# Crossbar-Based Memristive Logic-In-Memory Architecture

Georgios Papandroulidakis, Ioannis Vourkas, *Member, IEEE*, Angel Abusleme, *Member, IEEE*, Georgios Ch. Sirakoulis, *Member, IEEE*, and Antonio Rubio, *Senior Member, IEEE*

***Abstract*—The use of memristors and resistive random access memory (ReRAM) technology to perform logic computations has drawn considerable attention from researchers in recent years. However, the topological aspects of the underlying ReRAM architecture and its organization have received less attention, as the focus has mainly been on device-specific properties for functionally complete logic gates through conditional switching in ReRAM circuits. A careful investigation and optimization of the target geometry is thus highly desirable for the implementation of logic-in-memory architectures. In this paper, we propose a crossbar-based in-memory parallel processing system in which, through the heterogeneity of the resistive cross-point devices, we achieve local information processing in a state-of-the-art ReRAM crossbar architecture with vertical group-accessed transistors as cross-point selector devices. We primarily focus on the array organization, information storage, and processing flow, while proposing a novel geometry for the cross-point selection lines to mitigate current sneak-paths during an arbitrary number of possible parallel logic computations. We present an analysis of circuit resources, integration density, and logic computation parallelism and prove the proper functioning and potential capabilities of the proposed architecture through SPICE-level circuit simulations of half-adder (HA) and sum-of-products logic functions.**

***Index Terms*—computing, crossbar, digital logic, memristor, resistive RAM (ReRAM), resistive switching**

## I. Introduction

FOR some time now, advances in semiconductor technology have continued to boost both the memory capacity and computing speed of modern computers. In this regard, among today's various emerging memory technologies [1], resistive random access memory (ReRAM) based on resistive switching nanodevices (memristors or memristive devices) [2] stands out as one of the best-studied and most promising candidates for next-generation nonvolatile memory (NVM) applications [3]-[5]. ReRAM has several attractive properties, such as fast operation, low power consumption, multilevel single-cell storage, and very high integration density ($4F^2$ footprint, where $F$ is the minimum feature size of the process technology), owing to the simple, dense, high-connectivity structure of the nanocrossbar geometry [6], [7]. Additionally, a paradigm shift is needed in computing systems beyond the classical and so far dominant von Neumann architecture, which separates storage from computation in distinct units [8]. Data processing in von Neumann systems is normally carried out sequentially, thus requiring a lot of information exchange and communication between the central processing unit(s) (CPU(s)) and data storage module(s). Moreover, today's computers are able to process data at CPU speeds much faster than their memory-access speed, making the latter a true bottleneck [1]. In an attempt to overcome such limitations and further improve data-processing efficiency, research has therefore recently begun to focus on brain-inspired (neuromorphic) and, more generally, in-memory computing approaches [9]-[12].

In this context, the memristor provides an unconventional computing framework, ideally combining resistance-based information storage and processing in a single device [13]. Several recently published logic circuit design approaches [14]-[20] use memristors as binary elements in a digital platform, offering functionally complete logic gates and promising to maximize the benefits of digital computing in future system architectures in which memory and processing co-exist. Nevertheless, mostly owing to the fact that memristor device technology is still at an early stage, such papers have primarily focused on the digital logic realization procedure, logic gate implementation, and/or the device-level requirements, and almost all of them have omitted (or left for future research) the study of the underlying ReRAM organization, the target circuit architecture, and the impact of the driving circuitry. Further research is necessary at the circuit and/or architecture level to make logic computations as parallel as possible and thus enable practical application [21], [22]. It would be particularly interesting to see whether and how such logic design approaches could practically fit in compact memristive storage circuit architectures, exploiting the nonvolatility of memristors in normal power-off (and thus more energy efficient) logic-in-memory circuits.

In this paper, we aim to address precisely this gap. More specifically, we propose an initial approach to a crossbar-based in-memory processing system in which the binary information stored in memristors is locally processed in the same unit: i) without

the stored data being affected while it is used as logic input; and ii) keeping the logic result in the same state as the memristors, where the computation takes place. The basic concept is to take advantage of a dense crossbar array that is heterogeneous in terms of its cross-point resistive devices while also using a novel group-accessing scheme for the selection lines of the target ReRAM cross-points. Our study particularly focuses on the ReRAM organization, taking into consideration: i) device-level memristor properties, by incorporating a threshold-type SPICE-compatible switching model of bipolar voltage-controlled memristors [23], [24]; ii) circuit-level properties, by adopting a functionally complete memristor-based digital logic design methodology that allows for single-step multi-input parallel logic computations [20]; and iii) architecture-level details assuming state-of-the-art high-density and CMOS-compatible memristor-transistor crossbar geometry with a group-accessed vertical (which could be thought of as a nano-pillar vertical gate-all-around or VGAA) transistor as the cross-point selector underneath each memristive device [25]-[27]. In this way we achieve: i) memory and logic operations through gate-controlled resistance switching with no current sneak-paths [28]; ii) a much smaller cell footprint compared to that of traditionally planar transistors; and iii) lower operating power compared to that of a typical passive cross-point array. We justify our choice based on the memristor logic implementation methodology and comment on the overall system performance, the integration density, and the achieved parallelism of the logic computations. Finally, we provide SPICE circuit simulation results to confirm the correct operation of the proposed architecture for examples of sum-of-product and half-adder (HA) logic functions, highlighting the advantages of the proposed logic-in-memory approach.

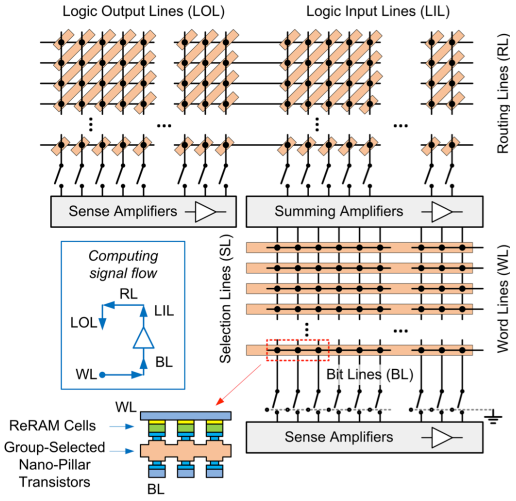


Fig. 1. Basic topological and organizational features of the proposed architecture. The insets show a cross-section of the vertical stacking structure and an indicative signal flow for the computing process.

## II. Topology Description and Analysis

### A. Main Topological Features

The general floor plan, showing the basic modules included in the proposed architecture, is given in Fig. 1. Overall, it consists of two banks of crossbar arrays and their supporting circuit modules. In fact, there are two separate nano-pillar (vertical) transistor-memristor (1T1R) crossbar arrays, i.e., the memory crossbar, at the bottom of the figure, and the computing crossbar, which is larger and located at the top. Black dots generally denote cross-points, whereas the horizontal and diagonal rectangles denote the topology of the transistor selection lines (SLs), i.e., the groups of the select transistors whose gate terminals are simultaneously driven.

