IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 March 2023, accepted 1 May 2023, date of publication 8 May 2023, date of current version 15 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3273899

RESEARCH ARTICLE

MemCA: All-Memristor Design for Deterministic and Probabilistic Cellular Automata Hardware Realization

VASILEIOS NTINAS[1,2,3], (Member, IEEE), IOSIF-ANGELOS FYRIGOS[3], (Member, IEEE), RAFAILIA-ELENI KARAMANI[3], (Member, IEEE), NIKOLAOS VASILEIADIS[3,4], (Member, IEEE), PANAGIOTIS DIMITRAKIS[4], (Senior Member, IEEE), ANTONIO RUBIO[1], (Senior Member, IEEE), AND GEORGIOS CH. SIRAKOULIS[3], (Member, IEEE)

[1]Department of Electronics Engineering, Universitat Politécnica de Catalunya (UPC), 08034 Barcelona, Spain
[2]Department of Electrical and Computer Engineering, Technische Universität Dresden, 01062 Dresden, Germany
[3]Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece
[4]Institute of Nanoscience and Nanotechnology, National Center for Scientific Research "Demokritos" (NCSR), 15341 Athens, Greece

Corresponding author: Vasileios Ntinas (vasileios.ntinas@upc.edu)

This work was supported in part by the Hellenic Foundation for Research and Innovation (HFRI) under the "First Call for HFRI Research Projects to support Faculty Members and Researchers and the Procurement of High-Cost Research Equipment Grant" under Project 3830, in part by UPC through FPI-UPC 2018 Grant, and in part by the Spanish AEI–Agencia Estatal de Investigación under Grant PID2019-103869RB-C33/AEI/10.13039/501100011033. The publication of the article in OA mode was financially supported by HEAL-Link.

ABSTRACT Inspired by the behavior of natural systems, Cellular Automata (CA) tackle the demanding long-distance information transfer of conventional computers by the massive parallel computation performed by a set of locally-coupled dynamical nodes. Although CA are envisioned as powerful deterministic computers, their intrinsic capabilities are expanded after the memristor's probabilistic switching is introduced into CA cells, resulting in new hybrid deterministic and probabilistic memristor-based CA (MemCA). In the proposed MemCA hardware realization, memristor devices are incorporated in both the cell and rule modules, composing the very first all-memristor CA hardware, designed with mixed CMOS/Memristor circuits. The proposed implementation accomplishes high operating speed and reduced area requirements, exploiting also memristor as an entropy source in every CA cell. MemCA's functioning is showcased in deterministic and probabilistic operation, which can be externally modified by the selection of programming voltage amplitude, without changing the design. Also, the proposed MemCA system includes a reconfigurable rule module implementation that allows for spatial and temporal rule inhomogeneity.

INDEX TERMS Memristor technology, cellular automata, emergent computing hardware, hybrid CMOS/memristor design.

I. INTRODUCTION

Traditional general-purpose *processor-centric* computing systems, i.e. von Neumann computers, cannot meet the modern demands for energy- and area-efficient implementation of computationally heavy tasks in compact autonomous devices, known as Edge Computing nodes. Mainly due to the high demands for information transfer between processor and memory, computation-intensive operations, e.g. demanding Artificial Intelligence (AI) tasks, do not fit well into energy- and area-constrained embedded hardware. Therefore, unconventional computing architectures should be investigated in order to achieve efficient AI *computing on the Edge*.

One of the most well-known unconventional computing architectures is Cellular Automata (CA) [1], which is a computing paradigm that is based on the interaction of simple building elements, namely CA cells. Such a decentralized computing system, which is a prime example of Emergent

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

Computing [1], [2], is capable of performing complex computational tasks owing to cells' collective behavior. Since processing is conducted locally in each CA cell, this computing paradigm lacks a focal computing module, such as the foreseen Central Processing Unit (CPU) in von Neumann computers. Furthermore, information storage is spread throughout the CA cells, forming an inherent co-locality of storage and processing, addressing the massive burden of information transmission required in classical computers.

The research on CA dates back to the 1940s, when von Neumann worked on the notion of self-reproducing automata and, together with Stanislaw Ulam, who was researching lattice networks, created an early model of today's CA [3]. Von Neumann's inspiration was drawn from nature, as he sought to demystify the computing capability of biological systems, like the human brain, by comparing their complexity to that of then-modern computers. Conway et al. [4] demonstrated the creation of patterns and self-organization inside a two-dimensional array of binary CA cells, garnering widespread recognition for CA. Stephen Wolfram subsequently created the contemporary mathematical foundation of CA based on his careful analysis of CA's complexity [5]. Coming from the realm of nonlinear systems, Leon Chua conducted an in-depth analysis of Wolfram's CA, concentrating on the particular situation of Elementary CA (ECA). In his work [6], Chua quantified the complexity of the ECA by establishing the index of complexity, created analytical formulations of the CA cells' interaction rules, and investigated symmetries between the various ECA rules. In addition, Chua introduced the idea of Cellular Nonlinear (or Neural) Networks (CNNs), an extended form of CA that functions in continuous time [7].

In the previous two decades, a wide range of CA-based applications have been explored and implemented, either in software or hardware. CA are appealing for the modeling of complex natural, biological, physical, or chemical systems because of their simple formulation since they can readily imitate the local interactions of such systems while effectively demonstrating global phenomena. CA modeling of chemical processes [8], for example, has been found to improve the understanding of chemical systems such as the Belousov-Zhabotinsky reaction [9], [10], [11], [12]. Furthermore, their applications in biological system modeling range from excitable and oscillating mediums that mimic brain events like nerve cell excitation to DNA sequencing [13], [14]. CA's potential for simulating biological systems such as plant succession and wildfire spread has also been examined [15], [16]. CA-based models, on the other hand, have been used in the modeling of semiconductors by realizing predictive models of their production processes [17]. Moreover, the dynamics of social systems during emergency situations, like urgent evacuation or pandemics spreading, have been successfully captured by CA models [18], [19]. Non-deterministic variants of CA have also been used to model complex problems such as stock market dynamics, virus spreading, swarm dynamics, neuropercolation, and self-organization in complex systems [20], [21], [22], [23], [24].

In terms of purely computational tasks, the local processing of spatial information in a parallel manner enhances CA's suitability for image processing tasks such as noise filtering, edge detection, and image sharpening, to name a few [25], [26], [27]. Furthermore, the temporal evolution of CA has been used in the creation of cryptosystems based on either simple or complex CA cells capable of generating strongly encrypted sequences [28], [29], [30].

Nonetheless, the implementation of CA in traditional computers is inefficient owing to the CPU's serial execution of operations, which restricts the execution speed of CA applications since the cells' activities are completed sequentially. However, the simplicity of the CA system's formulation has piqued the interest of hardware developers, who can simply convert CA to digital circuit designs and implement them with powerful Field-Programmable Gate Arrays (FPGAs), which are ideal for the hardware realization of CA's parallelism and module-based structure [31]. As a result, a great variety of CA adaptations have been shown on FPGAs in order to accomplish ultra-fast CA applications. FPGA-based CA has been utilized to improve the efficiency of software-based equivalents in real-time crowd evacuation [32], traffic management [33], wildfire spreading modeling [34], backtracking of DNA sequence evolution [35], modeling of microbes [36], and chemical processes [10], while, recently, CA's dynamical behavior has been exploited in Reservoir Computing applications for energy-efficient pattern recognition [37]. Although FPGAs have clearly dominated the field of CA hardware implementations, this sector lacks almost any alternative hardware system, such as application-specific integrated circuits (ASICs). The high prototyping cost and lack of adaptability of ASIC-based CA implementations seem to be prohibitive to their widespread use, whereas a generic CA ASIC design remains undiscovered [38]. Despite this, the efficient design of ASIC-based CA may yield faster and more compact solutions in terms of hardware area. It should be highlighted that the digital nature of FPGAs precludes the creation of non-deterministic CA variants, which call for vast pseudo-randomness generation, making it almost impossible for digital systems.

From a technological standpoint, the driving power of classical computer growth is the feasibility of transistor technology getting minimized, which enabled the skyrocketing of conventional complementary metal-oxide-semiconductor (CMOS) technology. However, reaching the physical limits of transistor miniaturization saturates the potential of conventional technology to further improve its energy and area requirements, which opens space for other novel semiconductor technologies to be explored. Among these lines, memristor technology gains ground over purely CMOS approaches as it enables ultra-dense non-volatile information storage near the processing unit, as well as the low-power implementation of artificial neural network (ANN) accelerators in hardware.

Since purely memristor-based circuits lack the necessary cascadability to achieve the realization of very-large-scale integration (VLSI) systems, hybrid CMOS/memristor technology is becoming more attractive as it exploits memristor's low-power, high-density, and non-volatility, along with CMOS technology's high speed and design scalability.

Considering dedicated hardware for CA applications, Itoh and Chua recommended using a memristor in a basic circuit to achieve the functioning of a CA cell and then building an array of such cells to make a CA grid [39]. In this technique, they used the intrinsic dynamics of idealized memristor devices to accomplish various logic operations that implement the essential cell interaction to solve computing tasks, demonstrating a wide range of image processing applications. Although this pioneering work provided the inspiration for numerous memristor-based CA systems, this technique is practically infeasible owing to the arbitrary selection of the memristor's nonlinear dynamics for each application, which does not correspond to the behavior of actual memristor devices. They did, however, describe certain characteristics for memristor-based CA, such as the correct separation of time-steps into read and write phases, which were used by subsequent memristor-based CA systems [40].

Thus, after the publication of Itoh and Chua's landmark article, other memristor-based CA methodologies for the execution of computing tasks, such as shortest route finding methods and sorting systems [41], have been presented. In addition, memristor-based CA has been suggested for image processing in medical applications [42], pseudo-randomness generation [43], the simulation of the propagation of epileptic brain activity [44], and the development of patterns via simulating the Game of Life [45]. Moreover, a unique strategy for edge detection using memristor-based CA has been developed by including the adaptation of CA's neighborhood [46]. Aside from their early uses, memristor-based CA lack a general architecture that would allow a broad range of individuals to utilize this computational tool. In addition, the integration of memristors in CA implementations conceals an unusual feature that permits the hardware-level realization of nondeterministic CA [47].

In the context of this work, the realization of efficient CA hardware for both deterministic and probabilistic tasks is investigated with the utilization of hybrid CMOS/Memristor circuits. As a computational tool, CA can be exploited in logical computations, since they have proven their computational capabilities as Turing machines [48], but also in the generation of probabilistic sequences, useful as generators of truly random numbers as well as necessary for stochastic computing systems. The transistor-level design of an all-memristor CA implementation is proposed, utilizing memristors in both state and rule implementation, an approach that is proven to be efficient for non-CA systems like memristor-based deep neural networks [49]. A memristor in each cell holds the cell state and allows its probabilistic transition, while a memristive crossbar array implements the CA rule for the cell. This approach allows for reprogramming the system to any ECA rule and enables the implementation of CA algorithms where versatile selection and alternation of the rule are necessary. In addition, the proposed design provides control of the transition probability by simply tuning the amplitude of the memristor programming voltage, which is a globally controlled signal and may be adjusted throughout the operation of the system.

The rest of the paper is structured as follows: In Section II, the necessary mathematical background on Elementary CA, Probabilistic CA and Memristor-based CA is provided. The novel hybrid CMOS/Memristor design of the all-memristor CA is detailed in Section III, while Section IV demonstrates MemCA's operation for both deterministic and probabilistic tasks. Finally, Section V concludes the paper.

