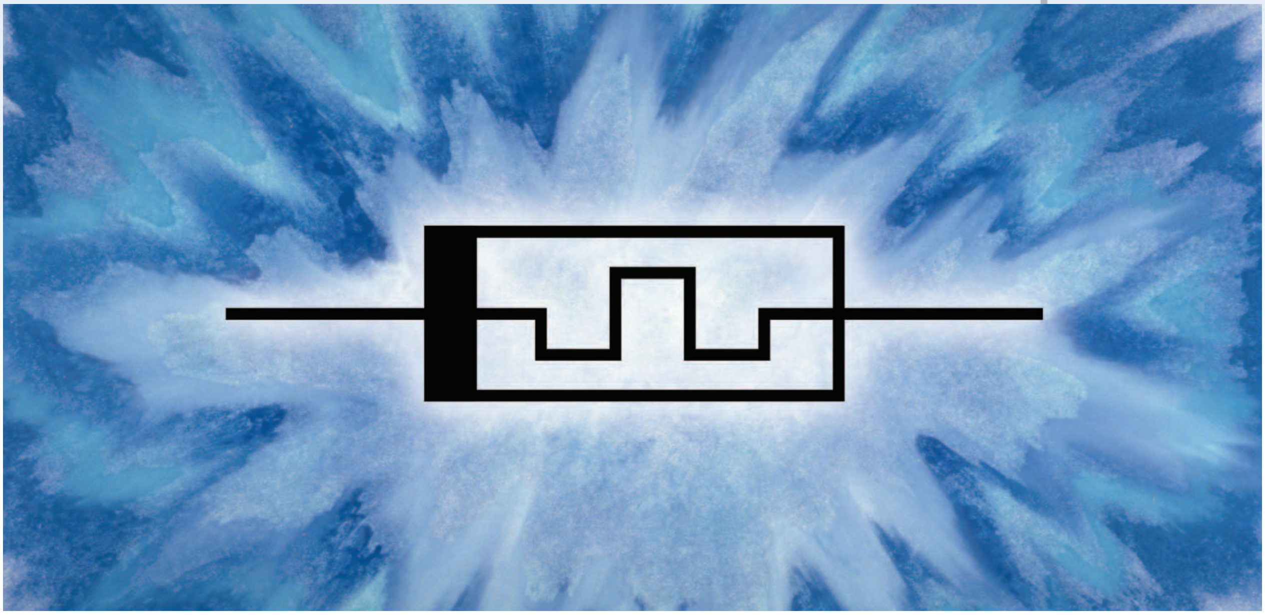
Emerging Memristor-Based Logic Circuit Design Approaches: A Review

Ioannis Vourkas and
Georgios Ch. Sirakoulis



©COREL

Abstract

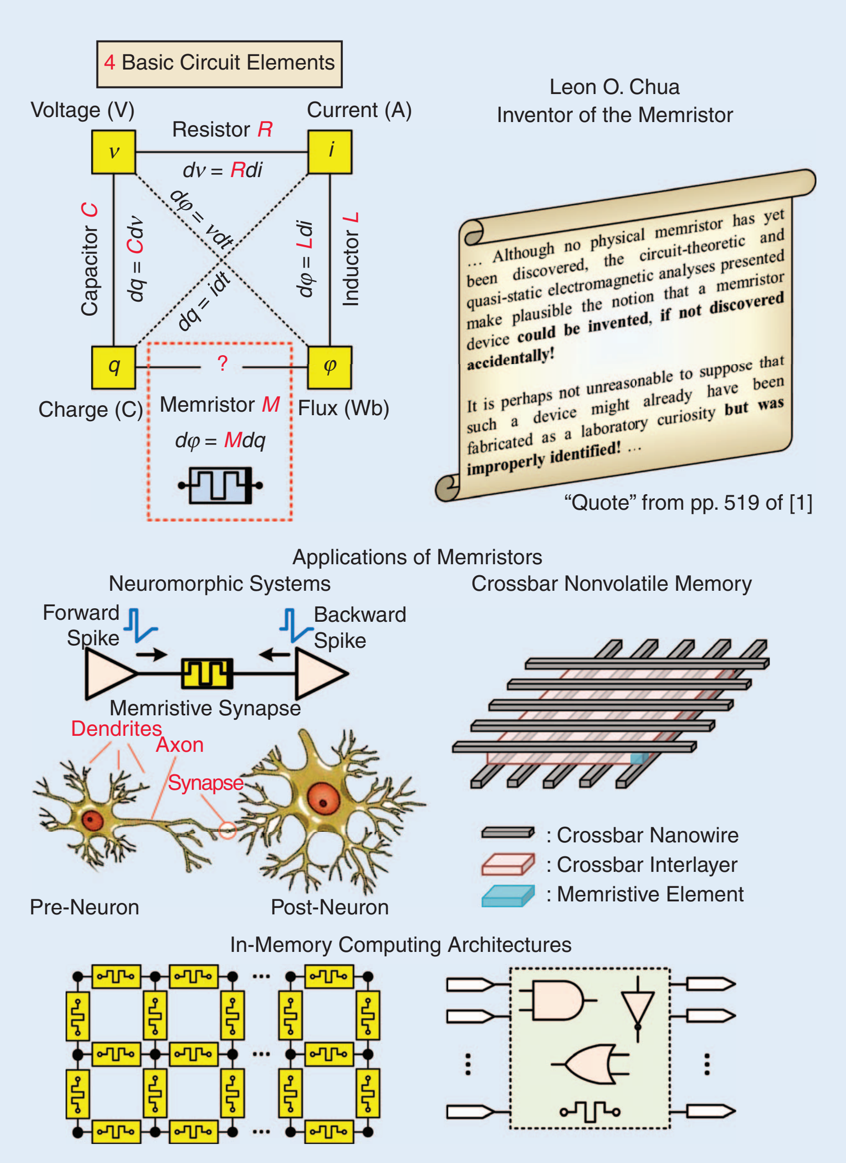
This article is a comprehensive review of the state-of-the-art of memristor-based logic circuit design concepts of the recent literature. Amongst all the identified circuit design approaches, those discussed here are all based on collective memristive dynamics and share a number of common characteristics which facilitate their comparison. The focus is on the evolution of the memristor-based logic circuit design strategies from the early proposed sequential stateful logic up to most recently published design schemes which support parallel processing of the applied input signals. The main operational properties of all the selected computational concepts are presented in an accessible manner, aiming to serve as an informative cornerstone for students and scientists who wish to get involved in emerging memristive logic circuit research and development.

Digital Object Identifier 10.1109/MCAS.2016.2583673
Date of publication: 19 August 2016

I. Introduction

As a result of his preliminary work in nonlinear circuit theory, in 1971 Leon Chua made an interesting observation that led to his discovery of the *memristor* as a mathematical entity and as the fourth fundamental circuit element, joining the resistor, the capacitor, and the inductor [1], as summarized in Fig. 1. He mathematically explored the properties of this new nonlinear circuit element and found that it was essentially a “resistor with memory”, so he called it a memristor [2] (hereinafter used interchangeably with the term “memristive device” [3]). Predicting the properties of a new “missing” circuit element, based only on symmetry principles and

This work was partially supported by the CONICYT FONDECYT POSTDOCTORADO No. 3160042/2016 Chilean government research grant.