As shown in Fig. 1, the cross-point stacking structure of the memory crossbar array comprises horizontal word lines (WLs) at the top and vertical bit lines (BLs) at the bottom. Drivers for the WLs and SLs are assumed on opposite sides of the array to better distribute the layouts of the peripheral circuitry of the CMOS devices used for the selection and application of the voltage pulses required in each access operation. The inset shows a cross-section of part of a memory word with the ReRAM cells (shown as a two-material stack without loss of generality) stacked directly on top of group-selected nano-pillar VGAA transistors. We assume that the ReRAM and the selection device layers are deposited sequentially, with the transistors placed on the bottom layer (fabricated as front-end transistors) to limit the influences of the parasitic capacitance and resistance and to minimize the area penalty. An excellent VGAA transistor in this case could be the Si nano-pillar MOSFET by [25], which demonstrates very good gate controllability and less than 0.1nA leakage current and renders a $4F^2$ footprint cross-point cell, much smaller than the 8-12$F^2$ of the traditionally planar 1T1R cell. Moreover, group-accessing of the SLs makes it possible to minimize the number of transistor gate lines from a total of WLs×BLs (when accessing every transistor separately) to only WLs, thereby simplifying significantly the crossbar fabrication process as specifically discussed in [3].

During the state programming and/or reading memory processes, the input signal simply flows from the WL to the BL. The latter (depending on the access operation) can be either grounded, left floating, or driven to the sense amplifiers via control switches, as shown at the bottom of Fig. 1. More specifically, write and read signals are applied to the target WL, whose respective SL is activated. Reading is a one-step process: all the BLs are driven to the sense amplifiers (one for each BL). Writing, however, is a two-step process: since the bias direction is mutually reversed in the SET and RESET processes for bipolar ReRAM cells, they are performed in separate cycles. For instance, the BLs of the ReRAM cells that will be SET are grounded first, while the rest of the BLs are left floating. In the very next cycle, the previously grounded BLs are left floating and

the rest are grounded for the RESET process.

On the opposite (top) side of this array, the BLs are connected to summing amplifiers (whose role will be explained later), which separate the memory array from the computing array, found at the top of the system floor plan. The cross-point stacking structure of the computing array is practically the same. For the orientation of its top/bottom nanowires we will assume that the vertical logic input/output lines (LILs and LOLs, respectively) are at the top of the structure, while the horizontal routing lines (RLs) are at the bottom. Whether each logic line is named LIL or LOL depends on whether it is connected to the summing or sense amplifiers, respectively. As shown in the respective arrow diagram in the inset, during computations the input signal follows a circular flow, starting from the WL of the memory array, moving to the BL and the LIL through the summing amplifiers, then to the RL, and finally to the LOL and the sense amplifiers. A set of control switches makes it possible to drive the output of the summing amplifiers to the LIL. Depending on the activated SL, the signal always flows through two 1T1R cells, i.e., through two memristors (with the same or opposite polarities) and two transistors, all connected in series via a common horizontal RL. The basic concept is to employ complementary material stacking structures in different cross-point cells of the computing array, i.e., the regular ReRAM stack and the one with a reversed material deposition order [29], [30], in order to have both forward- and reversely-polarized memristors for the purposes of the computations, which are explained in the following section.

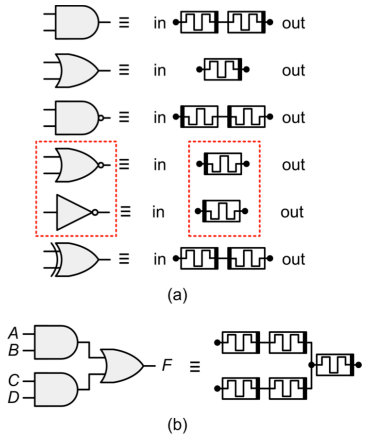


Fig. 2. Equivalent memristive circuit implementing: (a) 2-input AND, OR, NAND, NOR, XOR logic gates and a NOT gate (inverter); (b) a typical sum-of-products logic function with cascaded logic gates. The dashed rectangle highlights the use of a single reversely polarized memristor to implement NOR and NOT logic operations [20].

Furthermore, in order to maintain the high controllability and favorable implementation properties offered by the group-accessed cross-points, while also permitting the parallel execution of several logic computations, we introduced a novel geometry for the SL of the computing array, assuming twisted transistor gate nanowires. We will next show that when this group-selection strategy for the twisted SL is used, there are no disturbing current sneak paths when multiple SLs are driven, provided the simultaneously driven SLs follow certain acceptable patterns. The simple two-terminal structure of memristors and the proposed twisted SL topology enable the integration of digital logic computations in the crossbar, where the required serial ReRAM connection [20] is naturally achieved. However, this novel SL geometry entails a small area overhead. The 1T1R cross-points with the nano-pillar transistor will require a row- and column-pitch of $3F$ in order to accommodate the transistor channel width ($1F$), the gate surrounding the channel ($1F$), and the spacer ($1F$) [30]. Therefore, the computing cross-point cell area becomes $9F^2$, i.e., 2.25× larger than that of the $4F^2$ footprint memory crossbar.

### *B. Basics of ReRAM-based Logic Circuits*

The memristor-based logic computations in our work rely on the memristive logic family proposed by Papandroulidakis *et al*. in [20]. It is a parallel-processing, functionally complete logic design scheme, unlike some published sequential processing approaches, such as the CMOS-*like* [14], MAGIC [17], and IMPLY [19] logics. It enables the parallel execution of single-step digital logic operations, exploiting the threshold-dependent switching behavior of memristors and of their simple series connection. Compared to other parallel processing logic design concepts, such as the MRL proposed by Kvatinsky *et al*. [18], it enables the execution of a wider variety of logic operations based on devices with sharp transitions (filamentary or threshold-type) instead of linear (or homogeneous) switching devices, which generally respond more slowly to the applied input signals [31].