II. BACKGROUND MATERIAL

Given $N \in \mathbb{N}$ cells, a CA is defined as the regular spatial structure of interconnected cells. Such an arrangement constitutes the *CA grid* and may be n-dimensional; however, 1-, 2- or 3-dimensional typologies are mostly utilized. Each CA cell C_i stores the internal state variable s_i, where $i \in [1, N]$, which evolves over time owing to the local interaction between cells. In the simplest CA implementations, s_i is a single binary digit or a number in a higher radix, but in more complex CA, it might represent a state variable tuple.

The dynamics of s_i are described by a discrete-time mathematical expression, known as the *CA rule*, that involves the cell's state s_i as well as the states s_j of the neighboring cells, where $j \in K_{C,i}$. The neighborhood $K_{C,i}$ includes the cells within the interaction range r of each cell, which is commonly selected as unity $r = 1$ to reduce the computational demands and, most significantly, ease the hardware implementation requirements. For higher dimensions $n \geq 2$, the exact neighboring cells are differently defined by the choice of the distance function; for example, using Manhattan distance in a two-dimensional grid leads to the so-called von Neumann (4-cell) neighborhood, while using Chebyshev distance leads to the Moore (8-cell) neighborhood. The CA rule is commonly shared among the cells, known as the uniform rule, but in more complicated applications it may vary, leading to a non-uniform CA rule within the grid. During the neighborhood's definition, the boundary conditions of the grid should also be defined since the edge cells are missing some physical neighbors. This missing neighbor can be considered a constant-valued entity (fixed boundaries) or, most commonly, is replaced by the cells at the grid's other edges that also have missing neighbors (periodic or cyclic boundaries), forming a closed space.

Considering the above CA features and their variety, the design space for CA implementations is vast, and their applications are much more diverse. Elementary CA (ECA) is one of the most studied and materialized CA due to its simplicity, compatibility with digital systems, and Turing completeness. In this section, the mathematical foundation of ECA is

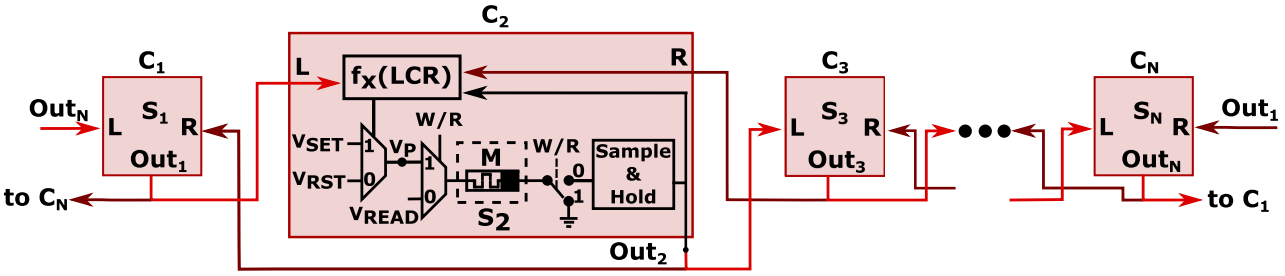


FIGURE 1. One-dimensional CA grid. The block diagram of the Memristor-based CA cell is elaborated within the second cell C_2 to abstractly visualize the circuit realization of the cell.

presented along with their generalization to Probabilistic CA, where the transition rules become probabilistic, and the special case of memristor-based CA (MemCA), where the probabilistic rules are governed by the memristor's probabilistic transitions.

A. ELEMENTARY CELLULAR AUTOMATA

ECA are defined as 1-D arrays of binary state cells, $s_i \in \{0, 1\}$, which interact only with the adjacent cells in $r = 1$, such as $K_{C,i} = \{C_j|_{|i-j|\leq 1}\}$, so the CA rule $f : \{0, 1\}^3 \rightarrow \{0, 1\}$ is defined as

$$s'_i = f(s_{i-1}, s_i, s_{i+1}) \tag{1}$$

where s'_i is the state of C_i at the next time-step. Therefore, the rule expression constitutes a Boolean function of the current state of the left neighbor L (s_{i-1}), the central cell itself C (s_i), and the right neighbor R (s_{i+1}), denoted as LCR, and its outcome defines the future state of the cell s'_i.

Stephen Wolfram proposed a mnemonic notation [50] for ECA's rules, where each rule is uniquely named after the truth table of f. Defining x_u as the outcome of $f(u)$ for $u = \text{LCR}_\text{D}$,[1] an 8-bit number ($x_\text{B} = \text{x}_7\text{x}_6\text{x}_5\text{x}_4\text{x}_3\text{x}_2\text{x}_1\text{x}_0$) represents the outcome for the eight input combinations. So, its decimal representation ($x_\text{D} = \sum_{u=0}^{7}(x_u 2^u)$) names the ECA rule. For example, the outcome set $x = 00011110_\text{B}$ is mapped to the ECA rule f_x for $x = 30_\text{D}$, as shown in TABLE 1. Moreover, each ECA rule f_x can be mapped to a digital circuit with only AND (Boolean multiplication) and OR (Boolean addition) operators using the Karnaugh map method.

B. PROBABILISTIC CELLULAR AUTOMATA

Once the deterministic transitions that are governed by the ECA rules are generalized to random transitions, probabilistic variants of ECA arise. In such variants, known as Probabilistic Cellular Automata (PCA) or Stochastic Cellular Automata, the transition rule describes all the conditional probabilities for a cell to be at a specific state s_x at the next time-step given every potential current state combination LCR. The rule function is defined as $\text{pf}^{s_x} : \{0, 1\}^3 \rightarrow \mathbb{R}_{\in[0,1]}$, where

$$\text{pf}^{s_x}(\text{LCR}) = P(s'_i = s_x|\text{LCR}). \tag{2}$$

According to the PCA notation proposed in [47], the probabilistic rule is simply defined as the conditional probability of a LCR combination resulting in the binary state $s_x = 1$, such as

$$\text{pf}(\text{LCR}) = P(s'_i = 1 \mid \text{LCR}). \tag{3}$$

Using the same notation as in ECA, now the outcome of each input combination is a probability such as $\text{pf}(u) = P(s'_i = 1 \mid u) = p_u$, where $p_u \in [0, 1]_{\in\mathbb{R}}$. However, due to the non-integer nature of p_u, it is impossible to define each probabilistic rule in a compact form as in the ECA case but rather as pf_pk, with $\text{pk} = \text{p}_7\text{p}_6\text{p}_5\text{p}_4\text{p}_3\text{p}_2\text{p}_1\text{p}_0$, shown in TABLE 1. A special case of the probabilistic rule is when any p_u is equal to 0 or 1 which corresponds to a deterministic transition, because the chances for $s'_i = 1$ are 0% or 100%, respectively. If all the probability coefficients are zero or one ($p_u \in \{0, 1\}\ \forall\ u$), the probabilistic rule degenerates into a deterministic ECA rule.

TABLE 1. ECA rules in Wolfram's notation and their probabilistic and memristor-based generalizations.

$u = \text{LCR}_\text{D}$	7	6	5	4	3	2	1	0
Wolfram notation value	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
LCR_B	111	110	101	100	011	010	001	000
Elementary Cellular Automata Rule								
f_x (LCR)	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
f_{30} (LCR)	0	0	0	1	1	1	1	0
Probabilistic Cellular Automata Rule								
pf_pk (LCR)	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0
Memristor-based Cellular Automata Rule								
$\text{pf}_{77,\text{MemCA}}$ (LCR)	p_R	1	0	0	1	1	0	p_S
f_{76} (LCR)	0	1	0	0	1	1	0	0
f_{77} (LCR)	0	1	0	0	1	1	0	1
f_{204} (LCR)	1	1	0	0	1	1	0	0
f_{205} (LCR)	1	1	0	0	1	1	0	1

$x_l \in \{0, 1\}$, $p_l \triangleq P(f(\text{LCR}) = 1|\text{LCR}) \in [0, 1]_{\in\mathbb{R}}$
$p_{R/S}$: Memristor's Reset/Set probability

C. MEMRISTOR-BASED CELLULAR AUTOMATA

Utilizing its resistive state, a memristor device can effectively store the binary cell state of an ECA cell. Therefore, a memristor-based ECA cell can be materialized by a memristor M for the cell state, a logic block for the rule, and auxiliary circuitry for the memristor's reprogramming, as illustrated in

[1]The subscript of a number defines its radix, i.e. B for binary and D for decimal number.

FIGURE 1. The rule block f_x gathers the signals L, C, and R, which carry the state information of the corresponding cell, and delivers the outcome of f_x to the control circuitry for memristor programming. For $f_x(\text{LCR}) = 1$ ($f_x(\text{LCR}) = 0$), memristor is programmed to the low (high) resistance state LRS (HRS) by selecting the applied voltage as $V_\text{P} = V_\text{SET}$ ($V_\text{P} = V_\text{RST}$).

Since the cell state is kept on a single memristor, reading and programming cannot be carried out in parallel. Hence, each CA time-step is divided into the reading phase and the programming phase, termed the writing phase. The signal W/R manages the transition between these phases. Firstly, W/R $= 0$ denotes the reading phase, where the memristor state is read by a small-amplitude pulse V_READ and temporarily stored in the *Sample & Hold* module. Following that, the memristor is grounded for the writing, and the voltage V_P, as determined by f_x, is applied to it.

Additionally, the MemCA cell can be exploited to execute ECA rules probabilistically by tailoring the duration PW and the amplitude V_P of memristor programming pulses. Stemming from the stochastic nature of memristor's switching dynamics [51], a voltage-controlled probabilistic switch can be implemented by utilizing memristor's switching time, which may be expressed as a Poisson process, such as

$$P_{SW}(t) = \frac{\Delta t}{\tau}\, e^{-t/\tau}. \tag{4}$$

In (4), Δt represents an infinitely small time interval and τ is the amplitude-dependent memristor's characteristic SET/RESET switching time, which is expressed as

$$\tau(V) = \tau_0 \exp(V/V_0) \tag{5}$$

where τ_0 and V_0 are fitting parameters acquired by measurements on fabricated devices [52]. Given the voltage-dependent switching probability, memristor devices may be employed as statistically adjustable circuit elements.

The mathematical formulation of MemCA rules, a subset of the generic PCA rules, has been proposed in [47] in light of the memristor's probabilistic switching as the required source of randomness inside each CA cell. In particular, probabilistic state transitions in MemCA are only observed when the rule demands a state transition, i.e. $s_i' \neq s_i$, since the memristor's non-volatility does not allow unwanted cell states when there is no state transition, i.e. $s_i' = s_i$. The deterministic rule f_{77}, for example, demands state transition in two cases

(a) $f_{77}(000) = 1$
(b) $f_{77}(111) = 0$

else $s_i' = s_i$. Given the SET and RESET probabilities p_S and p_R, respectively, the MemCA probabilistic version of f_{77} is represented as $\text{pf}_{77,\text{MemCA}} = \text{pf}_{\text{p}_R100110\text{p}_S}$ (TABLE 1).