... Although no physical memristor has yet been discovered, the circuit-theoretic and quasi-static electromagnetic analyses presented make plausible the notion that a memristor device **could be invented, if not discovered accidentally!**

It is perhaps not unreasonable to suppose that such a device might already have been fabricated as a laboratory curiosity **but was improperly identified!** ...

Figure 1. Memory-resistor (memristor): the fourth fundamental circuit element, proposed by L. Chua in 1971 [1], is much more than just memory.

without any experimental observation, was absolutely revolutionary. Chua himself declared in his 1971 paper that it was not obvious that a physical analog of such circuit element existed (the note in Fig. 1 is a summary of what is originally stressed in [1]). However, the memristor attracted most of attention only after 2008, when the first "modern" [4] practical memristor implementation was announced by the research group of Stanley Williams at the Hewlett-Packard Laboratories (HP Labs). For the first time the Chua's theory was then connected with practice, thus originating intense research activity in this new and promising research field [5].

This invention of the first TiO_2-based bipolar memristor by HP Labs was soon followed by the identification of more and more material compounds which exhibit memristive properties and can be the basis of memristive devices. A wide variety of such materials have been studied so far and they can be roughly categorized as transition metal oxides (such as NiO, CuO, ZrO_2, TiO_2, and TaO_x) [6], perovskites [7], and solid-state electrolytes [8]. Nevertheless, even though most of the proposed devices use such materials as the functional resistive switching layer, not all of these materials are compatible with current integrated circuit industry infrastructure, something that led to a focused interest in silicon and silicon oxide as well [9]. Generally, the favorable performance merits of memristors concern nonvolatility, fast switching speed, small area and low energy dissipation [10], [11]. Nevertheless, the choice of materials in memristor devices is truly dependent on the application, as clearly some features of specific memristive compounds may be beneficial in some context and disadvantageous otherwise [12]. Therefore, it is imperative that further materials science studies are conducted to identify the optimal memristors for various technical requirements. In the same context, well-defined and effective mathematical modeling of the behavior of memristive devices is a requirement for a deeper understanding of their dynamics and exploitation of their unique properties. So, a great effort has been made by the research community in this direction, with some of the most noteworthy models described in [13], [14].

Memristors are definitely likely to play a leading role in next generation low-power and high-density memory

Ioannis Vourkas is with the Centro de Investigación en Nanotecnología y Materiales Avanzados (CIEN-UC), Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile (e-mail: iovourkas@uc.cl). Georgios Ch. Sirakoulis is with the Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece (e-mail: gsirak@ee.duth.gr).

While reaching and exploring the post-CMOS and beyond-Von Neumann era, one thing is for sure; that memristor-based logic design and in-memory computing will attract more and more attention, giving rise to an evergrowing number of different computing approaches.

systems [15], something reflected in the planning and early commercialization of memristive memory (resistive RAM - ReRAM) products [16]–[18]. However, the incredible potential of memristors as memory devices is not necessarily the most interesting thing about them; in short, memristors can be much more than just memory devices. In fact, the characteristic pinched current-voltage hysteresis feature, underlying the operation of memristors, indicates the potential of being used in a continuous operational mode as part of an analog computing paradigm, suitable for solving complex real world problems [19]. Relevant applications concern bio-inspired neuromorphic processing hardware (HW) of greatly reduced power requirements thanks to the memristive synaptic connection elements [20]. In this sense, Chua *et al.* [21] presented that the potassium ion-channels and the sodium ion-channels, distributed over the entire length of the axons of our neurons, are indeed locally-active memristors that exhibit all the respective fingerprints [22]. Building brain-like, massively parallel, compact systems in 3D HW has been challenging due to the lack of a compact electronic device which mimics the biological synapse, for these elements are most numerous (human brain consists of $\approx 10^{11}$ neurons and an extremely large number of synapses $\approx 10^{15}$) [23]. An experimental implementation of transistor-free memristive crossbar, with low enough device variability to allow operation of integrated neural networks, was recently demonstrated as an important step towards large and complex memristive neuromorphic networks [24]. On the other hand, using memristors as discrete storage elements in a digital platform within the still dominant digital computers realm, could be a more economically viable path to maximize the benefits from digital computing in *beyond-Von Neumann system architectures*, where memory and processing co-exist [25] (see application summary in Fig. 1). Memristors provide an unconventional computation framework which combines information processing and storage in the memory itself; i.e. the major distinction from the present day's computing technology.

This justifies the ever-growing attention gained by unconventional memristive computing and logic circuits [26]–[30]. Memristor-based logic circuits open new pathways for the exploration of advanced computing architectures as promising alternatives to conventional integrated circuit technologies which are facing serious challenges related to continuous scaling [10], [25], [31]. However, up to now no standard logic circuit design methodology exists. So, it is not immediately clear what kind of computing architectures would in practice benefit the most from the computing capabilities of memristors. Related work on memristor-based logic/computational circuits could be classified as follows:

- *Hybrid memristor/CMOS*: combination of memristors and CMOS components in Boolean logic [32], [33] and threshold logic computations [26], [34], [35];
- *Material implication* (*IMPLY*): computation of Boolean functions using *imply* and *reset* logic operations [36]–[39];
- *All memristive*: computation of Boolean functions in memristor-only circuits [40]–[42];
- *Programmable interconnects*: logic operations in crossbar arrays relying on programmable nanowire interconnections [43]–[47];
- *Network-based computations*: massively parallel computations within array-like structures which accommodate networks of memristive components [48]–[52].

All the aforementioned categories constitute novel design paradigms which make possible fundamental logic operations as well as sophisticated computations. Motivated by such an explosive growth of memristive logic-related publications, in this article we approach this emerging scientific area aiming to highlight current trends and future perspectives, while motivating for further research on circuit design strategies that comply with emerging memristive device technologies. We provide a comprehensive review of the most recognized memristive logic design concepts found so far in the current literature, which could serve as an informative cornerstone for students and scientists who wish to get involved in memristive logic circuit research. The discussed design concepts are based on collective memristive dynamics and share a number of common characteristics which facilitates their comparison. Moreover, the focus is on the evolution of the memristor-based logic circuit design strategies from the early proposed sequential stateful logic [36] up to most recently published design schemes which support parallel processing of input signals [33], [42]. The main operational characteristics of every discussed computational concept are presented in an accessible manner, whereas their advantages, limitations and drawbacks are also highlighted.

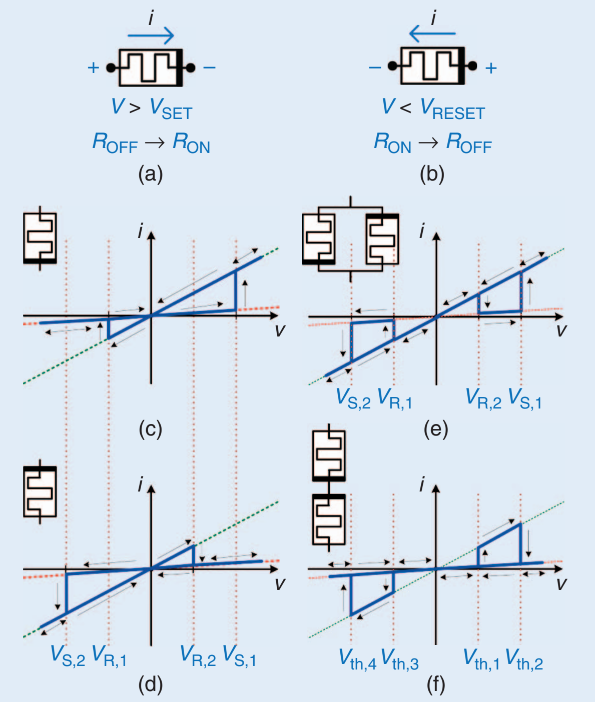


Figure 2. (a, b) Voltage-controlled polarity-dependent resistance switching of bipolar memristors. (c - f) Qualitative current-voltage ($i - v$) characteristics of memristors with opposite polarities and of their serial/parallel combination.

