 32, no. 5, pp. 590–592, May 2011, Conference Name: IEEE Electron Device Letters, ISSN: 1558-0563. DOI: 10.1109/LED.2011.2109933.

[90] H. Jiang, D. Belkin, S. E. Savel'ev, *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nature Communications*, vol. 8, no. 1, p. 882, Oct. 12, 2017, Bandiera_abtest: a Cc_license_type: cc_by Cg_type: Nature Research Journals Number: 1 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Electrical and electronic engineering;Electronic devices Subject_term_id: electrical-and-electronic-engineering;electronic-devices, ISSN: 2041-1723. DOI: 10.1038/s41467-017-00869-x. [Online]. Available: https://www.nature.com/articles/s41467-017-00869-x (visited on 01/14/2022).

[91] T. Zhang, M. Yin, C. Xu, *et al.*, "High-speed true random number generation based on paired memristors for security electronics," *Nanotechnology*, vol. 28, no. 45, p. 455 202, Nov. 10, 2017, ISSN: 0957-4484, 1361-6528. DOI: 10.1088/1361-6528/aa8b3a. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6528/aa8b3a (visited on 01/07/2022).

[92] J. Yang, J. Xu, B. Wang, *et al.*, "A low cost and high reliability true random number generator based on resistive random access memory," in *2015 IEEE 11th International Conference on ASIC (ASICON)*, ISSN: 2162-755X, Nov. 2015, pp. 1–4. DOI: 10.1109/ASICON.2015.7516996.

[93] R. Brederlow, R. Prakash, C. Paulus, and R. Thewes, "A low-power true random number generator using random telegraph noise of single oxide-traps," in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, ISSN: 2376-8606, Feb. 2006, pp. 1666–1675. DOI: 10.1109/ISSCC.2006.1696222.

[94] J. Brown, R. Gao, Z. Ji, *et al.*, "A low-power and high-speed true random number generator using generated RTN," in *2018 IEEE Symposium on VLSI Technology*, ISSN: 2158-9682, Jun. 2018, pp. 95–96. DOI: 10.1109/VLSIT.2018.8510671.

[95] Z. Wei, Y. Katoh, S. Ogasahara, *et al.*, "True random number generator using current difference based on a fractional stochastic model in 40-nm embedded ReRAM," in *2016 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, USA: IEEE, Dec. 2016, pp. 4.8.1–4.8.4, ISBN: 978-1-5090-3902-9. DOI: 10.1109/IEDM.2016.7838349. [Online]. Available: http://ieeexplore.ieee.org/document/7838349/ (visited on 01/07/2022).

[96] J. Kim, H. Nili, N. D. Truong, *et al.*, "Nano-intrinsic true random number generation: A device to data study," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2615–2626, Jul. 2019, ISSN: 1549-8328, 1558-0806. DOI: 10.1109/TCSI.2019.2895045. [Online]. Available: https://ieeexplore.ieee.org/document/8637965/ (visited on 01/07/2022).

[97] 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, p. 8, Jan. 10, 2018, ISSN: 1556-276X.

DOI: 10.1186/s11671-017-2419-8. [Online]. Available: https://doi.org/10.1186/s11671-017-2419-8 (visited on 11/04/2020).

[98] C. C. McAndrew, G. J. Coram, K. K. Gullapalli, *et al.*, "Best practices for compact modeling in verilog-a," *IEEE Journal of the Electron Devices Society*, vol. 3, no. 5, pp. 383–396, Sep. 2015, ISSN: 2168-6734. DOI: 10.1109/JEDS.2015.2455342.

[99] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Letters*, vol. 32, no. 10, pp. 1436–1438, Oct. 2011, ISSN: 0741-3106. DOI: 10.1109/LED.2011.2163292.

[100] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: ThrEshold adaptive memristor model," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013, ISSN: 1549-8328. DOI: 10.1109/TCSI.2012.2215714.

[101] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015, ISSN: 1549-7747. DOI: 10.1109/TCSII.2015.2433536.

[102] I. Messaris, A. Serb, S. Stathopoulos, A. Khiat, S. Nikolaidis, and T. Prodromakis, "A data-driven verilog-a ReRAM model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3151–3162, Dec. 2018, ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2018.2791468. [Online]. Available: https://ieeexplore.ieee.org/document/8252766/ (visited on 09/17/2020).

[103] C. Yakopcic, T. M. Taha, D. J. Mountain, T. Salter, M. J. Marinella, and M. McLean, "Memristor model optimization based on parameter extraction from device characterization data," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1084–1095, May 2020, Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, ISSN: 1937-4151. DOI: 10.1109/TCAD.2019.2912946.

[104] I. Messaris, S. Nikolaidis, A. Serb, *et al.*, "A TiO2 ReRAM parameter extraction method," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2379-447X, May 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050789.