III. MemCA CIRCUIT DESIGN

CA are often built using FPGA-based digital circuits, utilizing the well-established digital design tools, in contrast to memristor-based ICs, which are in their infancy and lack standard design procedures. However, local non-volatile information storage and the integration of an entropy source, implemented by digital randomness generators, in each CA cell necessitate the development of complicated digital circuits per cell, which is unfavorable to system scalability. Given that MemCA provides these features, its efficient hardware implementation is essential for the realization of MemCA processors that facilitate the formation of efficient emergent computing approaches.

Despite the fact that memristor-based crossbar array designs have already been presented [53], [54], [55], mixed circuits with individual memristor and transistor devices for ASICs need ad hoc design. Since MemCA transistor-level design does not constitute a trivial task, in this section, the transistor-level design of the proposed MemCA hardware realization is presented. The CA cells, where the cell state memristor is placed and programmed by the appropriate programming circuitry, and the CA Rule module, which executes the rule's logical operations at each time-step, are the core elements of the MemCA array. An all-memristor design strategy is devised in this innovative approach, with memristors employed in both CA cell and CA rule modules. Memristor introduces the essential stochasticity in CA cells to execute PCA tasks by utilizing memristor switching probability that is manipulated by the amplitude of the externally controlled programming pulse. Furthermore, the memristor-based crossbar array is employed in the CA rule module to allow rule's reprogramming *on-the-fly*, enabling the implementation of CA algorithms.

A. MemCA CELL

Following the description of the MemCA cell in Section II, for the proposed hybrid CMOS/Memristor design, the cell has been built with the following five blocks or elements: (a) programming and reading control module, (b) memristor device, (c) a control module for the alternation of reading and writing phases, (d) a sample-and-hold (S&H) module, and (e) a sensing module. The MemCA cell's block diagram is shown in FIGURE 2(a), which also depicts the external signals required for the cell's functioning, while the transistor-level design of each module of the CA cell is shown in FIGURE 2(b). The external control signals are distributed uniformly to all the cells, reducing the design complexity of the whole CA. These global signals, illustrated within the green area in FIGURE 2, are divided into the phase-related ones (*wave-like* lines), which control the transition between the time-step phases, and the constant global ones (flat lines) that manage the programming and reading voltages. An explanation of how each block operates follows.

V_applied *Control:* This module determines the voltage applied to the memristor at each CA time-step phase based on the CA rule's outcome. The external signal V_IN results from the CA rule module and manages the polarity of the memristor programming voltage. Additionally, the inverted write-enable $\overline{W_\text{EN}}$ and read-enable $\overline{R_\text{EN}}$ signals are used for the transition between the writing and reading

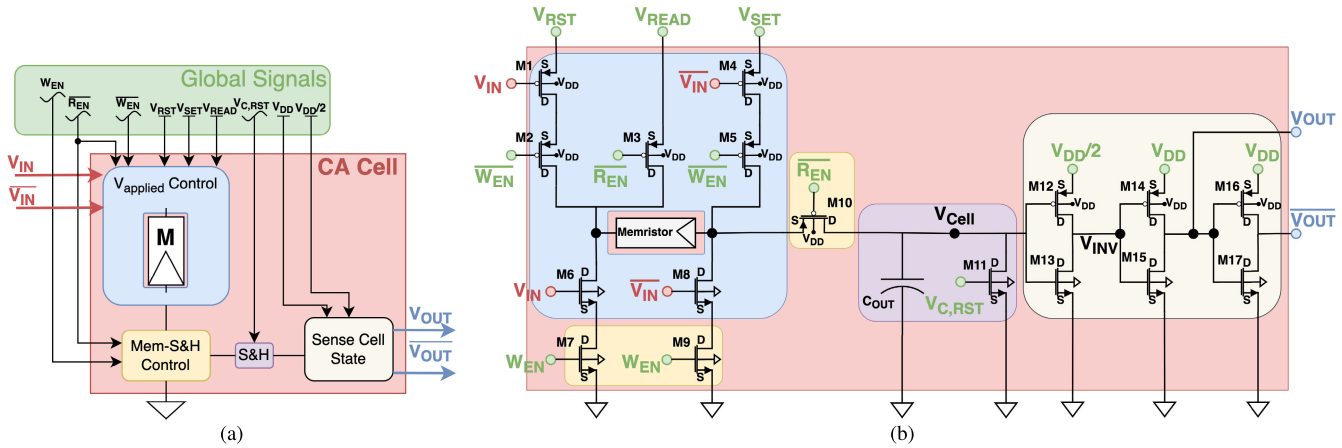


FIGURE 2. MemCA cell implementation. (a) Block diagram of the MemCA cell design where all the external local and global signals are also shown. (b) The transistor-level design of the hybrid CMOS/Memristor MemCA cell realization.

phases. Throughout the latter, the read voltage V_{READ} is provided to the memristor. In particular, the V_{applied} Control module consists of a number of PMOS and NMOS transistors that control the applied voltages on the memristor by alternating the programming path based on the CA rule's outcome, for writing, or applying V_{READ}, for reading.

Memristor: It is the core element of the proposed CA cell with a two-fold operation: (I) stores CA cell state in a non-volatile manner, and (II) introduces randomness in the MemCA cells.

Mem-S&H Control: It functions as a single-pole double-throw (SPDT) switch. It is composed of a pair of NMOS transistors (M7, M9) that ground the memristor for the writing phase and a PMOS transistor (M10) that connects the memristor to the S&H module for the reading phase. $\overline{R_{\mathrm{EN}}}$ and W_{EN} are utilized to manage the timing of the reading and writing phases.

S&H: Analog sample-and-hold element that gets charged during the reading phase and maintains its charge in the course of the writing phase while memristor is reprogrammed to the next state. This module is comprised of a capacitor-NMOS (C_{OUT}-M11) pair, where the CA cell state is temporarily stored. During reading, the NMOS transistor M11 is appropriately biased to modulate the charging time of the capacitor, but during writing, M11 is biased to cut-off to prevent the discharge of C_{OUT}. Moreover, at the end of the writing phase, M11 is biased to be fully conductive for the instant discharge of C_{OUT} before the reading phase of the next CA time-step. The sampling time is globally controlled by the signal $V_{\mathrm{C,RST}}$.

Sense Cell State: This module continuously monitors the value stored in S&H and shifts it to the logic levels required by the CA rule module. It is made up of three CMOS inverters, the first two of which serve as voltage level amplifiers since the amplitude of V_{READ} is low, and the third of which supplies the complementary cell output required by the CA rule module.

To illustrate the operation of all these CA cell module sub-circuits, all the phases of the CA time-step are described in chronological sequence, thus, first, reading, then writing, and lastly clearing, which is a brief sub-phase of writing.

1) READING PHASE

Memristor is connected to S&H, and a reading pulse with low voltage amplitude V_{READ} is applied through the V_{applied} Control circuit. The write-enable W_{EN} signal is LOW so M2, M5, M7, and M9 are in the cut-off region, detaching the programming voltages V_{SET} and V_{RST}. Therefore, the bias of M1, M4, M6, and M8 is insignificant. On the other hand, the PMOS transistors M3 and M10 are conductive, as the inverted read-enable $\overline{R_{\mathrm{EN}}}$ signal is LOW, and, in combination with the sub-threshold biasing of M11 ($V_{\mathrm{C,RST}} = V_{\mathrm{VD}}$), a conductive path that charges C_{OUT} through memristor is created. The selection of V_{VD} tunes M11's resistance (R_{OUT}) during the reading phase and manages the S&H module's time constant and the V_{Cell}'s charging amplitude. Based on the current memristor's state LRS or HRS, C_{OUT} is charged to a high or low V_{Cell} level, respectively. The reading phase conductive path is shown in FIGURE 3(a), where each transistor is illustrated by the corresponding open or closed switch, depending on its reading phase bias. The insignificant transistors for this phase are depicted by the *don't care (X)* symbol, while the potentiometer symbol indicates the adjustable output resistance.

2) WRITING PHASE

The memristor is grounded and receives the proper programming voltage from the V_{applied} Control circuit, while S&H holds the charged V_{Cell} level that controls CA cell's output and should be constant during the writing phase. In specific, $\overline{R_{\mathrm{EN}}}$ is HIGH, biasing M10 to the cut-off, while W_{EN} is HIGH, so the NMOS transistors M7 and M9 are fully conductive to allow the grounding of the memristor. Here is where the outcome of the CA rule module, driven

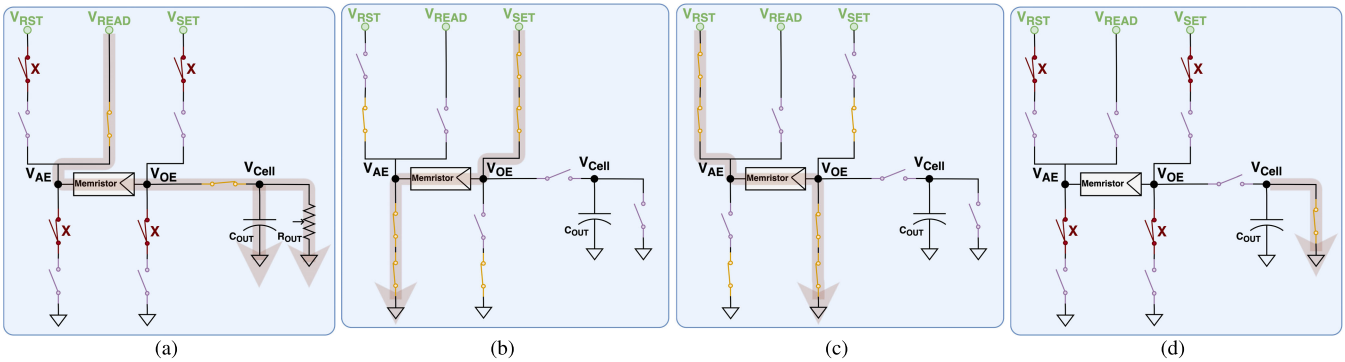


FIGURE 3. CA cell's operational part during (a) the reading phase, the writing phase of a CA time-step when V_{IN} is at (b) HIGH and (c) LOW voltage level, and (d) the clearing phase of a CA time-step.

into the CA cell by V_{IN} and $\overline{V_{IN}}$, plays an important role. When V_{IN} is HIGH (LOW), M1 and M8 (M4 and M6) are in the cut-off region, so the only conductive path is from V_{SET} (V_{RST}) to ground through M4-M5-memristor-M6-M7 (M1-M2-memristor-M8-M9), as shown in Figs. 3(b) and (c). In this manner, the programming of the memristor is accomplished based on the polarity of the voltage across the memristor with no need for negative applied voltage within the CA cell module. Also, $V_{C,RST}$ biases M11 to the cut-off region during the writing phase to prevent the C_{OUT}'s discharge.

3) CLEARING PHASE

Special consideration has to be given at the very end of the writing phase, which is dedicated to the complete discharging, i.e. *clearing*, of C_{OUT}. This is necessary to avoid any unwanted pre-charging of C_{OUT} from a prior time-step affects the charging level of V_{Cell} during the reading phase. In specific, $V_{C,RST}$ biases M11 to the conductive state for a very short time and V_{Cell} drops to zero. During this phase, all the other timing-dependent transistors of the CA cell are biased at the cut off region to prevent any undesired influence on the memristor since the CA rule module is directly affected by the change of V_{Cell}.