The rapid progress in the advancements of memristive technology, together with the R&D efforts in novel computer architectures, as indicated by the groundbreaking announcement of “the Machine” by HP [53], prove the ever-increasing interest and active involvement of industry-leading companies in the production of memristor-related future products and pioneering memristive computing architectures. Therefore, gathering the *state-of-the-art of memristive logic circuit design* approaches in a single review article is expected to further facilitate the incorporation of memristors in currently established logic circuit architectures, as well as in future electronic system design.

II. Switching Dynamics of Threshold-Type Memristors

Operation of almost all of the memristor-based logic circuits, discussed in this work, is based on the collective dynamics of multiple connected threshold-type voltage-controlled bipolar memristors. Understanding the overall circuit behavior requires comprehending the switching dynamics of individual memristors and of memristive compositions which comprise a network of devices with either the same or opposite polarities. We assume only bipolar devices, i.e. memristors which require both polarities to switch their state. Hereinafter we refer to a memristor being forward/reversely polarized (FPM/RPM) when the voltage is applied to the top/bottom terminal with the bottom/top terminal being grounded; bottom terminal is always denoted by the black thick line in the circuit schematics. For the employed devices we assume asymmetric voltage thresholds and, when not specifically indicated, the following initialization states: {RPM, FPM} = $\{R_{\mathrm{ON}}, R_{\mathrm{OFF}}\}$.

Fig. 2(a, b) shows the polarity-dependent resistance switching of memristors. We assume that the memristance will decrease when the memristor is forward-biased and it will increase when it is reversely-biased. For threshold-type switching memristors we consider that the resistance change-rate is very slow below (very fast above) a voltage threshold (namely V_{SET} or V_{RESET}), which is viewed as the minimum voltage required to impose a change on the memristance of the device.

In Fig. 2(c - f) we show qualitatively the current-voltage ($i - v$) characteristics of memristors with opposite polarities and of their serial/parallel combination under AC applied voltage. We assume that the current is piece-wise linear with the applied voltage, i.e. the memristor has linear ON and OFF states [54], [55]. Memristors with opposite polarities generally demonstrate reversed behavior to the applied signals. According to Fig. 2(c), when a positive voltage, applied to a FPM, reaches its V_{SET} threshold ($V_{\mathrm{S},1}$), the device switches to its low resistive state (R_{ON}). Then, an ohmic behavior is observed until a V_{RESET} voltage threshold ($V_{\mathrm{R},1}$) is reached and the device returns to the high resistive state (R_{OFF}). The $i - v$ graph of a RPM, shown in Fig. 2(d), is symmetric to that of a FPM and has the opposite voltage thresholds $\{V_{S,2}, V_{R,2}\}$.

Fig. 2(e) illustrates the composite $i - v$ response of two reciprocal memristors connected in parallel. When a positive applied voltage reaches the reset threshold of the RPM ($V_{R,2}$) its state is changed and the total current drops. Next, when the voltage exceeds the set threshold ($V_{S,1}$) of the FPM it switches to R_{ON} and the total current rises again. Afterwards the composite device exhibits an ohmic behavior unless a negative voltage is applied. This behavior is opposite to that of the anti-serially connected memristors (forming a complementary resistive switch—CRS [56]) presented in Fig. 2(f). In the latter, the voltage thresholds $V_{\mathrm{th},1} - V_{\mathrm{th},4}$ cannot be formerly known exactly because the memristors here form a voltage divider; the voltage drop over each element depends on the total applied voltage, on the internal states of the devices, and on their particular switching characteristics [15]. In such composite memristive structures there is no need to particularly access each memristive device so as to adjust its memristance. In fact, regardless of the current state of the devices, resetting is done with a negative applied voltage which exceeds the leftmost voltage threshold shown in the corresponding $i - v$ graph. In either parallel or serial connection, such a negative voltage pulse will eventually reset all memristors to the initial boundary resistive states (either R_{ON} or R_{OFF}) depending on their polarity.

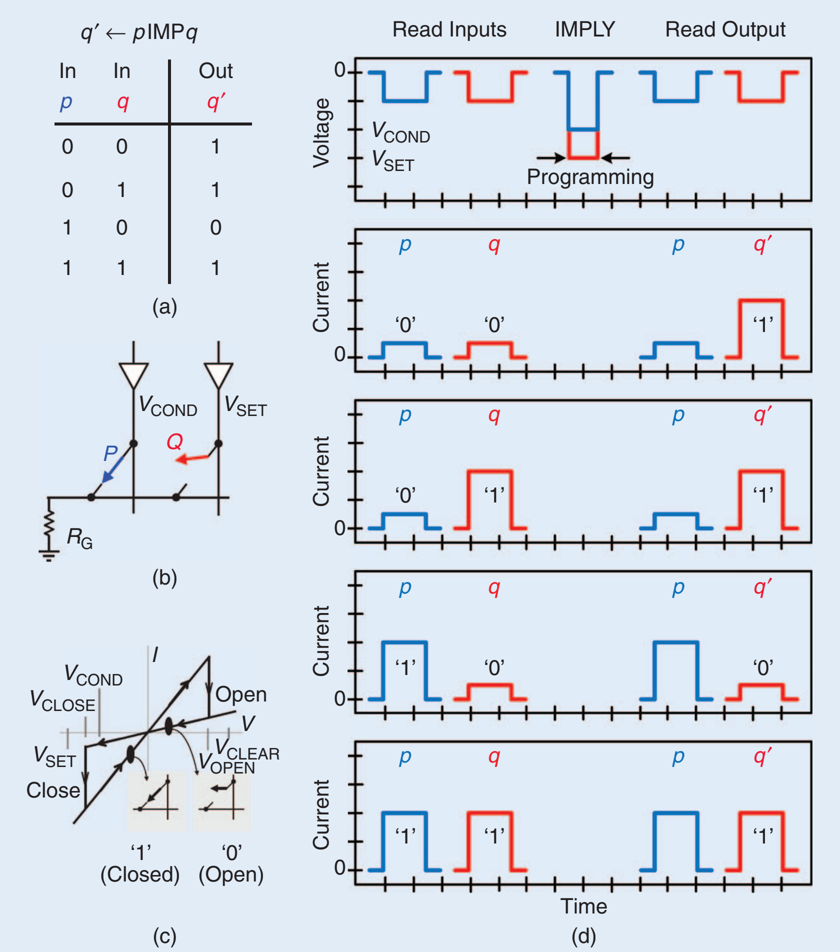
An outstanding feature that appears for both the series and the parallel connection of reciprocal memristors is the perfectly symmetric $i - v$ curve which resembles a "truncated" Ohm's Law. However, if the devices are all placed with the same polarity, their overall behavior resembles that of a memristor of the same polarity but whose properties combine the properties of the individual devices [51], [57]. E.g., two FPMs in series will both switch their states simultaneously when the applied voltage exceeds the sum of the individual thresholds which are: $2 \times \{V_{\mathrm{RESET}}, V_{\mathrm{SET}}\}$, whereas the composite memristance will take values within the $2 \times [R_{\mathrm{ON}}, R_{\mathrm{OFF}}]$ range. Similarly, having two memristors with the same polarity in parallel reproduces the individual memristive behavior while achieving higher total current values because of the lower composite memristance which varies within the $\frac{1}{2} \times [R_{\mathrm{ON}}, R_{\mathrm{OFF}}]$ range.