[105] X. Guan, S. Yu, and H.-P. Wong, "A SPICE compact model of metal oxide resistive switching memory with variations," *IEEE Electron Device Letters*, vol. 33, no. 10, pp. 1405–1407, Oct. 2012, ISSN: 0741-3106. DOI: 10.1109/LED.2012.2210856.

[106] P. Huang, B. Chen, H. Li, *et al.*, "Parameters extraction on HfOX based RRAM," in *2014 44th European Solid State Device Research Conference (ESSDERC)*, ISSN: 2378-6558, Sep. 2014, pp. 250–253. DOI: 10.1109/ESSDERC.2014.6948807.

[107] C. L. Torre, A. F. Zurhelle, T. Breuer, R. Waser, and S. Menzel, "Compact modeling of complementary switching in oxide-based ReRAM devices," *IEEE Transactions on Electron Devices*, vol. 66, no. 3, pp. 1268–1275, Mar. 2019, ISSN: 0018-9383. DOI: 10.1109/TED.2019.2892997.

[108] C. Bengel, A. Siemon, F. Cuppers, *et al.*, "Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using SPICE level compact models," *IEEE Transactions on Circuits and*

Systems I: Regular Papers, vol. 67, no. 12, pp. 4618–4630, Dec. 2020, ISSN: 1549-8328, 1558-0806. DOI: 10.1109/TCSI.2020.3018502. [Online]. Available: https://ieeexplore.ieee.org/document/9181475/ (visited on 12/02/2021).

[109] G González-Cordero, M. B. González, F Campabadal, F Jiménez-Molinos, and J. B. Roldán, "A physically based SPICE model for RRAMs including RTN," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, ISSN: 2640-5563, Nov. 2020, pp. 1–6. DOI: 10.1109/DCIS51330.2020.9268665.

[110] G. González-Cordero, J. B. Roldan, F. Jiménez-Molinos, J. Suñé, S. Long, and M. Liu, "A new compact model for bipolar RRAMs based on truncated-cone conductive filaments—a verilog-a approach," *Semiconductor Science and Technology*, vol. 31, no. 11, p. 115013, Oct. 2016, Publisher: IOP Publishing, ISSN: 0268-1242. DOI: 10.1088/0268-1242/31/11/115013. [Online]. Available: https://doi.org/10.1088/0268-1242/31/11/115013 (visited on 11/29/2021).

[111] G. González-Cordero, M. B. González, H. García, *et al.*, "A physically based model for resistive memories including a detailed temperature and variability description," *Microelectronic Engineering*, Special issue of Insulating Films on Semiconductors (INFOS 2017), vol. 178, pp. 26–29, Jun. 25, 2017, ISSN: 0167-9317. DOI: 10.1016/j.mee.2017.04.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167931717301533 (visited on 11/29/2021).

[112] F. M. Puglisi, T. Zanotti, and P. Pavan, "Unimore resistive random access memory (RRAM) verilog-a model," *nanoHUB*, 2019. DOI: 10.21981/15GF-KX29.

[113] T. Wang, "Modelling multistability and hysteresis in ESD clamps, memristors and other devices," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2017, pp. 1–10. DOI: 10.1109/CICC.2017.7993681.

[114] L. Larcher, F. M. Puglisi, P. Pavan, A. Padovani, L. Vandelli, and G. Bersuker, "A compact model of program window in HfOx RRAM devices for conductive filament characteristics analysis," *IEEE Transactions on Electron Devices*, vol. 61, no. 8, pp. 2668–2673, Aug. 2014. DOI: 10.1109/TED.2014.2329020.

[115] S. Yu, Ximeng Guan, and H.-S. P. Wong, "On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, monte carlo simulation, and experimental characterization," in *2011 International Electron Devices Meeting*, Washington, DC, USA: IEEE, Dec. 2011, pp. 17.3.1–17.3.4, ISBN: 978-1-4577-0505-2 978-1-4577-0506-9 978-1-4577-0504-5. DOI: 10.1109/IEDM.2011.6131572. [Online]. Available: http://ieeexplore.ieee.org/document/6131572/ (visited on 09/06/2019).

[116] J. G. Simmons, "Generalized formula for the electric tunnel effect between similar electrodes separated by a thin insulating film," *Journal of Applied Physics*, vol. 34, no. 6, pp. 1793–1803, Jun. 1, 1963, Publisher: American Institute of Physics, ISSN: 0021-8979. DOI: 10.1063/1.1702682. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.1702682 (visited on 12/11/2021).

[117] T. Zanotti, F. M. Puglisi, and P. Pavan, "Reconfigurable smart in-memory computing platform supporting logic and binarized neural networks for low-power edge devices," *IEEE Journal on Emerging and*

Selected Topics in Circuits and Systems, vol. 10, no. 4, pp. 478–487, Dec. 2020. DOI: 10.1109/JETCAS.2020.3030542.