This memristive logic family uses the total conductance (memductance) of the devices for the parallel computation of AND, OR, NAND, NOR, XOR, and NOT logic operations. Fig. 2 summarizes the general circuit concept for the implementation of the aforementioned logic gates and sum-of-products logic functions. Understanding the overall circuit behavior, sometimes based on collective dynamics of two properly polarized memristors, requires comprehending the switching dynamics of individual memristors first. To this end, it is worth noting that bipolar memristors with opposite polarities will tend to switch their states in a reciprocal manner [32]. Hereinafter we will refer to a memristor being forward/reversely polarized (FPM/RPM) when the voltage is applied to one terminal (top or bottom) with the opposite terminal (bottom or top) being grounded; the bottom terminal is always denoted by the thick black line in the circuit schematics. Moreover, we will assume that the resistance will decrease when the memristor is forward-biased and increase when it is reversely biased. In fact, in threshold-type switching memristors the resistance change-rate is very fast above (and negligibly slow below) the voltage threshold $V_{\mathrm{SET}}$ or $V_{\mathrm{RESET}}$, which determines

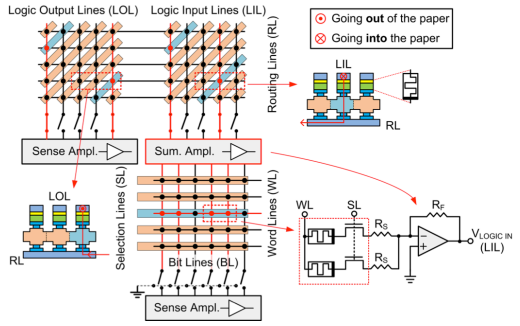


Fig. 3. System snapshot highlighting operational details for parallel logic computations. The inset shows the equivalent circuit schematic between word lines (WLs) and logic input lines (LILs). A cross-section of two parts of the computing array indicates the 1T1R cell structure, the memristor polarity, and the signal flow between LIL and LOL.

the SET ($R_{OFF}\rightarrow R_{ON}$) or RESET ($R_{ON}\rightarrow R_{OFF}$) transition, respectively.

In this logic design scheme [20] the input voltages consist of: i) a very low voltage for logic "0"; and ii) a voltage higher than the threshold $V_{SET}$ (and/or $|V_{RESET}|$) for logic "1." However, the key idea is that any binary input logic combination is encoded into a corresponding positive aggregate input voltage (the sum of the separate input voltages), which is then applied to the input terminal of the memristive gate, the latter being any of the six options shown in Fig. 2 (a). We will now briefly describe how the overall conductance (memductance) can be used for such logic computations. For example, a single FPM will switch from a low conductance (L) to a high conductance (H) if either (or both) of the applied inputs is logic "1," i.e., if it exceeds $V_{SET}$. Likewise, when two FPMs are in series, the composite memductance will rise from a low value (L') to a high value (H') only if both inputs are logic "1," meaning that the aggregate input voltage will exceed $2\times V_{SET}$. Apparently, memductance in these two cases defines the OR and AND logic operations, respectively, as functions of the aggregate applied input voltage. Thus, requiring the "aggregate" input voltage explains the use of summing amplifiers in the proposed system. The operation of the rest of the logic gates is explained in a similar manner in [20]. For instance, a single reversely polarized memristor (RPM) implements the NOR gate since it will switch from a high conductance (H) to low conductance (L) if either (or both) of the applied inputs is logic "1," i.e., if it exceeds $V_{RESET}$. Likewise, when only one logic input is considered, the RPM is equivalent to a NOT gate as well.

Because it is based on memristors connected only in series, this logic design fits well with the structural specifications of the proposed system. Furthermore, since memristors with opposite polarity are required for some of the logic operations, we assume that both FPMs and RPMs are readily available inside the heterogeneous (in terms of the cross-point devices) computing array. This concept of heterogeneity was first proposed in [33]. Logic operations are conducted via conditional switching of nonvolatile ReRAM cells, featuring no gain. Any logic input combination will have an irreversible effect on the conductance of the memristors. For instance, if we first apply either "10" or "01" and then, immediately after, "00," then due to the nonvolatility of the memristors, the final result will not be correct. Consequently, it is necessary to initialize every gate via a reset pulse in between each input logic combination, a requirement common to several memristive logic design approaches [17], [19]. To this end, initialization drivers access the RL and LIL/LOL and are assumed to be distributed around the computing-array. This notwithstanding, there is no need for a reset step between "01" and "10" inputs. The same is true if the next input, after a "10" or "01," is "11," meaning that algorithmically the efficiency of such logic circuits could be improved by evaluating the number of "1s" in the input logic combination; however, that falls beyond the scope of this paper.

### *C. Operational Features and Performance*

#### *1) Computing flow characteristics*

Having explained the basic topological characteristics of the proposed system, as well as the basics of the logic design scheme using memristors, we will now provide a specific example to highlight the most important system- and circuit-level operational properties. The system floor plan shown in Fig. 3 includes a 5×6 1T1R memory array at the bottom and a 5×10 1T1R computing crossbar at the top. These dimensions are merely indicative for the purposes of this example and do not demonstrate any particular requirement for the dimensional ratio of the two crossbar banks. However, it is worth noting that, in order to perform two-input logic computations, two BLs from the memory array should correspond to one LIL from the computing array, i.e., there should be 2x as many BLs as LILs. For logic operations with more than two logic input variables, more BLs should correspond to each LIL and the computing ReRAM cells should be changed accordingly, as described in [20].

In this context, Fig. 3 provides a snapshot of a system supporting only two-input logic operations. Among the selection lines, those being activated are highlighted in light blue. The aim is to involve the data stored in the memory array (without affecting them) in multiple single-step parallel logic operations. In this example, we assume that we want to perform two parallel logic operations with the two red-dot pairs of cells of the activated memory word. The red dots generally denote the cross-point cells currently being used. Thus, a read voltage pulse (with an amplitude lower than the switching threshold of the memristors) is applied to the target WL and the corresponding horizontal SL is driven. The BL control switches at the bottom are left floating and all the BLs are connected to the summing amplifiers at the top. The inset shows the equivalent circuit schematic; the WL input signal passes through two 1T1R cells before reaching the $R_S$ input resistors of the summing amplifiers. In this way, depending on the binary state of these memristors and, thus, on the voltage drop on them, the corresponding BL will have either a high (logic "1") or low (logic "0") voltage. In other words, through the summing amplifiers, we compute a weighted sum of the

commonly applied WL read voltage, which identifies the stored binary state of the memristors. For example, the case "$R_{OFF}$, $R_{OFF}$" == "00" will give a very small voltage sum, whereas "$R_{OFF}$, $R_{ON}$" == "01" (or, equivalently, "10") will give a high voltage sum, and "$R_{ON}$, $R_{ON}$" == "11" will result in the highest input voltage sum, depending on the $R_S$ and $R_F$ resistor values. This sum is in fact the input voltage for our logic computation and is driven to the LIL through the corresponding control switches. In this example we show only two of them in closed position, one for each parallel logic operation. Next, the LIL signal passes through the series-connected ReRAM cells (via the bottom RL), whose twisted SLs are driven, finally reaching the LOL, where another set of control switches (two of them are again shown closed) drive the logic output to the sense amplifiers.

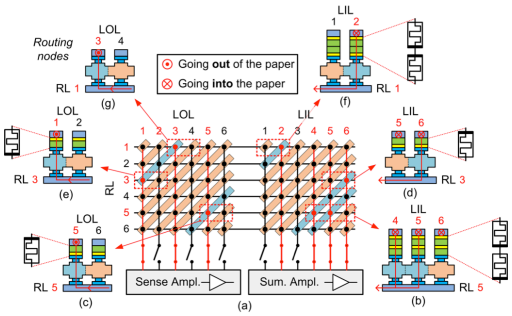


Fig. 4. Heterogeneous crossbar for ReRAM-based logic computations. (a) A snapshot of the computing crossbar bank highlighting the control line configuration (open/closed), the driven twisted SLs (light blue), and the cross-points involved in three parallel logic operations (red). (b)-(g) Cross-section views of the specific parts of the array where computation takes place, highlighting the ReRAM cell variety, including typical routing junctions, device polarity, and signal flow through the ReRAM cells whose series transistors are driven. Specifically: (b) and (c) show a sum-of-products operation; (d) and (e) show a NAND gate; and (f) and (g) show an XOR gate, all in parallel.