III. Popular Memristive Logic Circuit Design Concepts

A. Material Implication (IMPLY) Memristor Logic

In 1910 Whitehead and Russell [58] described four fundamental logic operations. Three of them were the well-known in the electrical engineering and computing communities: AND, OR, and NOT, which form a computationally complete set. However, there was another operation which Russell named "material implication", p IMP q (i.e. "p implies q" or "if p, then q") whose truth table is shown in Fig. 3(a). The IMP and FALSE operations (where the FALSE operation always yields logic '0') form a computationally complete logic basis. In [36] it was first shown by the HP Labs group that material implication is naturally realized in a simple circuit combining a resistor with two memristors, as shown in Fig. 3(b) where memristors are shown as two-state switches. The logic state variable of this computation framework is the memristance of the devices, which is why the IMPLY logic operation was characterized in the literature as "stateful".

Memristive IMPLY logic relies on using threshold-type switching memristors; the devices undergo a rapid memristance change if the applied voltage exceeds either the V_{OPEN} (i.e. V_{RESET}) or V_{CLOSE} (i.e. V_{SET}) voltage thresholds. We illustrate the qualitative $i - v$ graph of a RPM in Fig. 3(c) where we define the R_{CLOSED} (R_{ON}) state to represent logic '1' and the R_{OPEN} (R_{OFF}) state logic '0', respectively. The memristors are driven by tri-state voltage drivers. The latter provide a high-impedance output state when not driven. A given memristor may be "set" (assigned logic '1') by applying a voltage V_{SET} (this is the operation TRUE). Similarly, the device may be "cleared" (assigned logic '0') by applying a voltage V_{CLEAR} (i.e. the operation FALSE). Also, an auxiliary voltage V_{COND}, whose magnitude is



$q' \leftarrow p\mathrm{IMP}q$

In p	In q	Out q'
0	0	1
0	1	1
1	0	0
1	1	1

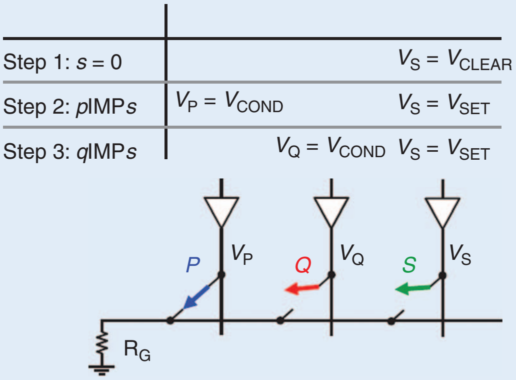
(a)

(b)

(c)

(d)

Figure 3. Illustration of the IMP operation for all valid input combinations of variables p and q. (a) Truth table of $q' \leftarrow p$ IMP q. (b) The circuit schematic comprising two memristors and a load resistor. (c) The qualitative memristive electrical characteristics with abrupt voltage thresholds. (d) Qualitative $v - t$ and $i - t$ graphs of memristive IMP; the blue (red) curves qualitatively demonstrate the applied voltages and the corresponding currents read at memristor P (Q) before and after the IMP voltage pulses [36].



Step 1 $s = 0$	Step 2 $s' \leftarrow p$IMP$s$		Step 3 $s'' \leftarrow q$IMPs'			Step 1, 2, 3 $s'' \leftarrow p$NANDq	
s	p s	s'	q s'	s''	=	p q	s''
0	0 0	1	0 1	1		0 0	1
0	0 0	1	1 1	1		0 1	1
0	1 0	0	0 0	1		1 0	1
0	1 0	0	1 0	0		1 1	0

(a)

p AND q = (p IMP (q IMP 0)) IMP 0

p OR q = (p IMP 0) IMP q

p NAND q = p IMP (q IMP 0)

p NOR q = ((p IMP 0) IMP q) IMP 0

NOT q = q IMP 0

NOT p = p IMP 0

p XOR q = (p IMP q) IMP ((q IMP p) IMP 0)

(b)

Figure 4. (a) $s \leftarrow p$ NAND q performed as a sequential logic operation with three memristors. The sequence of voltages applied to obtain the NAND operation with P, Q and S, the circuit schematic, and the truth tables showing the equivalence of the sequence of operations. (b) Several binary Boolean functions of two logic variables using only IMP and/or FALSE operations.

smaller than V_{SET}, is used to facilitate conditional state-switching.

The key to perform a memristive IMP operation is to understand the conditional toggling property. The operation $q \leftarrow p$ IMP q is implemented by simultaneously applying a V_{SET} pulse to Q and a V_{COND} pulse to P to execute a conditional switching operation. If V_{SET} is applied to Q alone, it executes the unconditional operation $q \leftarrow 1$, while the V_{COND} pulse applied to P alone implements $p \leftarrow p$. When applied together, though, the two pulses interact through P, Q and the load resistor R_G to cause state-changes depending on the current states of p and q. If P is in R_{OFF} state ($p =$ '0') it has little influence on the voltage divider formed by Q and R_G, thus Q is set ($q \leftarrow$ '1') while P is left unchanged ($p \leftarrow p$). However, if P is in R_{ON} state ($p =$ '1'), the V_{COND} pulse on P "shorts out" the voltage divider and both P and Q are not affected ($q \leftarrow q, p \leftarrow p$). The selection of R_G is important to the outcome of the operation. Its resistance is chosen such that $R_{ON} < R_G < R_{OFF}$. When both P and Q are in R_{OFF}, then V_{COND} and V_{SET}, respectively, drop mainly across each device because $R_G < R_{OFF}$. This leaves P in R_{OFF} but switches Q to R_{ON}. However, if P is in R_{ON}, the voltage on the common node is $\approx V_{COND}$ since $R_{ON} < R_G$, hence the voltage drop across Q is almost equal to $V_{SET} - V_{COND}$, which leaves Q in R_{OFF}.

In Fig. 3(d) we demonstrate the memristive IMP operation via qualitative $v - t$ and $i - t$ graphs, which correspond to the experiments published in [36]. First, P and Q memristors were initialized to the desired states by applying appropriate voltage pulses, V_{SET} or V_{CLEAR}. The initial states were verified by applying a small reading voltage and measuring the resulting current. Then V_{COND} and V_{SET} were simultaneously applied to memristors P and Q, respectively. The corresponding initial memristances p and q are the inputs of the logic gate, whereas the output is the final memristance of Q (the result is written in the logic state q). Note that the memristance of both memristors changes during operation, i.e. the computation is destructive for both inputs. The qualitative conditional toggling results of Fig. 3(d) are in line with the corresponding truth table of the memristive $q \leftarrow p$ IMP q in Fig. 3(a).