[118] F. M. Puglisi°, N. Zagni, L. Larcher, and P. Pavan, "A new verilog-a compact model of random telegraph noise in oxide-based RRAM for advanced circuit design," in *2017 47th European Solid-State Device Research Conference (ESSDERC)*, Sep. 2017, pp. 204–207. DOI: 10.1109/ESSDERC.2017.8066627.

[119] A. S. Foster, F. Lopez Gejo, A. L. Shluger, and R. M. Nieminen, "Vacancy and interstitial defects in hafnia," *Physical Review B*, vol. 65, no. 17, p. 174 117, May 2, 2002, Publisher: American Physical Society. DOI: 10.1103/PhysRevB.65.174117. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.65.174117 (visited on 01/15/2021).

[120] U. Celano, A. Fantini, R. Degraeve, M. Jurczak, L. Goux, and W. Vandervorst, "Scalability of valence change memory: From devices to tip-induced filaments," *AIP Advances*, vol. 6, no. 8, p. 085 009, Aug. 2016, ISSN: 2158-3226. DOI: 10.1063/1.4961150. [Online]. Available: http://aip.scitation.org/doi/10.1063/1.4961150 (visited on 12/03/2021).

[121] S. Balatti, S. Ambrogio, D. Ielmini, and D. C. Gilmer, "Variability and failure of set process in HfO2 RRAM," in *2013 5th IEEE International Memory Workshop*, ISSN: 2159-483X, 2159-4864, May 2013, pp. 38–41. DOI: 10.1109/IMW.2013.6582092.

[122] Z. Jiang, S. Yu, Y. Wu, J. H. Engel, X. Guan, and H.-P. Wong, "Verilog-a compact model for oxide-based resistive random access memory (RRAM)," in *2014 International Conference on Simulation of*

Semiconductor Processes and Devices (SISPAD), Sep. 2014, pp. 41–44. DOI: 10.1109/SISPAD.2014.6931558.

[123] Z. Jiang, Y. Wu, S. Yu, *et al.*, "A compact model for metal–oxide resistive random access memory with experiment verification," *IEEE Transactions on Electron Devices*, vol. 63, no. 5, pp. 1884–1892, May 2016, ISSN: 0018-9383. DOI: 10.1109/TED.2016.2545412.

[124] F. M. Puglisi, A. Qafa, and P. Pavan, "Temperature impact on the reset operation in HfO_2 RRAM," *IEEE Electron Device Letters*, vol. 36, no. 3, pp. 244–246, Mar. 2015, ISSN: 0741-3106, 1558-0563. DOI: 10.1109/LED.2015.2397192. [Online]. Available: http://ieeexplore.ieee.org/document/7036054/ (visited on 10/08/2019).

[125] E. Hildebrandt, J. Kurian, M. M. Müller, T. Schroeder, H.-J. Kleebe, and L. Alff, "Controlled oxygen vacancy induced p-type conductivity in HfO2-x thin films," *Applied Physics Letters*, vol. 99, no. 11, p. 112902, Sep. 12, 2011, ISSN: 0003-6951. DOI: 10.1063/1.3637603. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.3637603 (visited on 12/15/2021).

[126] N. Capron, P. Broqvist, and A. Pasquarello, "Migration of oxygen vacancy in HfO_2 and across the HfO_2/SiO_2 interface: A first-principles investigation," *Applied Physics Letters*, vol. 91, no. 19, p. 192905, Nov. 5, 2007, ISSN: 0003-6951. DOI: 10.1063/1.2807282. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.2807282 (visited on 12/15/2021).

[127] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and

large-scale modeling," in *2012 International Electron Devices Meeting*, ISSN: 2156-017X, Dec. 2012, pp. 10.4.1–10.4.4. DOI: 10.1109/IEDM.2012.6479018.

[128] J. Woo, J. Song, K. Moon, S. Lee, J. Park, and H. Hwang, "Multilevel conductance switching of a HfO2 RRAM array induced by controlled filament for neuromorphic applications," in *2016 IEEE Silicon Nanoelectronics Workshop (SNW)*, Honolulu, HI, USA: IEEE, Jun. 2016, pp. 40–41, ISBN: 978-1-5090-0726-4. DOI: 10.1109/SNW.2016.7577975. [Online]. Available: http://ieeexplore.ieee.org/document/7577975/ (visited on 10/09/2019).

[129] S. Yu, Y. Wu, Y. Chai, J. Provine, and H.-S. P. Wong, "Characterization of switching parameters and multilevel capability in HfOx/AlOx bi-layer RRAM devices," in *Proceedings of 2011 International Symposium on VLSI Technology, Systems and Applications*, Hsinchu, Taiwan: IEEE, Apr. 2011, pp. 1–2, ISBN: 978-1-4244-8493-5. DOI: 10.1109/VTSA.2011.5872251. [Online]. Available: http://ieeexplore.ieee.org/document/5872251/ (visited on 10/09/2019).