Importantly, the actual circuit topology and programming circuitry differ slightly from the abstract visualization in FIGURE 1. The special directional memristor programming circuit is proposed in this paper to avoid the use of negative voltages, which are necessary for the programming of bipolar memristors where SET and RESET are only possible with opposite polarities. In specific, incorporating negative voltage in CMOS-based design imposes extra limitations as the ground connection, used in digital circuits, should be replaced by the negative supply voltage, and, necessarily, the operational voltage span would be doubled. Since memristors need higher programming amplitude than the operational voltage of CMOS technology, duplicating the voltage span that also covers memristor programming amplitudes is prohibitive for modern power- and area-efficient transistor technologies with small feature sizes. As a result, the use of 180 nm or larger transistor technology for the mixed CMOS/Memristor circuits would be necessary, imposing frequency, power, and area limitations in the whole CA cell design. Thus, despite the slight increase in total transistor number per CA cell for the proposed directional memristor programming, the lack of negative programming voltage permits the use of 32 nm transistor technology, achieving ultra-fast operating speed that is not harmed by the transistor technology specification, as well as area efficiency and low power consumption.

In a conventional CMOS design, a CA cell should be designed as a 1-bit register to store cell state, which commonly means a single D flip-flop composed of 18 transistors in its simpler form. Comparing the proposed hybrid CMOS/Memristor CA cell with the purely CMOS conventional approach, the MemCA cell achieves a similar transistor count (17 transistors and a capacitor) while providing local non-volatile storage of cell state, which is impossible for its pure-CMOS counterpart.

B. MemCA RULE

In the proposed MemCA implementation, memristor-based crossbar arrays are employed for the realization of the CA rule module due to their reprogramming ability, non-volatile storage, and compact area footprint. Memristor crossbar is not only used to store the Boolean function of the CA rule but also to execute it, with the help of a read-out circuit, and provides the next time-step cell state to the CA cell module.

In particular, this unique memristor-based topology, in which memristors are arranged at the intersections of horizontal (rows) and vertical (columns) perpendicular nanowires, may be used for logical operations such as AND and OR logic gates. When voltages are applied to the rows (inputs), the current in each column is controlled by the conductance of the memristors at each crosspoint. Thus, by appropriately programming the conductance of each memristor, the column current (output) may perform arithmetic and logic operations.

Because of its tremendous efficiency in performing massively parallel matrix-vector multiplications and simulating synaptic connections in the brain, this topology is frequently employed in memristor-based neural network accelerators

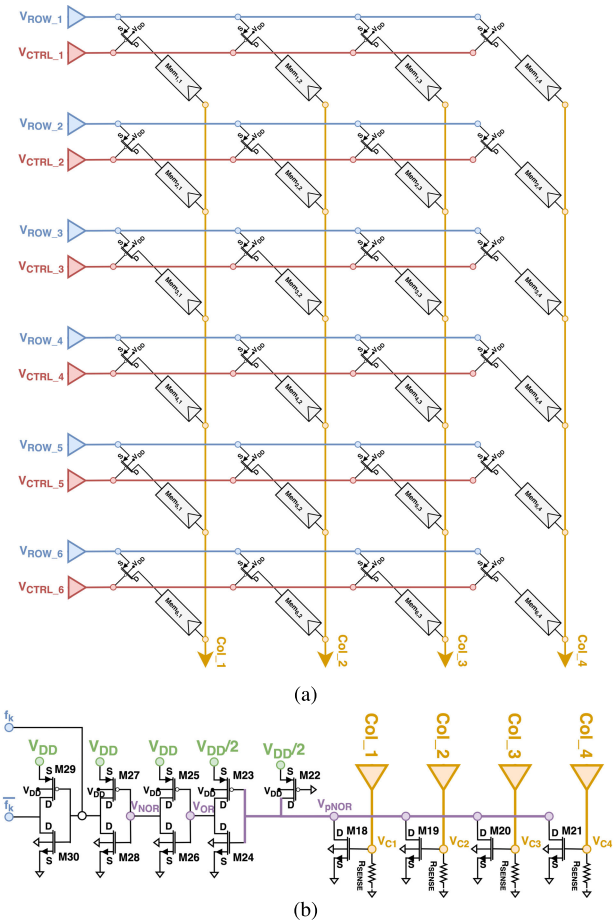


FIGURE 4. CA Rule module. (a) Memristor-based crossbar array and (b) read-out circuit for the implementation of any CA rule.

and neuromorphic computing today. However, multi-level memristor programming and/or continuous reprogramming are required in such systems. Both are considered memristor properties that are being developed and improved since the variability of memristors is harmful to precise programming and constant reprogramming of memristor conductance. Albeit, since the rule f_x in ECA is a predetermined Boolean expression, no crossbar array reprogramming or multi-level memristor programming is required.

Realizing any ECA rule f_x with a memristor crossbar, a small 6×4 crossbar is required for each CA cell, which is designed as illustrated in FIGURE 4(a). By appropriately setting each memristor to a high or low resistance state (HRS, LRS), each column may execute a single AND logic operation, while a read-out circuit senses column current and executes the necessary OR logic operation for all the columns. The rows operate as the f_x inputs, so six rows are necessary for the three cell state signals L, C, R and for their three complements $\overline{L}$, $\overline{C}$, $\overline{R}$, which drive the PMOS selectors, one per row, in order to enable only the selected crosspoints. On the other hand, the number of columns is defined by the number of OR operands in f_x, which can be up to 8 for an unoptimized three-input Boolean function, as there are 8 different input combinations. However, since any three-input Boolean expression can be optimized by the Karnaugh map method to have a maximum of four OR operands, a four-column crossbar is sufficient for implementing any ECA rule f_x; nevertheless, in their majority, fewer columns are required, so the memristors of the extra columns are programmed to HRS to prevent affecting the computation.

In regards to the way that ECA rules are implemented inside the crossbar, each crosspoint is made up of a PMOS transistor and a memristor (1T1R), where the memristor is used as a reprogrammable resistor. Each crossbar row nanowire (Row_i) is connected directly to the PMOS selectors of the row, where each selector is in series with a memristor and the selector's gate terminal is connected to a control line (CTRL_i). The other terminal of the memristor is connected to the column nanowire (Col_j) that accumulates the column's current. In this configuration, when PMOS is conductive, the row voltage drops on the memristor, leading to a current weighted by the memristor's state. When the memristor is in LRS (HRS), there is a high (low) contribution to the column's current. In the proposed CA rule implementation, a fixed voltage is applied to row nanowires (Row_{1-6}) and each selector line to a single input voltage. In this way, the CA cell outputs are isolated as they drive transistor gates and they are not affected by the resistance state of crossbar memristors.

More specifically, the necessary crossbar memristors must be programmed at the LRS for the execution of AND operations in a crossbar column so that when all of the required inputs are in logic '1', sufficient current will flow in the column. As long as PMOS selectors invert the gate signal, as they are conductive at low gate voltage, the memristor programming should also be inverted. Therefore, to implement any AND gate within the crossbar that needs a logical '1' at any input signal, the memristor at the crosspoint of this input signal should be programmed at HRS, while the memristor at the inverted of this input should be programmed at LRS. In addition, special consideration is needed when an input signal is not required at a specific AND operation, so memristors at both the input signal and its inverse should be programmed at LRS so at least one of them will provide the necessary current flow contribution to the column. This special rule is required to maintain the same current levels regardless of the number of AND gate inputs.

Given the proper crossbar memristor programming, the current at a Col_j will be adequate only when the designed AND gate yields to a logical '1'. All of the crossbar columns may be configured to perform various AND operations. The Rule Read-Out circuit performs column current sensing along with the OR operation on all columns. This circuit's role is to output the result of a designed ECA rule f_x and feed it to the CA cell module. More particularly, a sense resistor R_{SENSE} receives each column's current, and its voltage V_{Ci} drives the gate of a NMOS transistor, as shown in FIGURE 4(b). The drain terminals of the four sense transistors M18-M21 are linked together, forming pseudo-NOR gate with the PMOS transistor M22, which is fixed to the conductive state.

Pseudo-NOR's result is then reversed by a CMOS inverter, yielding the OR operation of the four columns. Because the amplitude on R_{SENSE} is low, it is insufficient to drive the pseudo-NOR to full supply amplitude; therefore, only half amplitude ($V_{DD}/2$) is supplied to the pseudo-NOR and the subsequent inverter. The full supply range is restored by two following full-amplitude inverters, while one more inverter is added to produce the complement of the rule's outcome, which is also needed in the CA cell module.

In comparison, a pure-CMOS implementation of a fully reprogrammable CA rule realization would require an 8:1 multiplexer that consists of more than 100 transistors and, still, cannot store the CA rule locally. So, the proposed MemCA rule design achieves both a lower transistor count and the non-volatile storing of the rule. Remarkably, in the proposed MemCA implementation, the probabilistic transition of each CA cell state is performed within the CA cell by properly selecting V_{SET} and V_{RST}; thus, the rule design remains unchanged for either deterministic or probabilistic CA operations.

C. SIMULATION MODELS AND DESIGN PARAMETERS

The proposed MemCA implementation is a general design technique that is capable of incorporating several kinds of bipolar memristor devices and transistor technologies. In order to accurately simulate the behavior of actual memristors in the MemCA circuit, the Jülich Aachen resistive switching tools (JART) valance change mechanism (VCM) device model is used [56], while the NMOS and PMOS transistors are modeled using the BSIM v4 transistor model with the parameter set for 32 nm transistor technology from the Predictive Transistor Model (PTM) [57].

The JART VCM memristor model captures the behavior of ReRAM devices in which state switching is dependent on valance change mechanisms. In these devices, the switching of the memristor's conductivity is evident when the applied signal induces an electric field within the switching layer that leads to the movement of the oxygen vacancies. The interface of the switching layer with the electrode forms a Schottky barrier that becomes lower as the oxygen vacancy concentration near the interface increases. When a negative voltage is provided to the device, SET is performed, attracting oxygen vacancies at the interface, lowering the Schottky barrier, and decreasing the resistance of that region. During RESET, a positive voltage at the device repels oxygen vacancies from the interface's vicinity, causing their concentration to drop and the Schottky barrier to rise.

The key processes of VCM-based devices have been investigated for a considerable amount of time [58], [59], [60], [61], since they have been observed in a broad variety of memristor device materials, like HfO_x, TiO_x, TaO_x, ZrO_x [58], [62]. Therefore, the JART VCM memristor model can properly describe the dynamics and conduction processes of diverse memristor devices. In addition, in [56], the inclusion of device variability into the memristor model is achieved by appropriately changing specific physical parameters inspired by the switching behavior observed in real devices. Noteworthy, to truly reproduce the gradual stochastic fluctuation of physical entities during the operation of fabricated memristor devices, the modification of the model parameters in [56] that arises from the cycle-to-cycle variability of the devices is performed during simulation by an algorithmic technique based on random walks. More specifically, the change of each variability-related parameter of the model in the course of the simulation is emulated by a *random walker* that performs small steps within the predefined parameter bounds in a randomly selected direction, i.e. positive or negative change.

TABLE 2. Parameter range for variability-aware JART VCM memristors.

Parameter	Min	Mean	Max
N_{min} (10^{23}m^{-3})	4	8	25
N_{max} (10^{26}m^{-3})	0.39	0.4	0.41
r (nm)	40.5	45	49.5
l_d (nm)	0.36	0.4	0.44

Utilizing the JART VCM memristor model and 32 nm transistors, the CA cell and CA rule circuits are developed properly to achieve rapid memristor programming and CA cell operation. It should be noted that the switching speed of a memristor is greatly dependent on the amplitude of the programming pulse, with a minor increase resulting in orders of magnitude quicker switching. However, since small-sized transistor technologies restrict the maximum voltage amplitude within the circuit, the switching speed of the memristor is limited. In the proposed MemCA design, a completely functional circuit is developed with a CA time-step of 100 ns.