A major disadvantage of memristive IMPLY logic is the necessity to perform lengthy sequences of stateful logic operations in order to synthesize a given Boolean function. According to [38], where a multi-input implication operation was introduced with complementary representation of input variables, up to $2^{n-1} + 1$ computational steps are required for the synthesis of an arbitrary n-input Boolean function. This drawback is particularly seen in Fig. 4(a) where the logic operation $s \leftarrow p$ NAND q is sequentially performed using three memristors. In the same fashion, the necessary sequential computational steps for several other binary Boolean functions on two logic variables are given in Fig. 4(b). Apparently, the practical utility of the memristive IMPLY logic design scheme requires operations to become as parallel as possible. To this end, further research is still necessary and researchers are focusing on improvements and modifications at the circuit and/or architecture level [37], [59].

B. MRL—Memristor-Ratioed Logic

Integrating memristors and CMOS to perform logical operations could be proven beneficial, given that the memristors could be fabricated within the CMOS metal layers, thus significant physical integration area could be saved while increasing the device density [44], [60]. To this end, Kvatinsky *et al.* [33] proposed a hybrid CMOS-memristive logic family which they called Memristor "Ratioed" Logic (MRL). MRL constitutes a combination of memristors and CMOS transistors which has the potential to save significant amount of chip area as the number of circuit (logic) inputs increases. In such CMOS-compatible logic family, AND and OR logic gates are based on memristors whereas a CMOS NOT gate is used to provide a complete logic gate-set and to restore degraded signals.

This logic family uses the programmable resistance of linear memristors, rather than threshold-type, to compute Boolean AND/OR functions with the voltage being the logic state variable. Both AND and OR logic gates consist of two memristors connected in series with opposite polarity, as shown in Fig. 5(a, b). The only difference between the two gates consists in the polarity of the memristors with respect to where the inputs are applied. The output node is the common node of the connected memristors, whereas each input signal is applied to the floating terminal of every memristor.

The overall behavior of both gates resembles that of a CRS [56]. Owing to their different polarity, the memristors tend to switch their states in a reciprocal way which depends on the applied input signals. Both logic gates react similarly to identical inputs (both being either logic '1' or logic '0'). Since the voltage drop between inputs is zero, the voltage at the output node follows the input voltage. However, when the inputs are different there is current flowing from the high voltage terminal (where the '1' is applied) to the grounded terminal (where the '0' is applied), thus potentially affecting the memristance of the devices.

This case is shown in Fig. 5(c) for an OR logic gate. Assuming initially $R_1 = R_{\mathrm{OFF}}$ and $R_2 = R_{\mathrm{ON}}$, at the end of the computational process the memristors have changed their initial states. For the AND logic gate the different polarities have as a result the state of each memristor to switch in the opposite manner, as shown in Fig. 5(d, f). Assuming a high memristance ratio $R_{\mathrm{OFF}}/R_{\mathrm{ON}}$, the output voltage of the logic gates is determined by the voltage divider across both memristors. However, this voltage divider impacts the level of the output signal. Although the signal degradation is minor when $R_{\mathrm{OFF}} \gg R_{\mathrm{ON}}$, for cascaded logic gates this degradation accumulates and may become significant [33]. Therefore, signal restoration is occasionally required so that the output signals are high enough to be fed to the following gate-stage. The number of logic inputs for both gates can be extended by connecting more memristors to a common node, similarly to diode logic [61], on which the MRL concept is based.

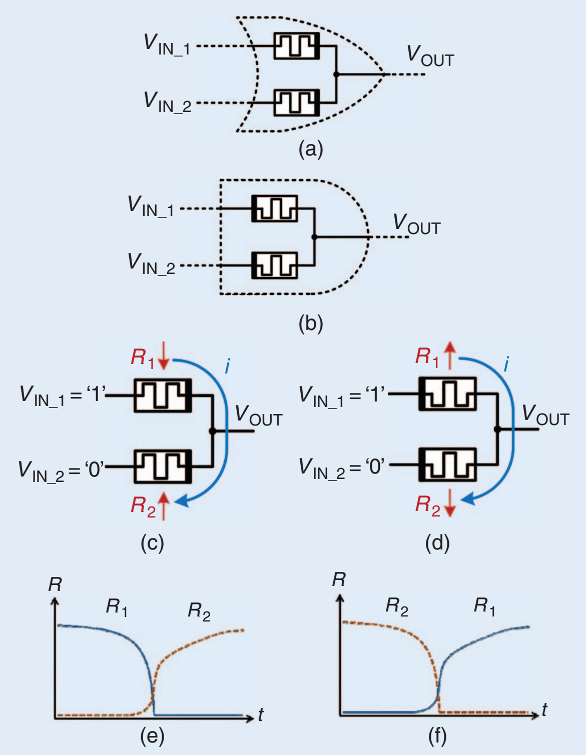


Figure 5. (a, b) Circuit schematic of the MRL OR/AND logic gates [33]. (c, d) Circuit behavior when different input logic signals are applied. (e, f) Qualitative time-evolution of the memristance of the two memristors for the case shown in (c, d).

In MRL, the topology of the circuit determines the logical function, whereas the computational process is composed of only a single step and the result is independent of the initial state of the memristors. Nevertheless, MRL provides only a restricted basic set of possible Boolean logic operations, while it relies on using linear memristors, which however are much slower than nonlinear (threshold-type) devices [10]. Therefore, the switching time is dependent on the applied voltage. A relatively low level of the applied input voltage further increases the delay time, whereas it is possible that the memristors will not fully switch their states if the input voltages are not applied for a sufficiently long time. In such case it would be difficult to distinguish between the possibly different output voltage levels corresponding to the different logic states.

C. MAGIC—Memristor-Aided Logic

In an attempt to simplify the logic circuit operation and decrease the circuit complexity, required by earlier proposed sequential logic concepts, while allowing for a more stable evaluation of the logic functions, Kvatinsky

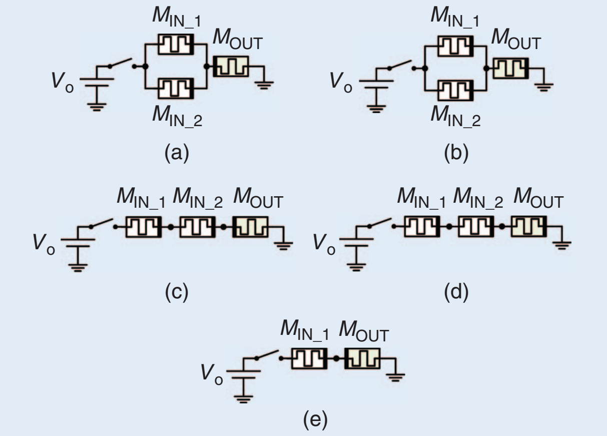


Figure 6. (a) Circuit schematic of a two-input (a) NOR, (b) OR, (c) NAND, (d) AND, and (e) a NOT (CRS) MAGIC logic gate [41].

et al. in [41] proposed a memristor-only sequential logic family which they called MAGIC, for "Memristor Aided loGIC". The MAGIC approach does not require any complicated circuit structure, whereas the stable output evaluation is achieved by applying a single input voltage pulse at the gateway of the logic circuits.