The logic gates always involve ReRAM cells that are connected to a common RL. The inset in Fig. 3 also shows a cross-section of two particular parts of the same RL of the computing array, with a view to clarifying the cross-point stack structure, the polarity of the memristors, and the current flow, which passes through the two series 1T1R cells. The symbol × inside a circle denotes a signal entering the plane of the paper vertically, whereas a ● inside a circle denotes a signal coming out of the plane of the paper at that point. As can be seen, the input logic signal crosses two memristors with opposite polarities, implementing an XOR gate (see Fig. 2). The other simultaneous two-bit logic operation shown in Fig. 3 (without specifying the cross-point cell type), involving the leftmost pair of red-dot memory cross-points, occurs in the same fashion.

*2) Exploitation of cross-point heterogeneity*

As mentioned previously, the proposed computing crossbar bank consists of an array that is heterogeneous in terms of its cross-point devices. Specifically, we assume that different existing ReRAM cells might have a reversed material deposition order, thus enabling both forward- and reversely polarized memristors that are physically connected in series via the RL to form the desired ReRAM-based logic gates. Furthermore, single-cell repetitive material-stacking structures could enable the implementation of two series ReRAM devices (with the same or opposite polarity) in a single cross-point [29], [30], i.e., 1T2R cells. Additionally, we assume the existence of typical 1T routing junctions (i.e., vertical transistor cross-points without ReRAM device) at certain RL-LOL intersections to directly drive a logic signal from the RL to the LOL. The latter are required, e.g., for single-memristor logic gates (see Fig. 2), where computation is completed in the LIL-RL cross-points.

This heterogeneity concept is summarized in Fig. 4, which shows the computing crossbar bank configuration for an example of three parallel logic operations. This example consists of a sum-of-products computation, a NAND, and an XOR gate, which take place in RL 5, 3, and 1, respectively. Ideally the array could be divided thematically in several islands (sub-arrays) depending on the cross-point type. Specifically, the LIL-RL cross-points may include either 1T1R or 1T2R cells with all possible polarity combinations, whereas 1T1R or 1T cells are assumed in the LOL-RL cross-points. The 1T1R LIL-RL cross-points [Fig. 4(d)] should be combined with 1T1R [Fig. 4(c), (e)] or 1T [Fig. 4(g)] LOL-RL cross-points to either physically connect two ReRAM devices [Fig. 4(d), (e)] or route a single ReRAM computation directly to the LOL, respectively. Similarly, 1T2R [Fig. 4(b), (f)] LIL-RL cross-points might be combined with either 1T1R or 1T LOL-RL cross-points to either cascade logic gates in sum-of-products computations [Fig. 4(b), (c)] or route a two-ReRAM computation directly to the LOL [Fig. 4(f), (g)], respectively.

*3) Twisted SL driving patterns*

The proposed twisted gate nanowires (SL) for the nano-pillar transistors of the computing bank simplify the array organization and the select/deselect schemes of the RRAM cells used in logic circuits. Given the need to be able to perform as many parallel computations as possible, using a different SL geometry would not work due to current leakage/sneak-paths [32], [34], which contribute to incorrect computations and/or increases in power consumption. For example, neither row nor column SLs would work with our approach. With column SLs, i.e., in-line with the LIL/LOL, the output current of a summing amplifier in a particular LIL would flow through all the cross-points sharing the same column SL. Likewise, if row SLs are used in the memory bank, only one logic operation could be performed at a time.

A relevant example of this row SL (horizontal) geometry is shown in Fig. 5 (a), which assumes two parallel logic computations taking place in RL 3 and 5, respectively. In the first, the current should flow only through LIL 4, then RL 3, and finally LOL 5. Similarly, in the second, the current should flow only through LIL 2, then RL 5, and finally LOL 3. Unfortunately, current from LIL 2 and LIL 4 also flows through the cross-points indicated with dashed squares, which share the

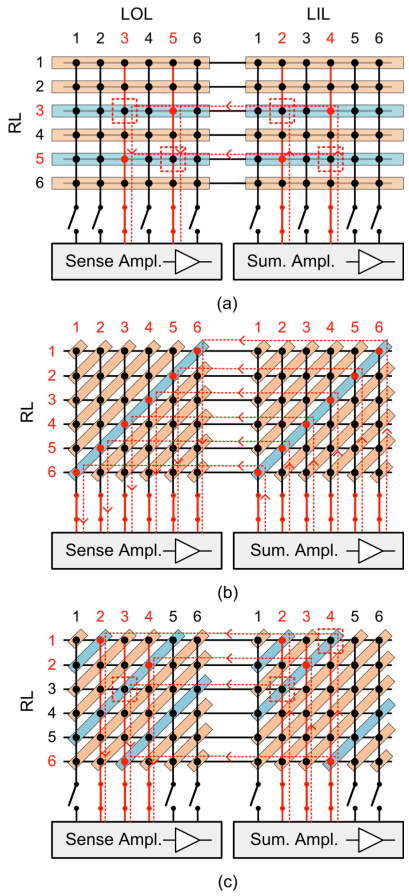


Fig. 5. Computing crossbar SL geometry and configuration examples. (a) Improper row-type (horizontal) SL. (b) Twisted SL, highlighting the current flow when the maximum number of parallel logic operations takes place. (c) Possible sneak-path effect when improper SL gating patterns are used. Dashed lines (rectangles) denote the current flow (the leaking cross-points).

same gated SL, thereby increasing the likelihood of incorrect logic computations. Not only could such current sneak-paths affect the results of logic computations, even if they did not, they would increase power consumption.

On the other hand, when the twisted SLs are properly gated, they successfully address the sneak-path effect and, most importantly, enable the maximum number of simultaneous logic computations, which is equal to the number of LOLs. Fig. 5(b) shows a snapshot of such an array, where all LIL and LOL control switches are closed and the maximum number of logic operations takes place in parallel using the ReRAM cells of the anti-diagonal (it could be the main diagonal if the symmetric orientation were selected for the twisted SLs) of the two sub-arrays defined by the LOL and LIL columns. However, current sneak-paths could still appear, as shown in Fig. 5(c), which shows the SLs and the control line configuration for three parallel logic computations in RL 1, 2, and 6. As in Fig. 5(a), here currents from LIL 2 and 4 also contribute to incorrect computations.