During simulation, the JART VCM Verilog-A implementation is used, as provided in [63], with the proposed parameters fitting the behavior of a Pt/Ti/TiO_x/HfO_2/Pt device. The random walk approach was used to determine the variability of each individual CA cell memristor, so multiple parameter sets were produced using the parameter ranges specified in [56] (cf. TABLE 2). Considering MemCA's simple requirements for the memristor crossbar operation, i.e. binary memristor programming and lack of reprogramming during simulation, we are not considering here the peripheral circuitry for the programming of crossbar memristors, which is well-studied in the literature [53]. Also, for the simulations, the CA rule module memristors are initialized to a maximum or minimum resistance state based on the needs of the desired ECA rule, but no variability is taken into account since they are only programmed once at the start of system operation and are not subjected to switching.

Taking into account the memristor's SET and RESET voltages and resistance values LRS and HRS, the CA cell is properly built in such a way that the maximum available V_{SET} and V_{RST} values allow for deterministic switching of the memristor device in the desired time-frame. In addition, a key goal of the proposed design is the ability to perform

TABLE 3. Channel W_{Mi}/L_{Mi} ratio for the transistors of the MemCA design.

M1	M2	M3	M4	M5	M6	M7	M8
60	60	20	8	8	3	3	25
M9	M10	M11	M12	M13	M14	M15	M16
25	20	15	1	4	1	4	1
M17	M18	M19	M20	M21	M22	M23	M24
4	6	6	6	6	3	1	10
M25	M26	M27	M28	M29	M30	M_{xbar}	
1	20	1	2	1	12	15	

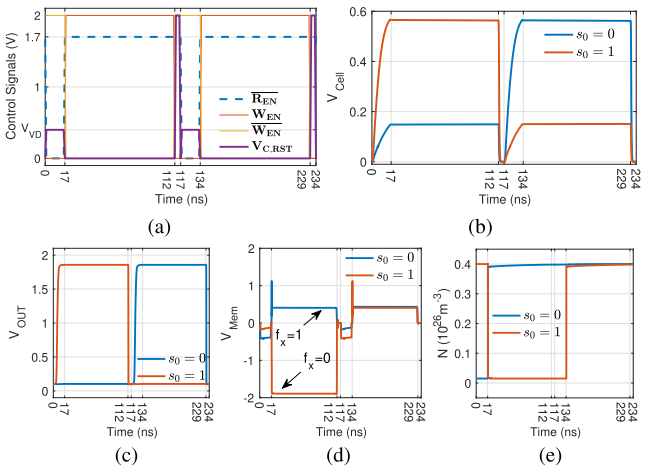


FIGURE 5. Single CA cell operation for $s_0 = 0$ and $s_0 = 1$. (a) Control signals. (b) Internal voltage in the S&H module V_{Cell}. (c) Output signal of the CA cell V_{OUT}. (d) Voltage dropped on the memristor and (e) memristor internal state variable.

probabilistic switching with no design changes other than the selection of V_{SET} and V_{RST}.

Delving into the design parameters, the power supply is set to V_{DD} =2 V to ensure that all signals at HIGH voltage level are equal to 2 V and at LOW voltage level are equal to 0 V. The $\overline{R_{EN}}$ signal, which controls transistor M10, is given extra care with a special $V_{high} = 0.85V_{DD}$, as neither of its terminals is connected to supply or ground. Also, $V_{DD}/2$ is chosen for the reading pulse, while C_{OUT} =714 fF and V_{VD} =0.4 V are selected to obtain a low S&H's time constant, achieving a short reading phase duration, i.e. τ_{read} =17 ns. The length of the writing phase is set at τ_{write} =100 ns to provide sufficient time for memristor switching, and the last t_{clear} =5 ns of the writing pulse are utilized to discharge V_{Cell}.

Regarding the CA rule module, each row is connected to a fixed voltage $V_{row} = V_{DD}/2$, while the sense resistances are selected as R_{SENSE} =1.4 kΩ. The length of each transistor channel is set throughout the design at the minimum node size of the technology, thus L_{Mi} =32 nm. On the contrary, the transistor channel's width is carefully determined for each individual transistor, resulting in different W_{Mi}/L_{Mi} ratios shown in TABLE 3. Due to the considerably higher current that runs through the memristor in LRS, the transistors in the CA cell's RESET path (M1, M2, M8, M9) must be substantially larger than those in the SET path (M4, M5, M6, M7).

TABLE 4. Memristor array programming to implement ECA rule f_{30}.

	V_{sel}	Col_1	Col_2	Col_3	Col_4
Row_1	$\overline{L}$	HRS	HRS	LRS	HRS
Row_2	L	LRS	LRS	HRS	HRS
Row_3	$\overline{C}$	LRS	LRS	HRS	HRS
Row_4	C	LRS	HRS	LRS	HRS
Row_5	$\overline{R}$	LRS	LRS	HRS	HRS
Row_6	R	HRS	LRS	LRS	HRS

IV. MemCA OPERATION

This section demonstrates both the deterministic and probabilistic operations of the proposed MemCA implementation. To begin with, a single cell starting from two distinct initial states $s_0 = \{0, 1\}$ is showcased in FIGURE 5 to illustrate the global control signals, the internal cell voltages, and the memristor's state during two CA time-steps. Moreover, to highlight the memristor's programming within the proposed cell, the two possible rule outcomes $f_x = \{1, 0\}$ are also depicted. By selecting the maximum programming pulse amplitudes ($V_{SET} = V_{RST} = V_{DD}$), the rapid switching of the memristor's state is obvious in FIGURE 5(e), ensuring deterministic CA operation.

Beyond the single cell, a CA rule module is required to implement the desired rule for each CA cell module in order to build a complete CA array. The interconnection between CA cell modules and the CA rule modules for three adjacent cells is depicted in FIGURE 6. The global signals are shared among the CA array elements for the synchronization of the massive parallel cells' operation. On the other hand, the CA cell's outputs are only locally distributed to the CA rule modules of the neighbors, keeping the information processing purely local.

A. DETERMINISTIC OPERATION

The rule f_{30} is implemented to exemplify the deterministic functioning of the whole CA array. So, the memristors of each crossbar are properly programmed to execute rule 30 through the following optimized Boolean function

$$\begin{aligned} f_{30,xbar} &= f_{30,Col_1} + f_{30,Col_2} + f_{30,Col_3} + f_{30,Col_4} \\ &= \overline{L} \cdot R + \overline{L} \cdot C + L \cdot \overline{C} \cdot \overline{R} + 0 \end{aligned} \tag{6}$$

As long as the optimized Boolean function of rule 30 is comprised of three non-zero terms, memristor devices from only three crossbar columns need to be selectively programmed to LRS; hence, (6) is realized by programming the crossbar array shown in FIGURE 4(a) according to TABLE 4.

An exemplary CA grid, with $N = 8$ and the rule 30 in deterministic mode, is simulated to verify the system's proper operation. The cells are initialized to logical '0', i.e. CA cell module's memristor starts at HRS, apart from a single cell that starts from logical '1', i.e. its memristor begins at LRS, in order to trigger the CA array's evolution. The boundary conditions are periodic. FIGURE 7(a) depicts the time evolution of the CA rule module for each cell $f_{30,i}$ (left y-axis) and the memristor's state in each cell (right y-axis), while

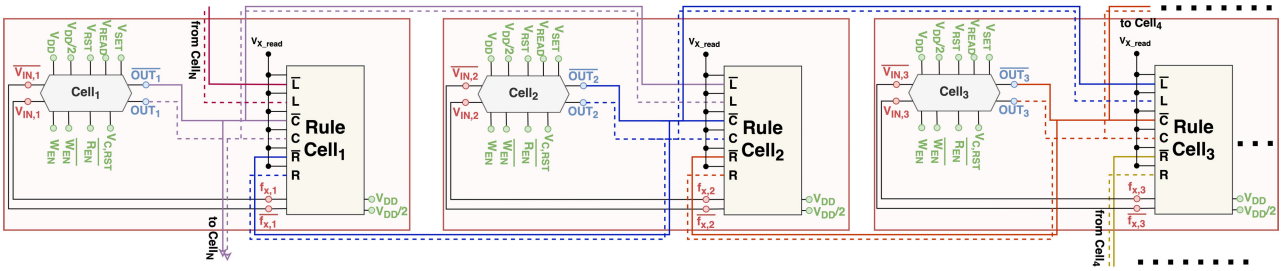


FIGURE 6. Block diagram of CA array's realization with the proposed memristor-based modules. Three consecutive cells are presented and the connectivity of the CA rule module signals is highlighted.

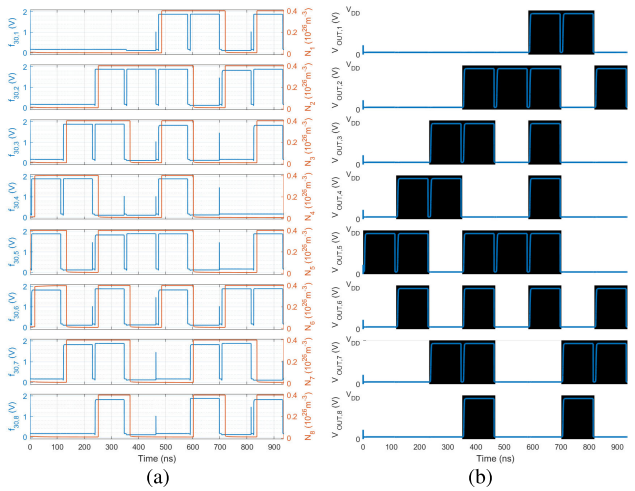


FIGURE 7. Time evolution of the (a) CA rule module, memristor's state and (b) output voltage for each CA cell for ECA rule f_{30}.

FIGURE 7(b) shows the time evolution of each CA cell's output voltage $V_{\mathrm{OUT,i}}$.

B. PROBABILISTIC OPERATION

In the context of the probabilistic MemCA, the choice of lower programming voltages results in the probabilistic switching of memristor devices in CA cells. To elaborate on the probabilistic operation, the effect of the memristor's probabilistic programming is first investigated on a single cell, while the spatiotemporal behavior of a MemCA structure with all cells in probabilistic mode is later studied.