[130] J. Woo, K. Moon, J. Song, *et al.*, "Improved synaptic behavior under identical pulses using AlOx/HfO2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Letters*, vol. 37, no. 8, pp. 994–997, Aug. 2016. DOI: 10.1109/LED.2016.2582859.

[131] Z. Fang, H. Y. Yu, X. Li, N. Singh, G. Q. Lo, and D. L. Kwong, "HfOx/TiOx/HfOx/TiOx multilayer-based forming-free RRAM devices with excellent uniformity," *IEEE Electron Device Letters*, vol. 32, no. 4, pp. 566–568, Apr. 2011, Conference Name: IEEE Electron Device Letters, ISSN: 1558-0563. DOI: 10.1109/LED.2011.2109033.

[132] T. Zanotti, P. Pavan, and F. M. Puglisi, "Multi-input logic-in-memory for ultra-low power non-von neumann computing," *Micromachines*, vol. 12, no. 10, p. 1243, Oct. 2021. DOI: 10.3390/mi12101243. (visited on 10/18/2021).

[133] T. Zanotti, F. M. Puglisi, and P. Pavan, "Smart logic-in-memory architecture for ultra-low power large fan-in operations," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Aug. 2020, pp. 31–35. DOI: 10.1109/AICAS48895.2020.9073870.

[134] T. Zanotti, F. M. Puglisi, and P. Pavan, "Low-bit precision neural network architecture with high immunity to variability and random telegraph noise based on resistive memories," in *2021 IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2021, pp. 1–6. DOI: 10.1109/IRPS46558.2021.9405103.

[135] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, Jul. 2009, pp. 33–36. DOI: 10.1109/NANOARCH.2009.5226356.

[136] E. Lehtonen, J. H. Poikonen, and M. Laiho, "Two memristors suffice to compute all boolean functions," *Electronics Letters*, vol. 46, no. 3, pp. 239–240, Feb. 2010, ISSN: 0013-5194. DOI: 10.1049/el.2010.3407.

[137] B. R. a. A. N. Whitehead, *Principia Mathematica Vol I.* Cambridge University Press, 1910. [Online]. Available: https://commons.wikimedia.org/wiki/File:Russell,_Whitehead_-_Principia_Mathematica,_vol._I,_1910.djvu (visited on 10/07/2019).

[138] M. E. Fouda, A. M. Eltawil, and F. Kurdahi, "Modeling and analysis of passive switching crossbar arrays," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 270–282, Jan. 2018, ISSN: 1558-0806. DOI: 10.1109/TCSI.2017.2714101.

[139] J. E. Stine, I. Castellanos, M. Wood, *et al.*, "FreePDK: An open-source variation-aware design kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, Jun. 2007, pp. 173–174. DOI: 10.1109/MSE.2007.44.

[140] J. H. Poikonen, E. Lehtonen, and M. Laiho, "On synthesis of boolean expressions for memristive devices using sequential implication logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 1129–1134, Jul. 2012, ISSN: 0278-0070. DOI: 10.1109/TCAD.2012.2187524.

[141] A. Siemon, R. Drabinski, M. J. Schultis, *et al.*, "Stateful three-input logic with memristive switches," *Scientific Reports*, vol. 9, no. 1, pp. 1–13, Oct. 10, 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-51039-6. [Online]. Available: https://www.nature.com/articles/s41598-019-51039-6 (visited on 10/15/2019).

[142] F. S. Marranghello, V. Callegaro, M. G. A. Martins, A. I. Reis, and R. P. Ribas, "Factored forms for memristive material implication stateful logic," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 267–278, Jun. 2015, Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems, ISSN: 2156-3365. DOI: 10.1109/JETCAS.2015.2426511.

[143] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Fast logic synthesis for RRAM-based in-memory computing using majority-

inverter graphs," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, ISSN: 1558-1101, Mar. 2016, pp. 948–953.

[144] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Logic synthesis for RRAM-based in-memory computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1422–1435, Jul. 2018, Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, ISSN: 1937-4151. DOI: 10.1109/TCAD.2017.2750064.

[145] M. S. Rahman, R. Hasib, B. Sultana, M. G. Hussain, M. Rahman, and M. A. Rahaman, "An extensive karnaugh mapping tool for boolean expression simplification," in *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, Dec. 2019, pp. 1–5. DOI: 10.1109/STI47673.2019.9068037.

[146] Y. M. Movsisyan and V. A. Aslanyan, "On computation of de morgan and quasi-de morgan functions," in *Ninth International Conference on Computer Science and Information Technologies Revised Selected Papers*, Sep. 2013, pp. 1–6. DOI: 10.1109/CSITechnol.2013.6710334.

[147] M. C. Morgül and M. Altun, "Optimal and heuristic algorithms to synthesize lattices of four-terminal switches," *Integration*, vol. 64, pp. 60–70, Jan. 1, 2019, ISSN: 0167-9260. DOI: 10.1016/j.vlsi.2018.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926018301524 (visited on 12/29/2021).