Therefore, in order to perform more than one parallel logic computation safely, the twisted SLs of both the RL-LOL and the RL-LIL sub-arrays of the computing crossbar bank should be properly configured to prevent the creation of current-leakage paths. One simple rule for achieving this is as follows: *any pair of an LIL and an LOL simultaneously in use should never have* (*i*) *two unused gated cross-points or* (*ii*) *one unused and one used gated cross-point connected via an RL* (*i.e. in the same row*). Fig. 5(c) shows case (i) in RL 3 for LOL 3 and LIL 2 and case (ii) in RL 1 for LOL 2 and LIL 4, respectively.

In keeping with the aforementioned rule, in Fig. 6 we show all six SL gating combinations enabling maximum, safe utilization of the LIL/LOL columns simultaneously, using an indicative 6×6 array example. According to these patterns, every LIL/LOL column of the array has only one gated cross-point in use (red dot), such that every RL connects only two gated cross-points. In fact, assuming an array with $m$ rows and $n$ columns ($m$×$n$), there are $m$ different SL gating patterns that enable the simultaneous use of all $n$ LIL/LOL columns, utilizing in parallel up to $n$ of the $m$×$n$ total cross-points. Finally, it is worth mentioning that using the same LIL signal in two simultaneous logic operations is also possible if the SL gating pattern used complies with the aforementioned sneak-path avoidance rule.

## III. SPICE Simulation Results

As a proof of concept, in this section we present a simulation-based validation of the normally-off ReRAM logic circuits implemented in the proposed crossbar-based topology, using a Cadence PSPICE simulation environment. The logic computations were performed through conditional resistance switching in ReRAM cells, serially connected according to the applied transistor gating signals, thereby implementing the logic gate structures. The logic circuit examples consist of a half adder (HA) with XOR and AND logic gates for the Sum and Carry bits, and a sum-of-products function (see Fig. 2). In the simulations, we employed a SPICE-compatible threshold-type model of a voltage-controlled bipolar memristor, which attributes the resistance-switching effect to the modulation of an effective tunneling distance [23]. Values of the parameters of the model were set as: $\{a, b, c, m, f_0, L_0\} = \{15000, 0, 0.1, 82, 310\}$, with $R_{OFF} = 200\text{K}\Omega$, and $R_{ON} = 2\text{K}\Omega$. Variation of switching thresholds (i.e., $V_{SET,RESET} \pm v_o$) should ideally be minimized as much as possible to ensure high reliability [34], [35]. In the simulations, the threshold values $V_{SET} = |V_{RESET}| = 0.3\text{V}$ were used as mean values of normal distributions with a small standard deviation, which we then used to decide on the proper programming pulse amplitudes. Generally, the memristor switched its state rapidly (at a ms regime based on the abovementioned parameter values) as soon as the voltage drop on it exceeded either of its voltage

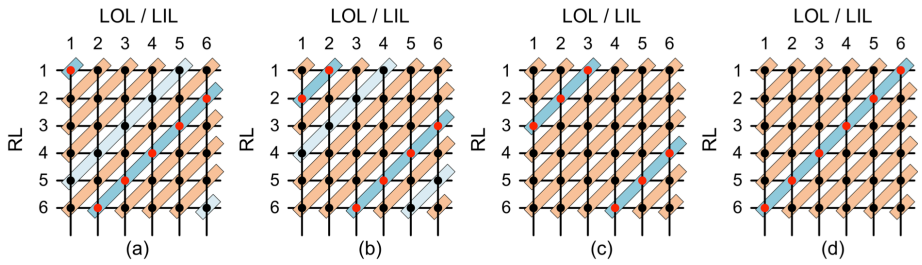


Fig. 6. (a)-(d) Computing crossbar safe SL gating patterns for a 6×6 array example. The gated nanowires are highlighted in light blue and the cross-points used, in red. (a) and (b) also include the symmetric version of the shown pattern, highlighted in a different color.

thresholds.

In keeping with the proposed system floor plan shown in Fig. 1, Fig. 7 shows in detail the equivalent circuit we simulated in SPICE. This circuit consisted of a 4×4 1T1R memory array with horizontal WLs and SLs, and a 5×4 1T1R heterogeneous computing array with two LILs, two LOLs, and twisted SLs (TSLs). The latter were called input TSLs (iTSLs) or output TSLs (oTSLs) depending on whether they were in the RL-LIL or the RL-LOL sub-array of the computing bank. In the memory bank below, the data stored in $BL_{1\text{-}2}$ ($BL_{3\text{-}4}$) defined the two-bit logic input voltage in $LIL_1$ ($LIL_2$). Thus, in the four memory words, the memristor pairs in $BL_{1\text{-}2}$ and $BL_{3\text{-}4}$ were previously programmed accordingly so as to hold all possible two-bit input combinations, which were sequentially driven to the logic gates as we applied a 0.2V read voltage to one of the WLs ($WL_{1\text{-}4}$).

Cross-point heterogeneity is example-specific. Consequently, we introduced variations in the simulated circuit, i.e., since AND and XOR gates require memristors with the same and opposite polarity, respectively, we assumed both FPM and RPM cells for the HA example, as shown in Fig. 7. On the other hand, for the sum-of-products example, we used 1T2R cross-point cells with two FPMs stacked in series (AND gate) in the same cross-point in the RL-LIL sub-array (as shown previously in Fig. 4). Moreover, all the CMOS control switches and the cross-point nano-pillar transistors in SPICE were modeled using n-type FETs for their wide channel resistance range between the ON and OFF states. The applied transistor gate voltages were either 0V or 1.1V. For the summing amplifiers, we used $\{R_S, R_F\} = \{1, 8\}$KΩ resistors. Their output consisted of a voltage lower than the memristor thresholds for logic "00," a voltage ≈-0.54V for logic "01" or "10," and a voltage ≈-1.1V for logic "11." This voltage was driven to the LIL through the input control lines (CILs), whereas the logic output was obtained at the intermediate node of the LOL and the 2KΩ pull-down resistor ($R_{PD}$) network via the output control lines (COLs). Due to the negative sign of the summing amplifier output, the devices in the computing bank had the opposite polarity of that shown in Fig. 2, without loss of generality. Moreover, for simplicity and to avoid resetting the memristive gates in between the application of different input combinations, we alternated between different memristive gates of the same type for different logic input combinations, so the target memristors were always correctly initialized.