1) SINGLE MemCA CELL

In an effort to stress the switching rate of a single CA cell, the rule f_{51} is selected, which is implemented by a single NOT logic gate and, therefore, always results in a cell state change. Rule f_{51} could serve as an ideal random number generator whether both the memristor's SET and RESET probabilities are equal to 50%. Unfortunately, such a condition is not feasible for manufactured memristors mostly because their device-to-device variability impacts the exact $\{V_{\mathrm{SET}}, V_{\mathrm{RST}}\}$ combination required for each memristor device to attain a 50% switching probability. Additionally, the RESET switching time greatly relies on the resistance state of the memristor at the start of programming. Therefore, the slower RESET

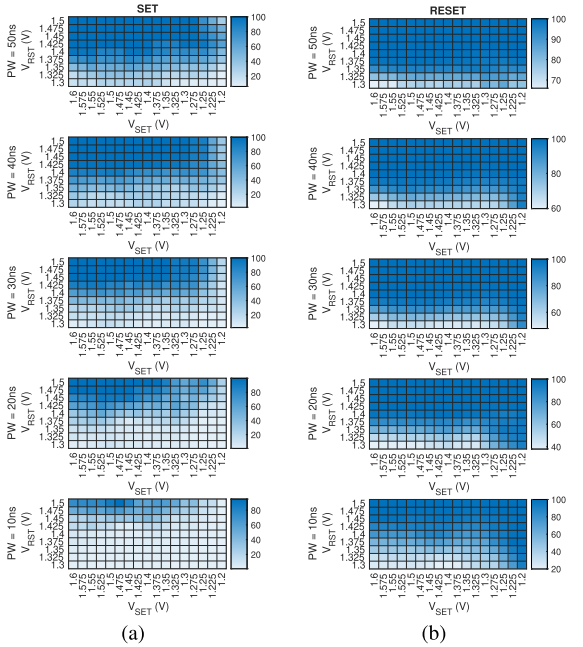


FIGURE 8. Single cell (a) SET and (b) RESET probability for various V_{SET} and V_{RST} voltages and programming pulse time PW.

speed of a memristor when it begins with a lower LRS value results in a multi-factorial control of switching, so the RESET probability is not only determined by V_{RST} in practical applications.

To analyze MemCA cell's probabilistic state transitions in the proposed implementation, an individual cell with the rule f_{51} is simulated for various $\{V_{\mathrm{SET}}, V_{\mathrm{RST}}\}$ combinations. Additionally, the influence of the programming pulse's duration on the MemCA cell's transition probability is examined. Such is plausible with this novel design by modulating the time of the clearing phase (τ_{clear}), which is part of the writing phase. Because the memristor is separated from any applied voltage during the clearing phase, the programming pulse length may be modulated by manipulating the external global signals without changing the circuitry. As a benchmark, a single CA cell is simulated for 400 CA time-steps to calculate the SET and RESET switching probabilities of the whole cell. V_{SET} and V_{RST} were set to different values in the range [1.2, 1.6]V and [1.3, 1.5]V, respectively, and the programming pulse width was set to five distinct values,

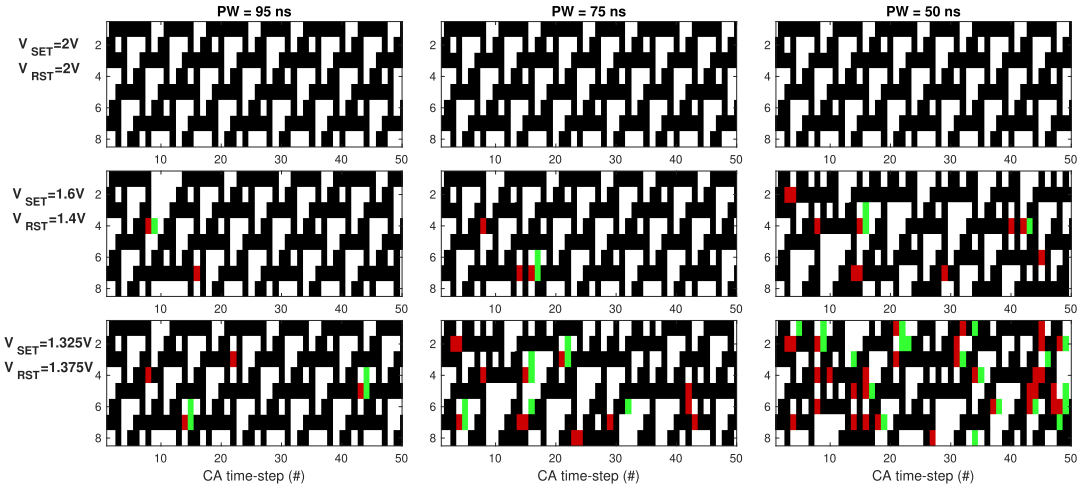


FIGURE 9. CA evolution diagram for the 8-cell MemCA with rule f_{110} for PW = {95, 75, 50}ns and three {V_{SET}, V_{RST}} pairs, i.e. {2, 2}V, {1.6, 1.4}V and {1.325, 1.375}V. In each CA evolution diagram, the y-axis represents each cell of the 8-cell array and the x-axis corresponds to discrete CA time-steps. Each MemCA cell output $V_{OUT,i}$ is low ($s_{i,t} = 0$) for the white cells and $V_{OUT,i}$ is high ($s_{i,t} = 1$) for the black ones, while, the dark red and green cells represents the RESET and SET transition failure of the cell at that CA time-step, respectively.

i.e. PW = {50, 40, 30, 20, 10}ns. Figures 8(a) and (b) show the probability of switching for the SET and the RESET processes, respectively. A 2D data smoother was used to eliminate the effect of outliers in the visualization of switching probabilities with the heatmap and better highlight the average effect of PW. As shown, the switching probability for SET or RESET is not reliant only on V_{SET} or V_{RST}, respectively, but on the combination of $\{V_{SET}, V_{RST}\}$, as the exact LRS value reached after a V_{SET} pulse influences the subsequent RESET process.

2) MULTI-CELL MemCA ARRAY

An 8-bit MemCA is simulated to analyze the spatiotemporal characteristics of the probabilistic operation. A different rule is chosen to display the versatility of the proposed CA rule module. Particularly, the CA rule modules are set to the rule f_{110}, which was investigated from a theoretical standpoint in [47]. The optimized Boolean function for rule 110 is the following

$$\begin{aligned} f_{110,\text{xbar}} &= f_{110,\text{Col}_1} + f_{110,\text{Col}_2} + f_{110,\text{Col}_3} + f_{110,\text{Col}_4} \\ &= \overline{L} \cdot R + \overline{C} \cdot R + C \cdot \overline{R} + 0 \end{aligned} \tag{7}$$

which is realized by programming the crossbar array shown in FIGURE 4(a) according to TABLE 5.

In this example, as a single non-zero cell does not trigger any rich CA evolution for this rule, the CA array is initiated in the randomly selected initial state {0, 1, 1, 0, 0, 0, 1, 0}. Each simulation performed 200 CA time-steps, where both the cycle-to-cycle and the device-to-device variability of memristors are included. Moreover, aiming to study the influence of applied voltage on CA's evolution, three distinct $\{V_{SET}, V_{RST}\}$ combinations were simulated, i.e. $\{V_{SET}, V_{RST}\}$ = {2, 2}V, {1.6, 1.4}V, {1.325, 1.375}V. The first combination results in deterministic switching, whereas

TABLE 5. Memristor array programming to implement ECA rule f_{110}.

	V_{sel}	Col_1	Col_2	Col_3	Col_4
Row_1	$\overline{L}$	HRS	LRS	LRS	HRS
Row_2	L	LRS	LRS	LRS	HRS
Row_3	$\overline{C}$	LRS	HRS	LRS	HRS
Row_4	C	LRS	LRS	HRS	HRS
Row_5	$\overline{R}$	LRS	LRS	HRS	HRS
Row_6	R	HRS	HRS	LRS	HRS

the other two result in arbitrarily chosen probabilistic switching with different switching probabilities. Because of the larger amplitude of programming voltages, the combination $\{V_{SET}, V_{RST}\}$ = {1.6, 1.4}V is chosen to investigate low probability of switching failures, whereas the combination $\{V_{SET}, V_{RST}\}$ = {1.325, 1.375}V is chosen for high probability of switching failures. Furthermore, the effect of programming pulse duration is explored by simulating each $\{V_{SET}, V_{RST}\}$ combination for three different pulse widths, namely PW = {95, 75, 50}ns. Because memristor switching processes are heavily reliant on applied voltage amplitude, the influence of pulse length is anticipated to be less pronounced than the effect of pulse amplitude.

The temporal evolution of MemCA hardware, in discrete CA time-steps, is depicted for each of the 9 $\{V_{SET}, V_{RST}, PW\}$ simulated combinations in FIGURE 9. Only 50 first CA time-steps are displayed for readability, with low $V_{OUT,i}$ (s_i = 0) in white and high $V_{OUT,i}$ (s_i = 1) in black. Moreover, unsuccessful CA cell state transitions are characterized as transition failures and are colored differently in FIGURE 9; i.e., RESET transition failures are represented by dark red cells, whereas SET transition failures are represented by green cells.

MemCA time-evolution when $\{V_{SET}, V_{RST}\}$ = {2, 2}V follows the deterministic ECA evolution owing to the

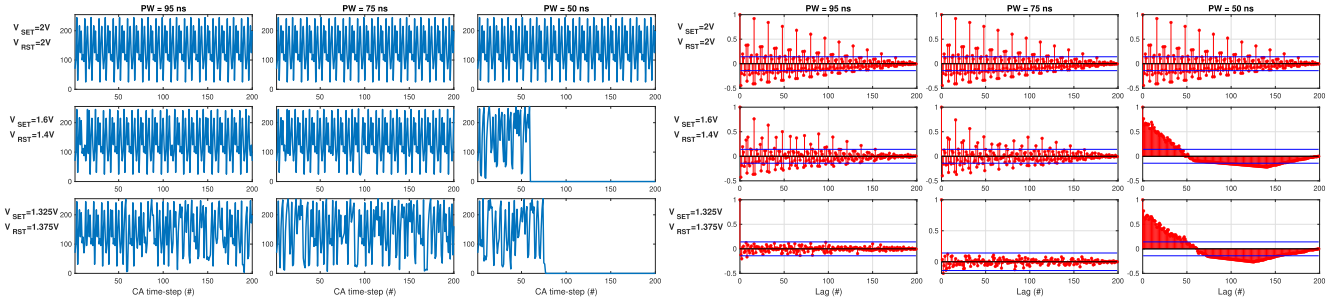


FIGURE 10. (a) Generated number sequence for each example in FIGURE 9. The 8-cell MemCA array is mapped to an 8-bit integer number and the corresponding time-series for 200 CA time-steps is illustrated. (b) Auto-correlation function of the CA time-series presented in (a).

assured memristor's switching for high-amplitude programming pulses, even when the pulse width is lowered (PW = $\{75ns, 50ns\}$), as illustrated in the first row of FIGURE 9. If lower programming voltages are used, transition failures occur, such as in the second and third rows of FIGURE 9. For both SET and RESET transitions, the {1.325, 1.375}V combination yields more frequent transition failures than the {1.6, 1.4}V combination, as predicted. Also, whenever the length of the programming pulse is shortened, the frequency of transition failures increases in each of the aforementioned two programming voltage combinations.

The generated number sequence of the 8-cell MemCA example also reveals the influence of the programming pulse amplitude and duration. The 8-cell MemCA array produces an 8-bit value in each CA time-step, since the MemCA cell state is binary, so the generated number sequence is the time-series of the integer representation of this 8-bit value. Figure 10(a) visualizes the generated number sequence for all the nine simulated $\{V_{\text{SET}}, V_{\text{RST}}, \text{PW}\}$ combinations for all the simulation time, i.e. 200 CA time-steps. Given that there are no transition failures for the deterministic mode, the time series for the highest programming voltage selection displays a repeated pattern that satisfies the deterministic ECA rule f_{110}. The other two voltage settings, on the other hand, cause transition failures that distort the time-series. The {1.6, 1.4}V case, in particular, has a few transition failures that cause minor deviations from the deterministic pattern, but the lowest programming pulse case exhibits numerous transition failures that regularly modify the evolution pattern, resulting in a less predictable time-series.