[148] C. Wang, H. Wu, B. Gao, *et al.*, "Ultrafast RESET analysis of HfOx-based RRAM by sub-nanosecond pulses," *Advanced Electronic Materials*, vol. 3, no. 12, p. 1 700 263, 2017, ISSN: 2199-160X. DOI: 10.1002/

aelm.201700263. [Online]. Available: https://www.onlinelibrary.wiley.com/doi/abs/10.1002/aelm.201700263 (visited on 08/26/2019).

[149] T. Zanotti, F. M. Puglisi, and P. Pavan, "Reliability and performance analysis of logic-in-memory based binarized neural networks," *IEEE Transactions on Device and Materials Reliability*, vol. 21, no. 2, pp. 183–191, Jun. 2021. DOI: 10.1109/TDMR.2021.3075200.

[150] T. Zanotti, F. M. Puglisi, and P. Pavan, "A smart logic-in-memory architecture for low-power non-von neumann computing," *IEEE Journal of the Electron Devices Society*, pp. 1–1, 2020. DOI: 10.1109/JEDS.2020.2987402.

[151] P. Junsangsri, J. Han, and F. Lombardi, "Logic-in-memory with a nonvolatile programmable metallization cell," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 2, pp. 521–529, Feb. 2016, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2015.2411258.

[152] S.-y. Park, D. Jung, J.-u. Kang, J.-s. Kim, and J. Lee, "CFLRU: A replacement algorithm for flash memory," in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES '06, event-place: Seoul, Korea, New York, NY, USA: ACM, 2006, pp. 234–241, ISBN: 978-1-59593-543-4. DOI: 10.1145/1176760.1176789. [Online]. Available: http://doi.acm.org/10.1145/1176760.1176789 (visited on 08/26/2019).

[153] M. Aguirre-Hernandez and M. Linares-Aranda, "CMOS full-adders for energy-efficient arithmetic applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 4, pp. 718–721, Apr. 2011, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2009.2038166.

[154] A. K. Yadav, B. P. Shrivatava, and A. K. Dadoriya, "Low power high speed 1-bit full adder circuit design at 45nm CMOS technology," in *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, Oct. 2017, pp. 427–432. DOI: 10.1109/RISE.2017.8378203.

[155] S. Sharma and G. Soni, "Comparision analysis of FinFET based 1-bit full adder cell implemented using different logic styles at 10, 22 and 32nm," in *2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS)*, Apr. 2016, pp. 660–667. DOI: 10.1109/ICEETS.2016.7583835.

[156] J. Zhou, F. Cai, Q. Wang, B. Chen, S. Gaba, and W. D. Lu, "Very low-programming-current RRAM with self-rectifying characteristics," *IEEE Electron Device Letters*, vol. 37, no. 4, pp. 404–407, Apr. 2016. DOI: 10.1109/LED.2016.2530942.

[157] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/5.726791.

[158] K. Berggren, Q. Xia, K. K. Likharev, *et al.*, "Roadmap on emerging hardware and technology for machine learning," *Nanotechnology*, vol. 32, no. 1, p. 012002, Oct. 2020, Publisher: IOP Publishing, ISSN: 0957-4484. DOI: 10.1088/1361-6528/aba70f. [Online]. Available: https://doi.org/10.1088%2F1361-6528%2Faba70f (visited on 11/01/2020).

[159] B. McDanel, S. Teerapittayanon, and H. T. Kung, "Embedded binarized neural networks," p. 6,

[160] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv:1606.06160 [cs]*, Feb. 1, 2018. arXiv: 1606.06160. [Online]. Available: http://arxiv.org/abs/1606.06160 (visited on 10/14/2020).

[161] W. He, S. Yin, Y. Kim, *et al.*, "2-bit-per-cell RRAM-based in-memory computing for area-/energy-efficient deep learning," *IEEE Solid-State Circuits Letters*, vol. 3, pp. 194–197, 2020, Conference Name: IEEE Solid-State Circuits Letters, ISSN: 2573-9603. DOI: 10.1109/LSSC.2020.3010795.

[162] D. Joksas, P. Freitas, Z. Chai, *et al.*, "Committee machines—a universal method to deal with non-idealities in memristor-based neural networks," *Nature Communications*, vol. 11, no. 1, p. 4273, Aug. 26, 2020, ISSN: 2041-1723. DOI: 10.1038/s41467-020-18098-0. [Online]. Available: https://www.nature.com/articles/s41467-020-18098-0 (visited on 09/25/2020).

[163] S. Yin, Y. Kim, X. Han, *et al.*, "Monolithically integrated RRAM- and CMOS-based in-memory computing optimizations for efficient deep learning," *IEEE Micro*, vol. 39, no. 6, pp. 54–63, Nov. 2019, ISSN: 1937-4143. DOI: 10.1109/MM.2019.2943047.