Fig. 8 shows the simulation results for the HA circuit. As we sequentially read $WL_{1\text{-}4}$, the logic input voltage at the output of the summing amplifiers corresponding to data in $BL_{1\text{-}2}$ ($BL_{3\text{-}4}$) is shown in the form of green (red) 100ms-wide pulses [Fig. 8(a)]. When driving $WL_1$, we simultaneously drove $iTSL_2$ and $oTSL_2$, respectively (see Fig. 7). Likewise, when driving the rest of the WLs ($WL_{2\text{-}4}$), we simultaneously drove $iTSL_{3\text{-}5}$ and $oTSL_{3\text{-}5}$, whereas $iTSL/oTSL_{1\text{ and }6}$ were never used. Cross-point cells in $LIL_1$ and $LOL_1$ corresponding to the AND gate and the Carry bit voltage output are shown in Fig. 8(b). Likewise, cross-point cells in $LIL_2$ and $LOL_2$ corresponding to the XOR gate and the Sum bit voltage output are shown in Fig. 8(c). The readout voltages for the logic "1" output reached approximately 110mV for the XOR gate and 350mV for the AND gate. The readout voltages confirm the expected operation of the memristive HA. The aforementioned difference in the output voltage amplitude can be attributed to the final composite series memristance in each gate, which is higher for the XOR gate; thus, a smaller voltage drops on the $R_{PD2}$ resistor. The memristive HA occupies four cross-points of the computing bank each with a cell area of $9F^2$ (due to the space-constraints between the TSLs), yielding a respective total circuit area of $36F^2$.

Fig. 9 presents the simulation results for the sum-of-products logic circuit. Unlike in the previous example, the RL-LIL sub-array cross-points are now 1T2R-type. The rest of the circuit remained as shown in Fig. 7. For each WL datum we simultaneously drove two AND gates and one OR gate connected via the same RL. Thus, when driving $WL_1$, we simultaneously drove $iTSL_1$, $iTSL_2$, and $oTSL_1$, respectively. Likewise, when driving $WL_2$, we simultaneously drove $iTSL_2$, $iTSL_3$, and $oTSL_2$, etc. Each LIL corresponded to one logic product (logic AND), and the logic sum of the two products was computed in the cross-point cell of $LOL_1$. The cross-points of $RL_5$ were never used. Fig. 9(a) shows the input voltage at the output of the summing amplifiers, whereas Fig. 9(b) shows the readout voltage on the $R_{PD1}$ resistor. The latter reached approximately 140mV for the logic "1" output only when we applied an input combination of "11," i.e., in the first and last case, thereby confirming the expected response. Fig. 9(c) shows the change in the memristance of the memristors in the computing bank. For clarity, we show only the memristance of the six cross-point ReRAM cells used in $RL_1$ and $RL_4$, where we expect to see changes in accordance

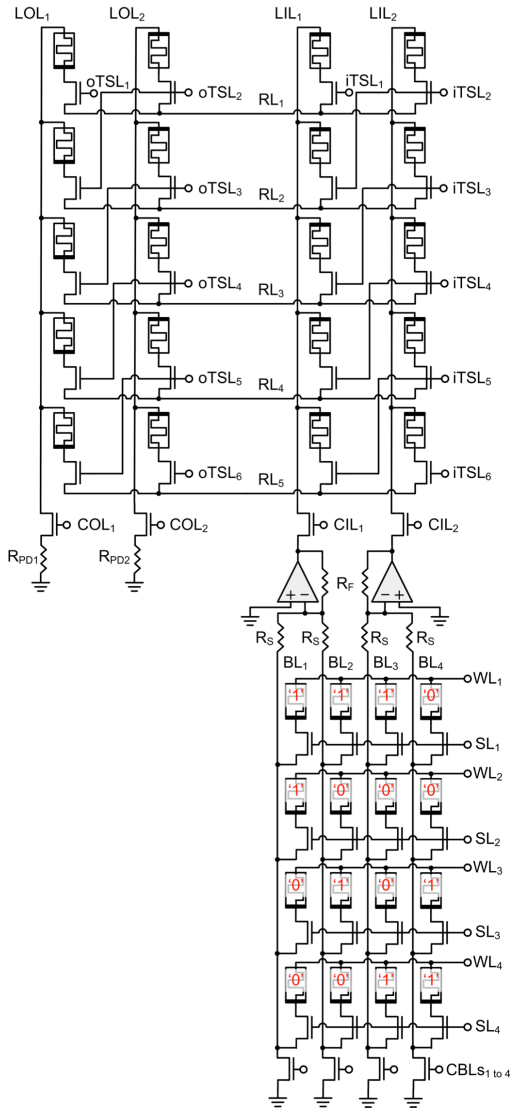


Fig. 7. Circuit schematic used in simulations, corresponding to the system floor plan shown in Fig. 1. The stored logic state of the memory cross-point ReRAM cells is shown in red.

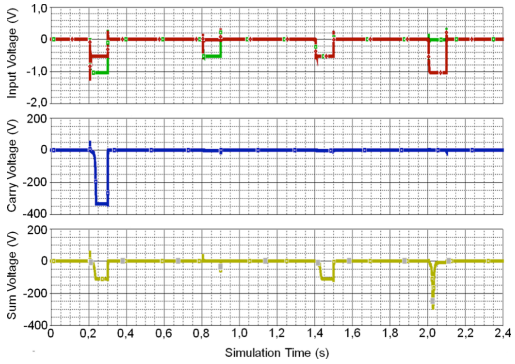


Fig. 8. Half-adder circuit simulation results. (a) Logic input voltages measured at $LIL_1$ (green) and $LIL_2$ (red). (b) Carry bit (AND gate) readout voltage on $R_{PD1}$. (c) Sum bit (XOR gate) readout voltage on $R_{PD2}$.

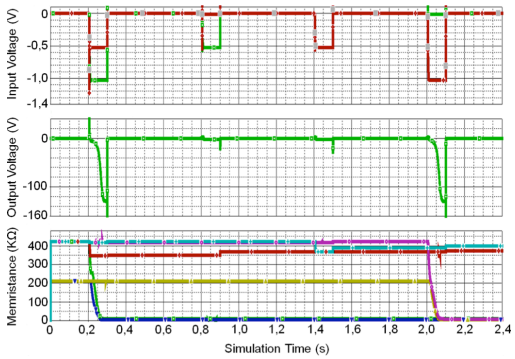


Fig. 9. Sum-of-products circuit simulation results. (a) Logic input voltages measured at $LIL_1$ (green) and $LIL_2$ (red). (b) Readout voltage on $R_{PD1}$. (c) Measured memristance values of the six cross-point cells in $RL_1$ and $RL_4$, showing that the corresponding change of state occurs only when a "11" input is applied to any of the AND gates.

with the inputs applied. Regarding $RL_1$, the memristors of the AND gate in $LIL_1$, which receives the "11" input, changed their composite memristance (green curve) from 400KΩ to approximately 4KΩ, whereas those forming the other AND gate in $LIL_2$ did not change their state (cyan curve). The change in the $LIL_1$ devices caused the subsequent change in the memristor in $LOL_1$ (blue curve), which corresponds to the OR gate. Similarly, in $RL_4$, only the memristors of the AND gate in $LIL_2$ changed their composite memristance (magenta curve), whereas those forming the other AND gate in $LIL_1$ did not change their state (red curve). This last change caused the corresponding OR gate memristor (yellow curve) in $LOL_1$ to change its state, as expected. As for circuit area, the sum-of-products implementation occupies three cross-point cells, yielding a total circuit area of $27F^2$. Overall, all the simulation results confirm the correct operation of the logic circuits, and all current sneak-paths are apparently successfully mitigated in the proposed architecture.