Analyzing now the decreasing programming duration, the two lowest programming pulse pairs exhibit a less predictable pattern because transition failure occurrence rises with decreasing PW. Nevertheless, CA's state transition ceases to exist for the lowest PW as the lack of proper CA rule evolution leads to certain global CA states that do not yield state transition, leaving the whole CA grid stuck at this state. Each CA rule may show its unique *stuck-at* states that result from the local rule, the initial state, and the CA array size. Due to its multi-parametric nature, it is hard to forecast such states via theoretical investigations. For instance, f_{30} has a stuck-at global state only when the CA array size is even [64].

In the presence of transition failure, CA's time-series may deviate from the deterministic pattern, which is visible in MemCA for lower and/or shorter programming pulses. These deviations may lead to normally inaccessible global states from the deterministic pattern, which results in more visited global states and a higher time-series' entropy, which is beneficial for random sequence generation. Conversely, it permits the undesired emergence of *stuck-at* global states.

3) EVALUATION OF MemCA TIME-SERIES PREDICTABILITY

Using the auto-correlation function (ACF), the generated pattern's predictability is assessed. ACF, in particular, calculates the correlation between a signal and its delayed duplicate. The formula for the auto-correlation coefficient is

$$r_q = \frac{1}{T} \frac{\sum_{t=1}^{T-q} (y_t - \bar{y})\,(y_{t+k} - \bar{y})}{\text{Var}(y)} \tag{8}$$

where q is the *lag*, which denotes the number of time-steps delay for the delayed time-series, T stands for the total number of time-steps, y_t is the value of the time-series at the t_{th} time-step, and $\bar{y}$ and $\text{Var}(y)$ are the mean value and the variance of the time-series, respectively.

In order to portray the periodic occurrence of repeating patterns in the time-series, the ACF of each simulated CA time-series is calculated. For the simulated time-series, the lag may be any integer number between 0 to 199 and $T = 200$, due to the 200 simulated CA time-steps. The autocorrelation coefficient is always maximal for zero lag, i.e. $r_0 = 1$, whereas $r_q \in (-1, 1) \forall q > 0$. When r_q is outside of the confidence interval, a statistically significant correlation is detected. To attain a 95% confidence level, the confidence interval is chosen as twice the standard error (SE), which is expressed as $SE = 1/\sqrt{T}$. The calculated ACF for the nine $\{V_{\text{SET}}, V_{\text{RST}}, \text{PW}\}$ combinations is shown in FIGURE 10(b) with the 95% confidence level bounds denoted by the blue horizontal lines.

In the highest programming amplitude cases, the ACF demonstrates high r_q for a certain q, which corresponds to the period of a repetitive pattern. It is obvious that many r_q values exceed the confidence interval bounds, meaning that there is a significant correlation at these lags. For the {1.6, 1.4}V scenario, the amplitude of the high r_q values is decreased due

to the small deviations in the time-series that were induced by the transition failures, however, there are still many r_q out of the confidence interval, in both PW = 95ns and PW = 75ns cases. On the other hand, for the lowest amplitude pair, the higher occurrence frequency of transition failures led to a much more unpredictable time-series, which is observed in the third row of FIGURE 10(b) by the much lower r_q value that lies within or very near to the confidence interval. Interestingly, the ACF of the {1.325V, 1.375V, 75ns} combination shows that the generated time-series is strongly uncorrelated, with only one coefficient, i.e. r_2, out, but very near, of the confidence interval. Furthermore, the two cases where the CA grid arrived at a *stuck-at* global state, show a strong correlation as the time-series becomes constant.

V. CONCLUSION

Cellular Automata hardware is currently severely constrained, owing to the unavailability of universal CA realization that can provide reprogrammable CA for any rule and neighborhood setup. Employing memristors in CA hardware implementations, on the other hand, has been found to improve CA dynamics and enable fast CA applications. Despite the fact that deterministic approaches of memristor-based CA implementations yield high-speed, low-power, and dense-area implementations, probabilistic memristor-based CA variations have been poorly investigated yet.

Among these lines, an all-memristor CA design with a detailed transistor-level hybrid CMOS/memristor approach is proposed, using memristors in both the implementation of the CA cell and the CA rule circuits. Compared to conventional CMOS approaches, MemCA achieves similar or lower transistor counts while offering local non-volatile storing of both cell states and CA rules. Such an innovative approach provides high versatility in the selection of the ECA rule and the memristor's switching probability, which could form a general CA hardware. The operation of the MemCA approach is shown in both a deterministic and probabilistic manner, illustrating the fast speed of the system and the ability to increase CA's unpredictability by considering the memristor as the necessary entropy source. The proposed hardware realization allows the implementation of CA algorithms that can provide efficient solutions to real-time applications like cryptography, stochastic computing and reservoir computing. Consequently, the proposed hybrid CMOS/Memristor design permits the modification of both the CA rule, even locally inside each cell, and the transition probability, resulting in a generic framework for implementing ECA and PCA at the nanoscale level.

REFERENCES

[1] S. Wolfram, "Cellular automata as models of complexity," *Nature*, vol. 311, pp. 419–424, Oct. 1984.

[2] K. A. Brunner, *What's Emergent in Emergent Computing?* Princeton, NJ, USA: Citeseer, 2002.

[3] J. Neumann, *Theory of Self-Reproducing Automata*. Champaign, IL, USA: Univ. Illinois Press, 1966.

[4] J. Conway, "The game of life," *Sci. Amer.*, vol. 223, no. 4, p. 4, 1970.

[5] S. Wolfram, *A New Kind of Science*, vol. 5. Champaign, IL, USA: Wolfram Media, 2002.

[6] L. O. Chua, *A Nonlinear Dynamics Perspective of Wolfram's New Kind of Science*, vol. 4. Singapore: World Scientific, 2011.

[7] L. O. Chua, *CNN: A Paradigm for Complexity*, vol. 31. Singapore: World Scientific, 1998.

[8] L. B. Kier, P. G. Seybold, and C.-K. Cheng, *Modeling Chemical Systems Using Cellular Automata*. Cham, Switzerland: Springer, 2005.

[9] A. Turner, *A Simple Model of the Belousov–Zhabotinsky Reaction From First Principles*. London, U.K.: Bartlett School of Graduate Studies, 2009.

[10] N. I. Dourvas, G. C. Sirakoulis, and A. Adamatzky, "Cellular automaton Belousov–Zhabotinsky model for binary full adder," *Int. J. Bifurcation Chaos*, vol. 27, no. 6, Jun. 2017, Art. no. 1750089.

[11] M.-A. Tsompanas, I.-A. Fyrigos, V. Ntinas, A. Adamatzky, and G. C. Sirakoulis, "Light sensitive Belousov–Zhabotinsky medium accommodates multiple logic gates," *Biosystems*, vol. 206, Aug. 2021, Art. no. 104447.

[12] M.-A. Tsompanas, I.-A. Fyrigos, V. Ntinas, A. Adamatzky, and G. C. Sirakoulis, "Cellular automata implementation of oregonator simulating light-sensitive Belousov–Zhabotinsky medium," *Nonlinear Dyn.*, vol. 104, no. 4, pp. 4103–4115, Jun. 2021.

[13] G. B. Ermentrout and L. Edelstein-Keshet, "Cellular automata approaches to biological modeling," *J. Theor. Biol.*, vol. 160, no. 1, pp. 97–133, Jan. 1993.

[14] G. C. Sirakoulis, "Parallel application of hybrid DNA cellular automata for pseudorandom number generation," *J. Cellular Automata*, vol. 11, no. 1, pp. 1–15, 2016.

[15] P. Hogeweg, "Cellular automata as a paradigm for ecological modeling," *Appl. Math. Comput.*, vol. 27, no. 1, pp. 81–100, Jul. 1988.

[16] I. Karafyllidis and A. Thanailakis, "A model for predicting forest fire spreading using cellular automata," *Ecol. Model.*, vol. 99, no. 1, pp. 87–97, Jun. 1997.

[17] G. C. Sirakoulis, "A TCAD system for VLSI implementation of the CVD process using VHDL," *Integration*, vol. 37, no. 1, pp. 63–81, Feb. 2004.

[18] N. Pelechano and A. Malkawi, "Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches," *Autom. Constr.*, vol. 17, no. 4, pp. 377–385, May 2008.

[19] G. C. Sirakoulis, I. Karafyllidis, and A. Thanailakis, "A cellular automaton model for the effects of population movement and vaccination on epidemic propagation," *Ecol. Model.*, vol. 133, no. 3, pp. 209–223, Sep. 2000.

[20] M. Bartolozzi and A. W. Thomas, "Stochastic cellular automata model for stock market dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 4, Apr. 2004, Art. no. 046112.

[21] A. M. D. Rey, "A computer virus spread model based on cellular automata on graphs," in *Proc. Int. Work-Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2009, pp. 503–506.

[22] D. A. Lima and G. M. B. Oliveira, "A probabilistic cellular automata ant memory model for a swarm of foraging robots," in *Proc. 14th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2016, pp. 1–6.

[23] R. Kozma, M. Puljic, P. Balister, B. Bollobas, and W. J. Freeman, "Neuropercolation: A random cellular automata approach to spatio-temporal neurodynamics," in *Proc. Int. Conf. Cellular Automata*. Cham, Switzerland: Springer, 2004, pp. 435–443.

[24] D. Griffeath, "Self-organization of random cellular automata: Four snapshots," in *Probability and Phase Transition*. Cham, Switzerland: Springer, 1994, pp. 49–67.

[25] G. Hernández and H. J. Herrmann, "Cellular automata for elementary image enhancement," *Graph. Models Image Process.*, vol. 58, no. 1, pp. 82–89, Jan. 1996.

[26] P. L. Rosin, "Image processing using 3-state cellular automata," *Comput. Vis. Image Understand.*, vol. 114, pp. 790–802, Jul. 2010.

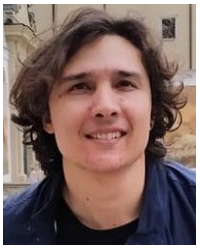
[27] P. L. Rosin, A. Adamatzky, and X. Sun, *Cellular Automata in Image Processing and Geometry*. Cham, Switzerland: Springer, 2014.

[28] S. Wolfram, "Cryptography with cellular automata," in *Proc. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1985, pp. 429–432.

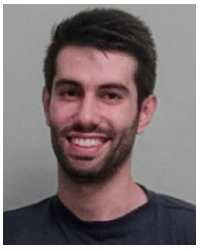
[29] S. Nandi, B. K. Kar, and P. P. Chaudhuri, "Theory and applications of cellular automata in cryptography," *IEEE Trans. Comput.*, vol. 43, no. 12, pp. 1346–1357, Dec. 1994.

[30] F. Seredynski, P. Bouvry, and A. Y. Zomaya, "Cellular automata computations and secret key cryptography," *Parallel Comput.*, vol. 30, nos. 5–6, pp. 753–766, May 2004.