The corresponding circuit schematics of the universal MAGIC gate set, namely AND, OR, NOT, NAND, and NOR, is shown in Fig. 6. MAGIC requires only threshold-type switching memristors within the logic gates. Similarly to the IMPLY logic, the logic state variable is the memristance of the devices. Specifically, the high (R_{OFF}) and low (R_{ON}) resistive states correspond to logic '0' and logic '1', respectively. According to Fig. 6, in a MAGIC logic gate there are memristors which serve as inputs (denoted as $M_{IN\_}1,2,\ldots$), connected in series to an additional memristor which serves as the output (denoted as M_{OUT}). The number of input memristors is equal to the number of logic inputs. The actual logic inputs of the MAGIC gates are the initial resistive states of the input memristors, which need to be previously programmed accordingly, and the logic output is the final resistive state of the output memristor, after the input voltage V_0 has been applied to the circuit gateway.

More specifically, operation of a MAGIC gate consists in two sequential stages: (i) The initialization of the output memristor to a specific resistive state; and (ii) the initialization of the input memristors and finally the application of a voltage pulse of appropriate amplitude V_0 across the logic gate. Initialization of every memristor is performed as a regular write operation, by applying a voltage of proper amplitude and polarity on the terminals of every device separately. Overall, in all the MAGIC circuits there is a voltage divider formed between the input memristive combination and the single output memristor. So, while the V_0 pulse is applied, the voltage drop on the output memristor depends on the predefined composite resistive state of the input memristors and, consequently, it may cause a partial or full switching of the output memristor, meaning that the circuit operation is destructive for the output memristor which holds the logic result.

Taking for example the two-input NOR MAGIC gate shown in Fig. 6(a), at stage (i) the output memristor is initialized to R_{ON}, i.e. logic '1', whereas at stage (ii) the logic input combination is sequentially encoded in the resistive states of the input memristors. Finally, the input voltage pulse V_0 is applied to evaluate the logic output. Assuming the "00" as the input combination, both of the input memristors are set to R_{OFF}, hence the input composite memristance is $R_{OFF} \parallel R_{OFF} = 1/2 \times R_{OFF}$, i.e. still much higher than the resistance of the output memristor if we assume $R_{OFF} \gg R_{ON}$ (with $\parallel$ we define two parallel memristors). As a result, the voltage drop on the output memristor does not exceed its $|V_{RESET}|$ threshold, so its logic state remains '1'. However, for any other input combination among {"01", "10", "11"}, the input composite memristance becomes $\{R_{OFF} \parallel R_{ON} \approx R_{ON}, R_{ON} \parallel R_{OFF} \approx R_{ON}, R_{ON} \parallel R_{ON} = 1/2 \times R_{ON}\}$ and then the voltage across the output memristor is high enough to cause its switching to logic '0'.

Increasing the amplitude V_0 of the applied input voltage decreases the delay of the logic gates [41]. However, considering the cases when the voltage drop on the output memristor is above or below the switching voltage threshold, leads to particular operational limitations and design constraints regarding the proper V_0 amplitude and the preferable properties of the memristive devices (e.g. $|V_{SET}|>|V_{RESET}|$), described in detail in [41]. Including more inputs in the logic circuits requires using more input memristors which should be accessed and programmed separately every time a new input combination is applied. Of course, resetting the output memristor, between the application of different input combinations, may also be required.

Based on the same design principles, more Boolean functions are included to the MAGIC family. In fact, connecting the input memristors of the NOR gate in series, rather than in parallel, while keeping the same polarity, produces a NAND gate. Moreover, the OR and AND logic gates have the same structure as their NOR and NAND counterparts, except for the opposite polarity of the output memristor, as shown in Fig. 6(b, d). Unlike in NAND and NOR gates, the OR and AND gates require that the output memristor is initially set to logic '0'. In the same context, a MAGIC NOT gate (inverter) simply consists of a single input memristor and of the output memristor, connected in series with an opposite polarity, i.e. forming a CRS [56]. In the NOT gate the output memristor is initially set to logic ‘1’. So, depending on the resistive state of the input memristor, the application of the input voltage V_0 will result in an output voltage drop which will either force the output memristor to switch state or not.

Unlike in IMPLY logic, in MAGIC circuits the input and output devices are separate devices. However, although there is only one applied voltage which controls the function of the logic gates, MAGIC gates impose strict limitations over the input voltage V_0 amplitude, which varies depending on the type of the logic gate, the switching characteristics of the memristors, and the number of logic inputs [41]. Additionally, MAGIC is a sequential logic concept where cascading of logic gates, in order to create complex logic computations, seems difficult. Moreover, despite the apparent simplicity of the logic gate circuits, their operation requires accessing and programming every memristor separately, thus having an impact on the necessary driving circuitry. E.g. an n-input operation requires the prior separate programming of up to $n + 1$ memristors.

D. CMOS/Memristor Threshold Logic

Threshold logic provides powerful computational properties beyond Boolean logic and the development of high-performance threshold logic circuits would benefit a number of important applications in computing. The transfer function of a n-input linear threshold gate (LTG) is defined as:

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \sum_{i=1}^{n} w_i x_i \geq T \\ 0, & \text{else} \end{cases} \qquad (1)$$

where x_i is a Boolean input variable, w_i is the integer weight of the corresponding input i, and T is an integer threshold [62]. A special case of LTG is the symmetric LTG with identical weights for different inputs. A n-input symmetric LTG can implement a maximum of n nontrivial Boolean functions $f^{(k)}(x_1, x_2, \ldots, x_n)$ defined by $w_1^{(k)} = w_2^{(k)} = \ldots = w_n^{(k)} = 1/k$ and $T^{(k)} = 1$, where $k = 1, 2, \ldots, n$.

LTG is a powerful universal gate capable of reducing on average up to two times (2 ×) the gate count for representative benchmark circuits [63]. Various implementations of LTGs have been investigated [64]. In most cases the weights are fixed and cannot be changed in-field. However, the in-field reconfiguration of the LTG’s weights is a very desirable feature for most of the applications. The resistance switching property of memristors renders them well-suited for implementing weights in LTGs. Memristors enable much better scaling prospects and hence a much denser and more potent LTG implementation, compared to those based on floating-gate transistor approaches [34], [35].

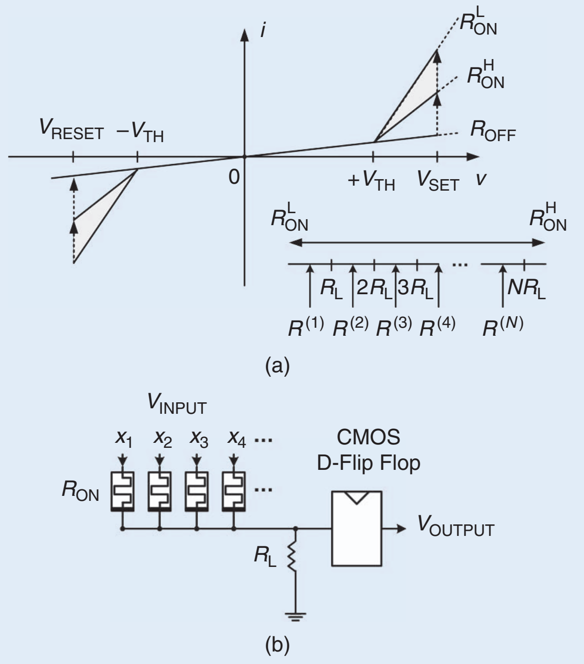


Figure 7. (a) Qualitative $i - v$ characteristics of memristors in series with a set of anti-parallel connected diodes. The shaded area shows the range of possible intermediate resistive states utilized for the implementation of the weights. (b) Circuit schematic of a LTG implemented with memristors, a load resistor, and a CMOS D flip-flop [34].