## IV. Conclusions

This work presented a novel crossbar-based high-density memristor-transistor ReRAM architecture enabling the integration of

logic-in-memory using a logic design scheme for parallel and single-step logic computations that enables greatly reduced computation time (compared to other sequential logic schemes) and simplified logic operation with normally-off (thus more energy-efficient) dense ReRAM-based logic gates. The architecture is CMOS compatible, with transistors in the bottom layer only (front-end transistors). The twisted transistor gate nanowires and the proposed gating patterns address the current-sneak paths and permit the safe execution of an arbitrary number of logic operations in parallel on $9F^2$ heterogeneous cross-point cells, using the information stored in memory as input logic data.

## REFERENCES


[1] "International Technology Roadmap for Semiconductors (ITRS)," 2013. [Online]. Available: http://www.itrs.net/. [Accessed June 2014].
[2] L. O. Chua, "If it's pinched it's a memristor," *Semicond. Sci. Technol.*, vol. 29, no. 10, pp. 104001, 2014
[3] J. Y. Seok, *et al*., "A Review of Three-Dimensional Resistive Switching Cross-Bar Array Memories from the Integration and Materials Property Points of View," *Adv. Funct. Mater*. vol. 24, no.34, pp. 5316-5339, 2014
[4] H.-S. P. Wong, *et al*., "Metal-Oxide RRAM," *IEEE Proc.*, vol. 100, no. 6, pp. 1951-1970, 2012
[5] J. Mustafa and R. Waser, "A Novel Reference Scheme for Reading Passive Resistive Crossbar Memories," *IEEE Trans. Nanotechnol.*, vol. 5, no. 6, pp. 687–691, 2006
[6] M. M. Ziegler and M. R. Stan, "CMOS/nano co-design for crossbar-based molecular electronic systems," *IEEE Trans. Nanotechnol*., vol. 2, no. 4, pp. 217–230, 2003
[7] S. Paul and S. Bhunia, "A scalable memory-based reconfigurable computing framework for nanoscale crossbar," *IEEE Trans. Nanotechnol.*, vol. 11, no. 3, pp. 451-462, 2012
[8] E. Linn, R. Rosezin, S. Tappertzhofen, U. Bottger and R. Waser, "Beyond von Neumann-logic operations in passive crossbar arrays alongside memory operations," *Nanotechnology*, vol. 23, no. 305205, 2012
[9] R. K. Cavin, P. Lugli, and V. Zhirnov, "Science and Engineering Beyond Moore's Law," *IEEE Proc*., vol. 100, Special Centennial Issue, pp. 1720 - 1749, 2012
[10] M. Prezioso, *et al*., "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521 (7550), pp. 61–64, 2015
[11] G. Indiveri, and S.-C. Liu, "Memory and Information Processing in Neuromorphic Systems," *IEEE Proc*., vol. 103, no. 8, pp. 1379 - 1397, 2015
[12] S. Saighi, *et al*., "Plasticity in memristive devices for spiking neural networks," *Front. Neurosci*., vol. 9 (51), March 2015
[13] J. J. Yang, D. B. Strukov and D. R. Stewart, "Memristive devices for computing," *Nat. Nano*., vol. 8, pp. 13–24, 2013
[14] I. Vourkas and G. C. Sirakoulis, "A Novel Design and Modeling Paradigm for Memristor-based Crossbar Circuits," *IEEE Trans. Nanotechnol*., vol. 11, no. 6, pp. 1151-1159, 2012
[15] S. Balatti, S. Ambrogio, and D. Ielmini, "Normally-off Logic Based on Resistive Switches—Part I: Logic Gates," *IEEE Trans. Electron Devices*, vol. 62, no. 6, pp. 1831-1838, 2015
[16] G. Ligang, F. Alibart and D. B. Strukov, "Programmable CMOS/memristor threshold logic," *IEEE Trans. Nanotechnol*., vol. 12, no. 2, pp. 115-119, 2013
[17] S. Kvatinsky, *et al*., "MAGIC – Memristor Aided LoGIC," *IEEE Trans. Circuits Syst. II, Expr. Briefs*, vol. 61, no. 11, pp. 895 - 899, 2014
[18] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser and E. G. Friedman, "MRL - Memristor Ratioed Logic," 2012 *Int. Workshop on Cellular Nanoscale Netw. and Appl*. (*CNNA*), Turin, Italy, Aug. 29 – 31, pp. 1 - 6
[19] 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, p. 873–876, 2010
[20] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, and G. Ch. Sirakoulis, "Boolean logic operations and computing circuits based on memristors," *IEEE Trans. Circuits Syst. II Expr. Briefs*, vol. 61, no. 12, pp. 972-976, 2014
[21] E. Lehtonen, J. Tissari, J. Poikonen, M. Laiho, and L. Koskinen, "A cellular computing architecture for parallel memristive stateful logic," *Microelectronics Jour*., vol. 45, no. 11, pp. 1438-1449, 2014
[22] K. C. Rahman, D. Hammerstrom, Y. Li, H. Castagnaro, and M. A. Perkowski, "Methodology and Design of a Massively Parallel Memristive Stateful IMPLY Logic based Reconfigurable Architecture," *IEEE Trans. Nanotechnol*. (2016), in press
[23] I. Vourkas, A. Batsos, and G. Ch. Sirakoulis, "SPICE modeling of nonlinear memristive behavior," *Int. J. Circ. Theor. Appl*., vol. 43, no. 5, pp. 553–565, 2015
[24] Y. V. Pershin and M. Di Ventra, "Memory effects in complex materials and nanoscale systems," *Advances in Physics*, vol. 60, no. 2, pp. 145-227, 2011
[25] B. Chen, *et al*., "Highly Compact (4F2) and Well Behaved Nano-Pillar Transistor Controlled Resistive Switching Cell for Neuromorphic System Application," *Sci. Rep.*, vol. 4 (6863), 2014
[26] H.-Y. Chen, S. Yu, B. Gao, P. Huang, J. Kang, and H.-S. P. Wong, "HfOx Based Vertical Resistive Random Access Memory for Cost-Effective 3D Cross-Point Architecture without Cell Selector," *IEEE Int. Electron Devices Meeting* (*IEDM*), San Francisco, CA, 10-13 Dec. 2012, pp. 20.7.1 - 20.7.4
[27] B. Gao, *et al*., "3-D Cross-Point Array Operation on AlOy/HfOx-Based Vertical Resistive Switching Memory," *IEEE Trans. Electron Devices*, vol. 61, no. 5, pp. 1377 - 1381, 2014
[28] C. Jung, J. Choi and K. Min, "Two-step write scheme for reducing sneak-path leakage in complementary memristor array," *IEEE Trans. Nanotechnol*., vol. 11, no. 3, p. 611–618, 2012
[29] E. Linn, R. Rosezin, C. Kugeler and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nat. Mater.*, vol. 9, no. 5, p. 403–406, 2010
[30] Y.-C. Chen, H. Li, W. Zhang, and R. E. Pino, "The 3-D Stacking Bipolar RRAM for High Density," *IEEE Trans. Nanotechnol*., vol. 11, no. 5, pp. 948-956, 2012
[31] G. Sassine *et al*., "Interfacial versus filamentary resistive switching in TiO2 and HfO2 devices," *J. Vac. Sci. Technol*. vol. B34 (012202), 2016
[32] I. Vourkas, and G. Ch. Sirakoulis, "Memristor-Based Nanoelectronic Computing Circuits and Architectures," (1st Ed). Springer International Publishing series: *Emergence, Complexity, Computation*, vol. 19, 2015, DOI: 10.1007/978-3-319-22647-7
[33] G. Papandroulidakis, I. Vourkas, G. Ch. Sirakoulis, and A. Rubio, "Heterogeneous Memristive Crossbar for In-Memory Computing," 2015 *Int. Conf. on Memristive Systems* (*MEMRISYS*), Paphos, Cyprus, November 08-10, pp. 1–2
[34] P. Pouyan, E. Amat, and A. Rubio, "Reliability challenges in design of memristive memories," 5*th European Workshop on CMOS Variability* (*VARI*), Palma de Mallorca, Spain, Sept. 29-Oct. 1 2014, pp. 1-6
[35] R. Naous, M. Al-Shedivat, and K. N. Salama, "Stochasticity Modeling in Memristors," *IEEE Trans. Nanotechnol*., vol. 15, no. 1, pp. 15 - 28, 2016