[164] A. Sedra and K. Smith, "A second-generation current conveyor and its applications," *IEEE Transactions on Circuit Theory*, vol. 17, no. 1, pp. 132–134, Feb. 1970, Conference Name: IEEE Transactions on Circuit Theory, ISSN: 2374-9555. DOI: 10.1109/TCT.1970.1083067.

[165] P. Benoit, L. Dalmasso, G. Patrigeon, T. Gil, F. Bruguier, and L. Torres, "Edge-computing perspectives with reconfigurable hardware," in

2019 14th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), ISSN: 2642-7222, Jul. 2019, pp. 51–58. DOI: 10.1109/ReCoSoC48741.2019.9034961.

[166] J. Vieira, E. Giacomin, Y. Qureshi, *et al.*, "A product engine for energy-efficient execution of binary neural networks using resistive memories," in *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, ISSN: 2324-8440, Oct. 2019, pp. 160–165. DOI: 10.1109/VLSI-SoC.2019.8920343.

[167] W. Yi, Y. Kim, and J.-J. Kim, "Effect of device variation on mapping binary neural network to memristor crossbar array," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, ISSN: 1558-1101, Mar. 2019, pp. 320–323. DOI: 10.23919/DATE.2019.8714817.

[168] Y.-F. Qin, R. Kuang, X.-D. Huang, Y. Li, J. Chen, and X.-S. Miao, "Design of high robustness BNN inference accelerator based on binary memristors," *IEEE Transactions on Electron Devices*, vol. 67, no. 8, pp. 3435–3441, Aug. 2020, Conference Name: IEEE Transactions on Electron Devices, ISSN: 1557-9646. DOI: 10.1109/TED.2020.2998457.

[169] W.-H. Chen, C. Dou, K.-X. Li, *et al.*, "CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors," *Nature Electronics*, vol. 2, no. 9, pp. 420–428, Sep. 2019, ISSN: 2520-1131. DOI: 10.1038/s41928-019-0288-0. [Online]. Available: https://www.nature.com/articles/s41928-019-0288-0 (visited on 01/13/2020).

[170] W. Wan, R. Kubendran, B. Gao, *et al.*, "A voltage-mode sensing scheme with differential-row weight mapping for energy-efficient RRAM-

based in-memory computing," in *2020 IEEE Symposium on VLSI Technology*, ISSN: 2158-9682, Jun. 2020, pp. 1–2. DOI: 10.1109/VLSITechnology18217.2020.9265066.

[171] T. Zanotti, F. M. Puglisi, and P. Pavan, "Energy-efficient non-von neumann computing architecture supporting multiple computing paradigms for logic and binarized neural networks," *Journal of Low Power Electronics and Applications*, vol. 11, no. 3, p. 29, Sep. 2021. DOI: 10.3390/jlpea11030029. (visited on 08/30/2021).

[172] F. M. Puglisi, T. Zanotti, and P. Pavan, "Optimized synthesis method for ultra-low power multi-input material implication logic with emerging non-volatile memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–11, 2021. DOI: 10.1109/TCSI.2021.3079986.

[173] T. Zanotti, C. Zambelli, F. M. Puglisi, *et al.*, "Reliability of logic-in-memory circuits in resistive memory arrays," *IEEE Transactions on Electron Devices*, vol. 67, no. 11, pp. 4611–4615, Nov. 2020, ISSN: 1557-9646. DOI: 10.1109/TED.2020.3025271.

[174] F. M. Puglisi, T. Zanotti, and P. Pavan, "SIMPLY: Design of a RRAM-based smart logic-in-memory architecture using RRAM compact model," in *ESSDERC 2019 - 49th European Solid-State Device Research Conference (ESSDERC)*, ISSN: 1930-8876, Sep. 2019, pp. 130–133. DOI: 10.1109/ESSDERC.2019.8901731.

[175] T. Zanotti, F. M. Puglisi, and P. Pavan, "Circuit reliability analysis of RRAM-based logic-in-memory crossbar architectures including line parasitic effects, variability, and random telegraph noise," in *2020 IEEE International Reliability Physics Symposium (IRPS)*,

ISSN: 1938-1891, Apr. 2020, pp. 1–5. DOI: 10.1109/IRPS45951.2020.9128343.

[176] T. Zanotti, F. M. Puglisi, and P. Pavan, "Circuit reliability of low-power RRAM-based logic-in-memory architectures," in *2019 IEEE International Integrated Reliability Workshop (IIRW)*, Oct. 2019, pp. 1–5. DOI: 10.1109/IIRW47491.2019.8989875.

[177] T. Zanotti, F. M. Puglisi, and P. Pavan, "Circuit reliability analysis of in-memory inference in binarized neural networks," in *2020 IEEE International Integrated Reliability Workshop (IIRW)*, Oct. 2020, pp. 1–5. DOI: 10.1109/IIRW49815.2020.9312858.