Below we discuss a hybrid CMOS/memristor solution for the implementation of a programmable LTG where memristors implement “ratioed” diode-resistor logic, whereas CMOS circuitry is used for signal amplification and inversion. Ligang *et al.* in [34] consider threshold-type memristors with a linear conductance above the switching threshold and with a configurable slope (differential conductance) $1/R$, as shown in Fig. 7(a). We assume that this slope can be configured to any value between $1/R_{ON}^{L}$ and $1/R_{ON}^{H}$. Owing to their analog nature, memristors could be (at least theoretically) programmed to any intermediate resistive state between the two boundaries R_{ON} and R_{OFF} [65]. The voltage threshold $|V_{TH}|$ is implemented with two external diodes connected in anti-parallel fashion, in series with the memristor [34].

Fig. 7(b) shows the circuit implementation of configurable “ratioed” diode-resistor logic using n memristors connected to a pull-down resistor R_L, proposed in [34]. Generally, the common connection node is connected to a CMOS component (here a D flip-flop) so that the voltage swing is restored and the output can drive other logic gates. Assuming that the CMOS gate is designed to either restore a signal to the supply voltage V_{DD} (logical ‘1’) if the input voltage is larger than $1/2 \times (V_{DD} - V_{TH})$, or otherwise to the ground (logical ‘0’), then the whole circuit implements the LTG defined by (1) where

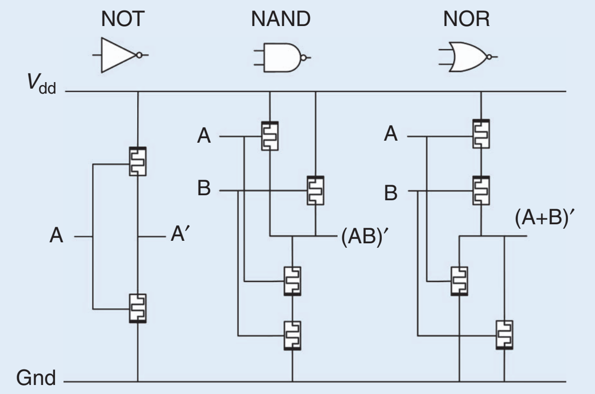


Figure 8. Overview of CMOS-*like* memristive circuit design of the universal logic gate-set utilizing FPMs and RPMs which replace the CMOS transistors [67].

$w_i = 1/R_i$ and $T = 1/R_\mathrm{L}$. For symmetric LTG the weights $w^{(k)} = 1/R^{(k)}$ and R_L must satisfy the following condition: $(k-1)\times R_\mathrm{L} < R^{(k)} < k\times R_\mathrm{L}$. The inset of Fig. 7(a) shows the possible spectrum of $R^{(k)}$ with respect to R_L. In order to implement all n Boolean functions with a symmetric LTG, the $[R_L,\ n\times R_\mathrm{L}]$ range should fit within a physically permitted range $[R_\mathrm{ON}{}^\mathrm{L},\ R_\mathrm{ON}{}^\mathrm{H}]$. Therefore $R_\mathrm{ON}{}^\mathrm{H}/R_\mathrm{ON}{}^\mathrm{L} = n$.

In principle, a simpler design of a symmetric LTG is possible with fixed weights and load resistor implemented with a memristor. However, such implementation is less suitable for circuit integration and cannot be used to implement more general LTGs. Nevertheless, having both R_L and the weights implemented with memristors may allow flexibility in choosing the optimal value of R_L [34].

Although being more suitable for artificial neural network applications, due to their functional similarity with biological neurons, LTGs have been a popular choice for the implementation of computer arithmetic circuits [66]. Evidently, an important feature of LTGs is that they do not rely on changing the state of memristive devices during the logic operation. However, in memristive LTGs the memristors need to be programmed with very high-precision [65], which is a limiting factor for the number of different memory levels that can be achieved given the high property variation among experimentally realizable memristors [10], [11].

E. CMOS-like Memristor Complementary Logic

Decreasing the overall circuit complexity of memristive IMPLY logic and achieving a straightforward circuit implementation of any Boolean logic function, were the two main pillars of the work of Vourkas *et al.* [67] which proposed a CMOS-*like* circuit design paradigm for the creation of memristive complementary logic circuits, based on the composite dynamics of groups of parallel memristors with opposite polarities.

More specifically, a 1-to-1 mapping of CMOS n- and p-type transistor networks onto an equivalent network of memristors was proposed. The word "complementary" here implies the use of symmetrical pairs of memristors with opposite polarization. The CMOS circuit design methodology is maintained, delivering similar circuit functionality; i.e. complementary digital logic circuits comprising two-state switching devices.

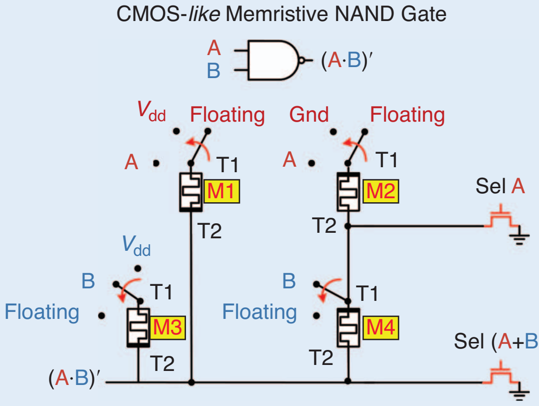
Fig. 8 demonstrates generally the CMOS-*like* memristive circuit implementation of the universal digital logic gate-set where the memristors are deliberately shown as three-terminal devices to underline similarities with the CMOS counterparts. More specifically, forward polarized memristors (FPMs) are used in the n-MOS area and reversely polarized memristors (RPMs) in the p-MOS area. The input logic signals are represented using positive voltage for logic '1' and negative voltage for logic '0'. A negative applied voltage changes the state of RPMs from R_OFF to R_ON, i.e. closes the switch. On the contrary, a positive voltage switches them from R_ON to R_OFF, i.e. the switch is open. As far as FPMs are concerned, a negative applied voltage changes their state from R_ON to R_OFF, though a positive voltage restores their state from R_OFF to R_ON. In other words, positive (negative) voltage turns the switches on (off), similar to how *n*-type FETs work for the corresponding logic input signals.

Any implemented Boolean logic function is determined by the topology of the circuit, which consists of an equivalent ohmic resistance for the upper and another one for the lower part of the CMOS-*like* design. Therefore, the output voltage V_OUT is always a fraction of the supply voltage (which is the read voltage $-V_\mathrm{DD}$), being dependent on the voltage divider across the two parts of the circuit. Output voltage values close to V_DD correspond to logic '1' whereas values close to zero (GND) correspond to logic '0'.