[31] M. Halbach and R. Hoffmann, "Implementing cellular automata in FPGA logic," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, 2004, p. 258.
[32] A. Tsiftsis, I. G. Georgoudas, and G. C. Sirakoulis, "Real data evaluation of a crowd supervising system for stadium evacuation and its hardware implementation," *IEEE Syst. J.*, vol. 10, no. 2, pp. 649–660, Jun. 2016.
[33] G. Kalogeropoulos, G. C. Sirakoulis, and I. Karafyllidis, "Cellular automata on FPGA for real-time urban traffic signals control," *J. Supercomput.*, vol. 65, no. 2, pp. 664–681, Aug. 2013.
[34] V. G. Ntinas, B. E. Moutafis, G. A. Trunfio, and G. C. Sirakoulis, "Parallel fuzzy cellular automata for data-driven simulation of wildfire spreading," *J. Comput. Sci.*, vol. 21, pp. 469–485, Jul. 2017.
[35] G. C. Sirakoulis, I. Karafyllidis, C. Mizas, V. Mardiris, A. Thanailakis, and P. Tsalides, "A cellular automaton model for the study of DNA sequence evolution," *Comput. Biol. Med.*, vol. 33, no. 5, pp. 439–453, Sep. 2003.
[36] N. Dourvas, M.-A. Tsompanas, G. C. Sirakoulis, and P. Tsalides, "Hardware acceleration of cellular automata physarum polycephalum model," *Parallel Process. Lett.*, vol. 25, no. 1, Mar. 2015, Art. no. 1540006.
[37] A. Moran, C. F. Frasser, M. Roca, and J. L. Rossello, "Energy-efficient pattern recognition hardware with elementary cellular automata," *IEEE Trans. Comput.*, vol. 69, no. 3, pp. 392–401, Mar. 2020.
[38] D. Monica, "Cellular automata hardware implementations-an overview," *Sci. Technol.*, vol. 19, no. 4, pp. 360–368, 2016.
[39] M. Itoh and L. O. Chua, "Memristor cellular automata and memristor discrete-time cellular neural networks," *Int. J. Bifurcation Chaos*, vol. 19, no. 11, pp. 3605–3656, Nov. 2009.
[40] V. Ntinas, R.-E. Karamani, I.-A. Fyrigos, N. Vasileiadis, D. Stathis, I. Vourkas, P. Dimitrakis, I. Karafyllidis, and G. C. Sirakoulis, "Cellular automata coupled with memristor devices: A fine unconventional computing paradigm," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2020, pp. 1–4.
[41] I. Vourkas, D. Stathis, and G. C. Sirakoulis, "Memristor-based parallel sorting approach using one-dimensional cellular automata," *Electron. Lett.*, vol. 50, no. 24, pp. 1819–1821, 2014.
[42] J. Secco, M. Farina, D. Demarchi, F. Corinto, and M. Gilli, "Memristor cellular automata for image pattern recognition and clinical applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1378–1381.
[43] R.-E. Karamani, V. Ntinas, I. Vourkas, and G. C. Sirakoulis, "1-D memristor-based cellular automaton for pseudo-random number generation," in *Proc. 27th Int. Symp. Power Timing Modeling, Optim. Simulation (PATMOS)*, Sep. 2017, pp. 1–6.
[44] R.-E. Karamani, I.-A. Fyrigos, V. Ntinas, I. Vourkas, G. C. Sirakoulis, and A. Rubio, "Memristive cellular automata for modeling of epileptic brain activity," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–5.
[45] R.-E. Karamani, I.-A. Fyrigos, V. Ntinas, I. Vourkas, and G. C. Sirakoulis, "Game of life in memristor cellular automata grid," in *Proc. 16th Int. Workshop Cellular Nanoscale Netw. Appl.*, 2018, pp. 1–4.
[46] R.-E. Karamani, I.-A. Fyrigos, K.-A. Tsakalos, V. Ntinas, M.-A. Tsompanas, and G. C. Sirakoulis, "Memristive learning cellular automata for edge detection," *Chaos, Solitons Fractals*, vol. 145, Apr. 2021, Art. no. 110700.
[47] V. Ntinas, G. C. Sirakoulis, and A. Rubio, "Memristor-based probabilistic cellular automata," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2021, pp. 792–795.
[48] M. Cook, "Universality in elementary cellular automata," *Complex Syst.*, vol. 15, no. 1, pp. 1–40, 2004.
[49] P. Wijesinghe, A. Ankit, A. Sengupta, and K. Roy, "An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 345–358, Oct. 2018.
[50] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Phys.*, vol. 55, no. 3, p. 601, 1983.
[51] S. H. Jo, K.-H. Kim, and W. Lu, "Programmable resistance switching in nanoscale two-terminal devices," *Nano Lett.*, vol. 9, no. 1, pp. 496–500, 2009.
[52] R. Naous, A. Siemon, M. Schulten, H. Alahmadi, A. Kindsmüller, M. Lübben, A. Heittmann, R. Waser, K. N. Salama, and S. Menzel, "Theory and experimental verification of configurable computing with stochastic memristors," *Sci. Rep.*, vol. 11, no. 1, pp. 1–11, Feb. 2021.
[53] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano Lett.*, vol. 12, no. 1, pp. 389–395, Dec. 2012.
[54] E. J. Merced-Grafals, N. Dávila, N. Ge, R. S. Williams, and J. P. Strachan, "Repeatable, accurate, and high speed multi-level programming of memristor 1T1R arrays for power efficient analog computing applications," *Nanotechnology*, vol. 27, no. 36, 2016, Art. no. 365202.
[55] M. Escudero-Lopez, F. Moll, A. Rubio, and I. Vourkas, "An on-line test strategy and analysis for a 1T1R crossbar memory," in *Proc. IEEE 23rd Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2017, pp. 120–125.
[56] C. Bengel, A. Siemon, F. Cüppers, S. Hoffmann-Eifert, A. Hardtdegen, M. von Witzleben, L. Hellmich, R. Waser, and S. Menzel, "Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using SPICE level compact models," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4618–4630, Dec. 2020.
[57] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.
[58] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges," *Adv. Mater.*, vol. 21, nos. 25–26, pp. 2632–2663, Jul. 2009.
[59] S. Menzel, U. Böttger, M. Wimmer, and M. Salinga, "Physics of the switching kinetics in resistive memories," *Adv. Funct. Mater.*, vol. 25, no. 40, pp. 6306–6325, Oct. 2015.
[60] F. Cüppers, S. Menzel, C. Bengel, A. Hardtdegen, M. von Witzleben, U. Böttger, R. Waser, and S. Hoffmann-Eifert, "Exploiting the switching dynamics of HfO_2-based ReRAM devices for reliable analog memristive behavior," *APL Mater.*, vol. 7, no. 9, Sep. 2019, Art. no. 091105.
[61] R. Dittmann, S. Menzel, and R. Waser, "Nanoionic memristive phenomena in metal oxides: The valence change mechanism," *Adv. Phys.*, vol. 70, no. 2, pp. 155–349, Apr. 2021.
[62] H. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.
[63] JARA-FIT. (2021). *JART VCM V1B VAR*. [Online]. Available: https://www.emrl.de/JART-files/JART%20VCM%201b%20verilog-a-var.va
[64] D. Gage, E. Laub, and B. McGarry, "Cellular automata: Is rule 30 random?" in *Proc. Midwest NKS Conf.*, 2005, pp. 1–15.



VASILEIOS NTINAS (Member, IEEE) received the D.Eng. and M.Sc. degrees in electrical and computer engineering from the Democritus University of Thrace (DUTh), Xanthi, Greece, in 2015 and 2017, respectively, and the Ph.D. degree from the Department of Electronic Engineering, Universitat Politecnica de Catalunia (UPC), Barcelona, Spain, in 2022. He is currently a Postdoctoral Research Associate with the Department of Electrical and Computer Engineering, Technische Universitat Dresden (TUD), Dresden, Germany. He has published more than 60 technical papers in peer-reviewed journals and conferences. His research interests include memristor modeling, stochastic resonance on memristor devices, and spatially-expanded memristor-based computing circuits and systems, i.e., cellular automata and cellular nonlinear networks.



IOSIF-ANGELOS FYRIGOS (Member, IEEE) received the D.Eng. degree in electrical and computer engineering from the Democritus University of Thrace (DUTh), Xanthi, Greece, in 2017, where he is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department. He has published more than 30 technical papers in peer-reviewed journals and conferences. His research interests include high performance computing, memristor modeling, and unconventional memristor-based electronic designs.

RAFAILIA-ELENI KARAMANI (Member, IEEE) received the D.Eng. degree from the Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece, in 2018, where she is currently pursuing the Ph.D. degree in electrical engineering. She is a Teaching Assistant with the Democritus University of Thrace. She has published 11 technical papers in peer-reviewed journals and conferences. For her Ph.D. studies, she has received scholarships from the Bodossaki Foundation (2018–2019) and the Hellenic Foundation for Research and Innovation (HFRI 2019–2022). Her research interests include memristor-based electronic design, bio-inspired circuits and systems, and cellular automata.

NIKOLAOS VASILEIADIS (Member, IEEE) received the D.Eng. and M.Sc. degrees in electrical and computer engineering from the Democritus University of Thrace (DUTh), Xanthi, Greece, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree. He is an Assistant Researcher with the Nanoscience and Nanotechnology Institute, National Center for Scientific Research ''Demokritos'' (NCSR). His research interests include bio-inspired hardware technologies, memristive hardware accelerators, neural networks, and electronic designs.

PANAGIOTIS DIMITRAKIS (Senior Member, IEEE) received the B.S. and M.S. degrees in physics from the University of Athens, Greece, in 1995 and 1998, respectively, and the Ph.D. degree from the National Technical University of Athens, Greece, in 2006. He joined NCSR ''Demokritos,'' in 2007, where he is currently the Director of research with the Institute of Nanoscience and Nanotechnology. He is the Manager of the Central Nanotechnology and Microsystems Cleanroom Facility. He is the author of more than 60 articles in peer-reviewed journals in the field of semiconductor devices, nonvolatile memories, and organic electronics. He is an Editor of the book series ''Charge-Trapping Nonvolatile Memories.'' He has organized various international conferences worldwide and edited their proceedings.

ANTONIO RUBIO (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees from the Industrial Engineering Faculty, Universitat Politécnica de Catalunya (UPC), Barcelona, Spain. He was an Associate Professor with the Electronic Engineering Department, UPC; and a Professor with the Physics Department, Balearic Islands University, Palma, Spain. He is currently a Professor of electronic technology with the Telecommunication Engineering Faculty, UPC. His research interests include VLSI design and test, device and circuit modeling, high-speed circuit design, and new emerging nanodevices, and nanoarchitectures. He has served as IEEE Computer Sciences Integrity Chair, from 2020 to 2021. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, a Senior Editor of the IEEE TRANSACTIONS OF NANOTECHNOLOGY, and the Chair of IEEE CASS TC Nano-Giga.

GEORGIOS CH. SIRAKOULIS (Member, IEEE) received the Dipl.Eng. and Ph.D. degrees in electrical and computer engineering from the Democritus University of Thrace (DUTh), Greece, in 1996 and 2001, respectively. He is a Professor and the Head of the Department of Electrical and Computer Engineering, DUTh. He has published more than 290 technical papers. He is the co-editor of seven books, the coauthor of 27 book chapters, and a guest editor of 14 special issues. His current research interests include complex electronic systems, future and emergent electronic devices, circuits, models, and architectures, unconventional computing memristors, cellular automata, quantum cellular automata, bioinspired computation/biocomputation and bioengineering, and modeling and simulation. He is an Editor of IEEE TRANSACTIONS ON NANOTECHNOLOGY, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, *Microelectronics Journal*, *Integration, the VLSI Journal*, and *Journal of Cellular Automata*.