[178] T. Zanotti, P. Pavan, and F. M. Puglisi, "Performances and trade-offs of low-bit precision neural networks based on resistive memories," in *2021 IEEE International Integrated Reliability Workshop (IIRW)*, ISSN: 2374-8036, Oct. 2021, pp. 1–5. DOI: 10.1109/IIRW53245.2021.9635626.

[179] F. M. Puglisi, P. Pavan, and T. Zanotti, "Metodo di lettura per circuiti del tipo logic-in-memory e relativa architettura circuitale," pat. IT201900014688A1, Nov. 12, 2019.

[180] R. Carboni and D. Ielmini, "Stochastic memory devices for security and computing," *Advanced Electronic Materials*, vol. 5, no. 9, p. 1 900 198, Sep. 2019, ISSN: 2199-160X, 2199-160X. DOI: 10.1002/aelm.201900198. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/aelm.201900198 (visited on 01/07/2022).

[181] X. Li, T. Zanotti, T. Wang, K. Zhu, F. M. Puglisi, and M. Lanza, "Random telegraph noise in metal-oxide memristors for true random

number generators: A materials study," *Advanced Functional Materials*, p. 2 102 172, Apr. 23, 2021.

[182] C. Wen, X. Li, T. Zanotti, *et al.*, "Advanced data encryption using 2d materials," *Advanced Materials*, vol. 33, no. 27, p. 2 100 185, 2021. DOI: 10.1002/adma.202100185.

[183] F. M. Puglisi, A. Padovani, L. Larcher, and P. Pavan, "Random telegraph noise: Measurement, data analysis, and interpretation," in *2017 IEEE 24th International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, ISSN: 1946-1550, Jul. 2017, pp. 1–9. DOI: 10.1109/IPFA.2017.8060057.

[184] R. Stępień and J. Walczak, "Statistical analysis of the LFSR generators in the NIST STS test suite," p. 7,

[185] O. Potii, N. Poluyanenko, I. Stelnyk, I. Revak, S. Kavun, and T. Kuznetsova, "Nonlinear-feedback shift registers for stream ciphers," in *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Jul. 2019, pp. 906–911. DOI: 10.1109/UKRCON.2019.8879786.

[186] E. Dubrova, "A list of maximum period NLFSRs," 166, 2012. [Online]. Available: http://eprint.iacr.org/2012/166 (visited on 11/12/2020).

[187] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-90b, Jan. 2018, NIST SP 800–90b. DOI: 10.6028/NIST.SP.800-90B. [Online]. Available: https:

//nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf (visited on 01/07/2022).

[188] R. De Rose, T. Zanotti, F. Maria Puglisi, F. Crupi, P. Pavan, and M. Lanuzza, "STT-MTJ based smart implication for energy-efficient logic-in-memory computing," *Solid-State Electronics*, p. 108 065, May 28, 2021. DOI: 10.1016/j.sse.2021.108065.

[189] S. Gao, B. Chen, Y. Qu, and Y. Zhao, "MRAM acceleration core for vector matrix multiplication and XNOR-binarized neural network inference," in *2020 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, ISSN: 1930-8868, Aug. 2020, pp. 153–154. DOI: 10.1109/VLSI-TSA48913.2020.9203740.

List of publications

International Refereed Journals

Zanotti, T., Puglisi, F. M., Pavan, P., "Reconfigurable smart in-memory computing platform supporting logic and binarized neural networks for low-power edge devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 478–487, Dec. 2020. DOI: 10.1109/JETCAS.2020.3030542.

Zanotti, T., Zambelli, C., Puglisi, F. M., "Reliability of logic-in-memory circuits in resistive memory arrays," *IEEE Transactions on Electron Devices*, vol. 67, no. 11, pp. 4611–4615, Nov. 2020, ISSN: 1557-9646. DOI: 10.1109/TED.2020.3025271.

Zanotti, T., Puglisi, F. M., Pavan, P., "A smart logic-in-memory architecture for low-power non-von neumann computing," *IEEE Journal of the Electron Devices Society*, pp. 1–1, 2020. DOI: 10.1109/JEDS.2020.2987402.

Zanotti, T., Puglisi, F. M., Pavan, P., "Reliability-aware design strategies for stateful logic-in-memory architectures," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 278–285, Jun. 2020. DOI: 10.1109/TDMR.2020.2981205.

De Rose, R., **Zanotti, T.**, Maria Puglisi, F., Crupi, F., Pavan, P., Lanuzza, M., "STT-MTJ based smart implication for energy-efficient logic-in-memory computing," *Solid-State Electronics*, p. 108 065, May 28, 2021. DOI: 10.1016/j.sse.2021.108065.

Li, X., **Zanotti, T.**, Wang, T., Zhu, K., Puglisi, F. M., Lanza, M., "Random telegraph noise in metal-oxide memristors for true random number generators: A materials study," *Advanced Functional Materials*, p. 2 102 172, Apr. 23, 2021.