**Georgios Papandroulidakis** received his M. Eng. degree in electrical and computer engineering (ECE) from Democritus University of Thrace (DUTh), Greece, in 2015. He is currently pursuing an M.Sc. in the ECE Dept. at DUTh under the supervision of Associate Prof. G. Ch. Sirakoulis. His thesis focuses on the design and simulation of novel nano-electronic

circuits and architectures comprising memristive elements with the use of simulation environments with integrated circuits emphasis (SPICE).

**Ioannis Vourkas** (S'12-M'16) received his M. Eng. diploma and Ph.D. in electrical and computer engineering (ECE) from Democritus University of Thrace (DUTh), Xanthi, Greece, in 2008 and 2014, respectively. He is currently with the Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile, working as a post-doctoral researcher and the principal investigator for the project CONICYT FONDECYT Postdoctorado No. 3160042 (2016).
In 2006 he was a summer intern at the Cultural & Educational Technology Institute in Xanthi, Greece. In 2009 he worked via the European vocational training program "Leonardo Da Vinci" at the Institute of Industrial & Control Engineering at Polytechnic University of Catalonia (UPC), Spain. His current research emphasis is novel nano-electronic circuits and architectures comprising memristors. His research interests include unconventional computing, software and hardware aspects of parallel complex computational (bio-inspired) circuits and systems, and cellular automata and their applications.
Dr. Vourkas received the 2014 Best Ph.D. Student award at ECE-DUTh and was runner up for the Best Ph.D. Research Award in the Ph.D. Forum Session of the 2013 IFIP/IEEE VLSI-SoC Conference in Istanbul, Turkey. He was an invited tutorial speaker at the 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS) in Cairo, Egypt. He was a scholar at the Greek BODOSSAKI Foundation (2011 to 2014).

**Angel Christian Abusleme Hoffman** (M'01) was born in Santiago de Chile in 1975. He earned his diploma and M.Sc. in electrical engineering from Pontificia Universidad Católica de Chile (PUC Chile) in 2000, and his Ph.D. in microelectronics from Stanford University, CA, USA, in 2011.
From 2001 to 2003 he was an Instructor at PUC Chile. From 2003 to 2011 he was a Ph.D. student at Stanford University. From 2011 to 2015 he was an Assistant Professor at PUC Chile. He is currently an Associate Professor of Electrical Engineering at PUC Chile. His research has focused on instrumentation circuits for particle physics experiments. He is a member of the ATLAS, FCAL and CLICdp projects. He is also a sponsoring researcher in the project CONICYT FONDECYT Postdoctorado No. 3160042 (2016).
Dr. Abusleme is a Professional Engineer in Chile and has received numerous teaching awards from his institution.

**Georgios Ch. Sirakoulis** (M'95) received his M. Eng. diploma and Ph.D. in electrical and computer engineering (ECE) from Democritus University of Thrace (DUTh), Greece, in 1996 and 2001, respectively.
He is a Tenured Associate Professor in the ECE Department of DUTh (2008-today). He has published over 200 technical papers, guest-edited eleven (11) special issues, co-edited five (5) books and co-authored fifteen (15) book chapters. He is EUROPRACTICE representative for DUTh, and he has served as a member of the EU IDEAS program. He has participated as a PI in more than 20 scientific programs and projects funded by the Greek government and industry and the European Commission. His current research interests include emergent electronic circuits and systems, memristors, green and unconventional computing, cellular automata theory and applications, complex systems, bioinspired computation/biocomputation, modeling and simulation.
Dr. Sirakoulis is a member of the IEEE Circuits and Systems (CAS) Society, IEEE Computer Society, IEEE Electron Devices, IEEE Circuits and Systems Education and Outreach TC, the Association for Computing Machinery (ACM), and the Technical Chamber of Greece (TEE). In 1996, he received the Prize of Distinction from TEE for his diploma thesis. He was a founding member and vice president of the IEEE Student Branch of Thrace from 2000 to 2001. He serves as an associate editor of the *Microelectronics Journal*, *Integration,* the *VLSI Journal*, the *International Journal of Parallel, Emergent and Distributed Systems*, the *Journal of Cellular Automata*, the *Journal of Applied Mathematics*, and *Recent Advances in Electrical and Electronic Engineering*. He was General Co-Chair of the 2014 International Conference on Cellular Automata for Research and Industry (ACRI), and several other conferences, including ACRI 2012, the 5th Pan-Hellenic ECE Students Conference, Special Sessions in the 2015 IEEE International Conference on Nanotechnology, the 2015 Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), PDP 2014, the 2014 HiPEAC Computing Systems Week, the 18th and 20th Pan-Hellenic Conference in Informatics, and the 20th IEEE International Conference on Electronics, Circuits, and Systems, among others.

**Antonio Rubio** (SM'96) received his M.Sc. and Ph.D. from the Industrial Engineering Faculty, Polytechnic University of Catalonia (UPC), Barcelona, Spain. He has been an Associate Professor in the Electronic Engineering Department, Industrial Engineering Faculty, UPC, and a Professor in the Physics Department, Balearic Islands University. He is currently a Professor of electronic technology in the Telecommunication Engineering Faculty, UPC.
His research interests include very large scale integration design and test, device and circuit modeling, high-speed circuit design, and new emerging nanodevices and nanoarchitectures.