Puglisi, F. M., **Zanotti, T.**, Pavan, P., "Optimized synthesis method for ultra-low power multi-input material implication logic with emerging non-volatile memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–11, 2021. DOI: 10.1109/TCSI.2021.3079986.

Wen, C., Li, X., **Zanotti, T.,** "Advanced data encryption using 2d materials," *Advanced Materials*, vol. 33, no. 27, p. 2 100 185, 2021. DOI: 10.1002/adma.202100185.

Zanotti, T., Pavan, P., Puglisi, F. M., "Multi-input logic-in-memory for ultra-low power non-von neumann computing," *Micromachines*, vol. 12, no. 10, p. 1243, Oct. 2021. DOI: 10.3390/mi12101243. (visited on 10/18/2021).

Zanotti, T., Puglisi, F. M., Pavan, P., "Energy-efficient non-von neumann computing architecture supporting multiple computing paradigms for logic and binarized neural networks," *Journal of Low Power Electronics and Applications*, vol. 11, no. 3, p. 29, Sep. 2021. DOI: 10.3390/jlpea11030029. (visited on 08/30/2021).

Zanotti, T., Puglisi, F. M., Pavan, P., "Reliability and performance analysis of logic-in-memory based binarized neural networks," *IEEE Transactions*

on Device and Materials Reliability, vol. 21, no. 2, pp. 183–191, Jun. 2021. DOI: 10.1109/TDMR.2021.3075200.

International Conference Proceedings

Puglisi, F. M., **Zanotti, T.**, Pavan, P., "SIMPLY: Design of a RRAM-based smart logic-in-memory architecture using RRAM compact model," in *ESSDERC 2019 - 49th European Solid-State Device Research Conference (ESSDERC)*, ISSN: 1930-8876, Sep. 2019, pp. 130–133. DOI: 10.1109/ESSDERC.2019.8901731.

Zanotti, T., Puglisi, F. M., Pavan, P., "Circuit reliability of low-power RRAM-based logic-in-memory architectures," in *2019 IEEE International Integrated Reliability Workshop (IIRW)*, Oct. 2019, pp. 1–5. DOI: 10.1109/IIRW47491.2019.8989875.

Zanotti, T., Puglisi, F. M., Pavan, P., "Circuit reliability analysis of in-memory inference in binarized neural networks," in *2020 IEEE International Integrated Reliability Workshop (IIRW)*, Oct. 2020, pp. 1–5. DOI: 10.1109/IIRW49815.2020.9312858.

Zanotti, T., Puglisi, F. M., Pavan, P., "Circuit reliability analysis of RRAM-based logic-in-memory crossbar architectures including line parasitic effects, variability, and random telegraph noise," in *2020 IEEE International Reliability Physics Symposium (IRPS)*, ISSN: 1938-1891, Apr. 2020, pp. 1–5. DOI: 10.1109/IRPS45951.2020.9128343.

Zanotti, T., Puglisi, F. M., Pavan, P., "Smart logic-in-memory architecture for ultra-low power large fan-in operations," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems*

(AICAS), Aug. 2020, pp. 31–35. DOI: 10.1109/AICAS48895.2020.9073870.

Zanotti, T., Pavan, P., Puglisi, F. M., "Performances and trade-offs of low-bit precision neural networks based on resistive memories," in *2021 IEEE International Integrated Reliability Workshop (IIRW)*, ISSN: 2374-8036, Oct. 2021, pp. 1–5. DOI: 10.1109/IIRW53245.2021.9635626.

Zanotti, T., Puglisi, F. M., Pavan, P., "Low-bit precision neural network architecture with high immunity to variability and random telegraph noise based on resistive memories," in *2021 IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2021, pp. 1–6. DOI: 10.1109/IRPS46558.2021.9405103.

Zanotti, T., "Reliability and prospects of logic-in-memory circuits," in *2022 IEEE 6th Electron Devices Technology and Manufacturing Conference (EDTM)*, Mar. 2022. DOI: (in\:press).

Compact Model

Puglisi, F. M., **Zanotti, T.**, Pavan, P., "Unimore resistive random access memory (RRAM) verilog-a model," *nanoHUB*, 2019. DOI: 10.21981/15GF-KX29.

Patent Applications

Puglisi, F. M., Pavan, P., **Zanotti, T.,** "Metodo di lettura per circuiti del tipo logic-in-memory e relativa architettura circuitale," pat. IT201900014688A1, Nov. 12, 2019.

Awards and Recognitions

Puglisi, F. M., **Zanotti, T.**, Pavan, P., "Best paper award," presented at the ESSDERC 2019 - 49th European Solid-State Device Research Conference (ESSDERC), Sep. 2019.

Zanotti, T., Puglisi, F. M., Pavan, P., "Best student paper award," presented at the 2020 IEEE International Integrated Reliability Workshop (IIRW), Oct. 2020.

Zanotti, T., "2021 travel award winner, granted by the journal of low power electronics and applications (JLPEA)," Feb. 2021.