The general circuit functionality is explained taking as example the NOT gate of Fig. 8: Assuming {FPM, RPM} set in $\{R_\mathrm{ON}, R_\mathrm{OFF}\}$, before any further applied input we get $V_\mathrm{OUT} = V_\mathrm{DD}\times R_\mathrm{ON}/(R_\mathrm{ON}+R_\mathrm{OFF}) \ll V_\mathrm{DD} \approx \mathrm{GND}$, since $R_\mathrm{OFF} \gg R_\mathrm{ON}$. A positive input voltage (logic '1') does not alter the memristor states, thus we get the same V_OUT. However, with a negative input voltage (logic '0') the state of the two anti-parallel memristors changes, so we get $V_\mathrm{OUT} = V_\mathrm{DD}\times R_\mathrm{OFF}/(R_\mathrm{ON}+R_\mathrm{OFF}) \approx V_\mathrm{DD}$. Afterwards, a positive input to the NOT gate will restore the memristors to their initial states. Consequently, V_OUT changes as expected for any input variation. Likewise, the proposed paradigm works for the rest of the universal digital logic gates, making the design of every digital logic circuit possible.

Certain modifications were proposed for the CMOS-*like* approach in [40] in order to overcome particular circuit design limitations concerning the sequential processing of the input logic signals, the two-terminal nature of memristors (compared to the three-terminal FETs), and the requirement for proper isolation of the groups of memristors where input voltages are simultaneously applied. As an example, Fig. 9(a) shows the detailed CMOS-*like* memristive design of a 2-input NAND gate. This circuit consists of both memristors and a few auxiliary transistors which are driven by appropriate selection lines in the driving circuitry. The latter facilitate the correct access operation to multiple memristors where the same input logic signal is applied. In this example there are four switches (tri-state voltage drivers M1 through M4) [68], which determine the logic gate function, providing the option to: (i) apply input (programming) signals, (ii) read the logic output via V_{DD} and GND signals, or (iii) leave the circuit idle with the switches set in floating position.

The duration and the amplitude of the input voltage pulses V_o, should be selected so as to exceed the V_{SET} and V_{RESET} thresholds and be longer than the corresponding switching time of the devices. The V_{DD} amplitude should be selected so that the voltage drop on every memristor never exceeds the voltage thresholds. In this way, the internal state of memristors remains unaffected during read-out regardless of the current flow from V_{DD} to GND.



(a)

	Reading Phase	Apply Input A = ‘0’	Apply Input A = ‘1’	Apply Input B = ‘0’	Apply Input B = ‘1’
Sel A	‘0’	‘1’	‘1’	‘0’	‘0’
Sel (A+B)	‘0’	‘1’	‘1’	‘1’	‘1’
Switch M1	V_{dd}	A = V_o	A = $-V_o$	Floating	Floating
Switch M2	GND	A = V_o	A = $-V_o$	Floating	Floating
Switch M3	V_{dd}	Floating	Floating	B = V_o	B = $-V_o$
Switch M4	Floating	Floating	Floating	B = V_o	B = $-V_o$

(b)

Figure 9. (a) Detailed CMOS-*like* implementation of the two-input NAND logic gate. (b) Summary of the appropriate voltages (switch positions) to be applied to the corresponding devices for all possible circuit operation phases [40].

Fig. 9(b) provides information regarding the positions of the switches and the logic values applied to the access transistors for every possible phase of the circuit operation. V_{DD} and GND voltages are applied only to read the circuit logic output. In all circuit branches the devices immediately connected to the output line can be accessed using a single select transistor; e.g. $Sel(A+B)$ refers to the gate control signal of a transistor which should conduct when either input signal A or B is to be applied. Moreover, for every single input signal change, e.g. for an input sequence from $AB=$ “01” to $AB=$ “11”, the circuit output is updated in just a single step, whereas for multiple signal changes the output response delay is multiplied by the number of the changed input signals. The pulsing details of Fig. 9(b) infer that it is not possible to have more than one input signal applied simultaneously.

Compared to IMPLY logic, the CMOS-*like* paradigm: (i) reduces the number of sequential steps needed to perform logic operations. In fact, the necessary steps are equal to the number of the circuit's inputs, plus (if necessary) the number of their complements, hence reaching a maximum of $2n$ for a n-input arbitrary Boolean function (inputs' complements are assumed readily available). Furthermore: (ii) it simplifies the circuit design procedure because similar design principles with conventional CMOS are applied. In terms of circuit area, the exact number of needed memristors cannot be formerly defined because a specific logic function has many equivalent CMOS-*like* circuit implementations. Finally, cascading CMOS-*like* logic circuits is not supported because the circuit output varies between GND $-$ V_{DD} whereas the input signals applied to the bipolar memristors require also negative voltages; hence input and output are incompatible.

F. Parallel Input-Processing Memristor Logic

The practical application of the memristive technology in logic circuit design will require substantial parallel operations in order to countervail the impact from the driving circuitry. The IMPLY, MAGIC, and CMOS-*like* design approaches consider serial processing of input signals. However, a design allowing for parallel processing of inputs would be certainly more promising and could accelerate the implementation of new generation memristive logic chips. The MRL paradigm was an early approach towards this end, also enabling significant physical integration savings. However, it provides only a restricted set of possible Boolean logic operations, while it uses linear memristors which respond slower than threshold-type devices. To this end, Papandroulidakis *et al.* in [42] presented a novel memristive logic family which enables parallel execution of digital logic operations, exploiting the threshold-dependent resistance switching behavior of memristors and of their compositions.

This memristive logic family uses the total conductance (memductance) of the devices for the parallel computation of AND, OR, NAND, NOR, XOR, and XNOR logic operations. Fig. 10 summarizes the general concept for the construction of two-input logic gates. The qualitative $i-v$ graphs of Fig. 10(a-c) show why memductance is inherently suitable to be the state variable for Boolean logic operations. According to Fig. 10(d), the logic input combination is encoded to a corresponding positive aggregate input voltage which is applied to a memristive ensemble that can be any of the six provided options, each implementing a different logic operation. The input voltages consist in 0V for logic '0', whereas logic '1' corresponds to a voltage which lies between the first (lower) and the second (higher) of the defined switching thresholds.

As shown in Fig. 10(a), a forward polarized memristor (FPM) will switch from a low conductance (L) to a high conductance (H) if any of the applied inputs (or both) is logic '1', i.e. it exceeds the set threshold $V_{s,1}$. Likewise, with two FPMs in series the composite memductance will rise from a low value (L') to a high value (H') only when both inputs are logic '1', so that the aggregate input voltage exceeds the cumulative set threshold $2 \times V_{s,1}$. Therefore, memductance in these two cases is defined by the following equations which describe OR and AND logic operations as functions of the aggregate applied voltage:

$$OR: G(V_{\mathrm{IN,SUM}}) = \begin{cases} H, V_{\mathrm{IN,SUM}} > V_{\mathrm{S},1} \\ L, \quad \text{otherwiswe} \end{cases} \tag{2}$$

$$\mathrm{AND}: G(V_{\mathrm{IN,SUM}}) = \begin{cases} H', V_{\mathrm{IN,SUM}} > 2 \times V_{\mathrm{S},1} \\ L', \quad \text{otherwiswe} \end{cases} \tag{3}$$

With a reversely polarized memristor (RPM) or two RPMs in series, as shown in Fig. 10(b), under similar principles the circuit implements a NOR and a NAND logic gate, respectively. Unlike in Fig. 10(a), now it is the reset thresholds $V_{\mathrm{R},1}$ and $2 \times V_{\mathrm{R},1}$ which define memristance switching. OR and NOR gates with more inputs require adding more than two signals to the applied sum; this is practically limited by the maximum input voltage a memristor can tolerate without being damaged. For

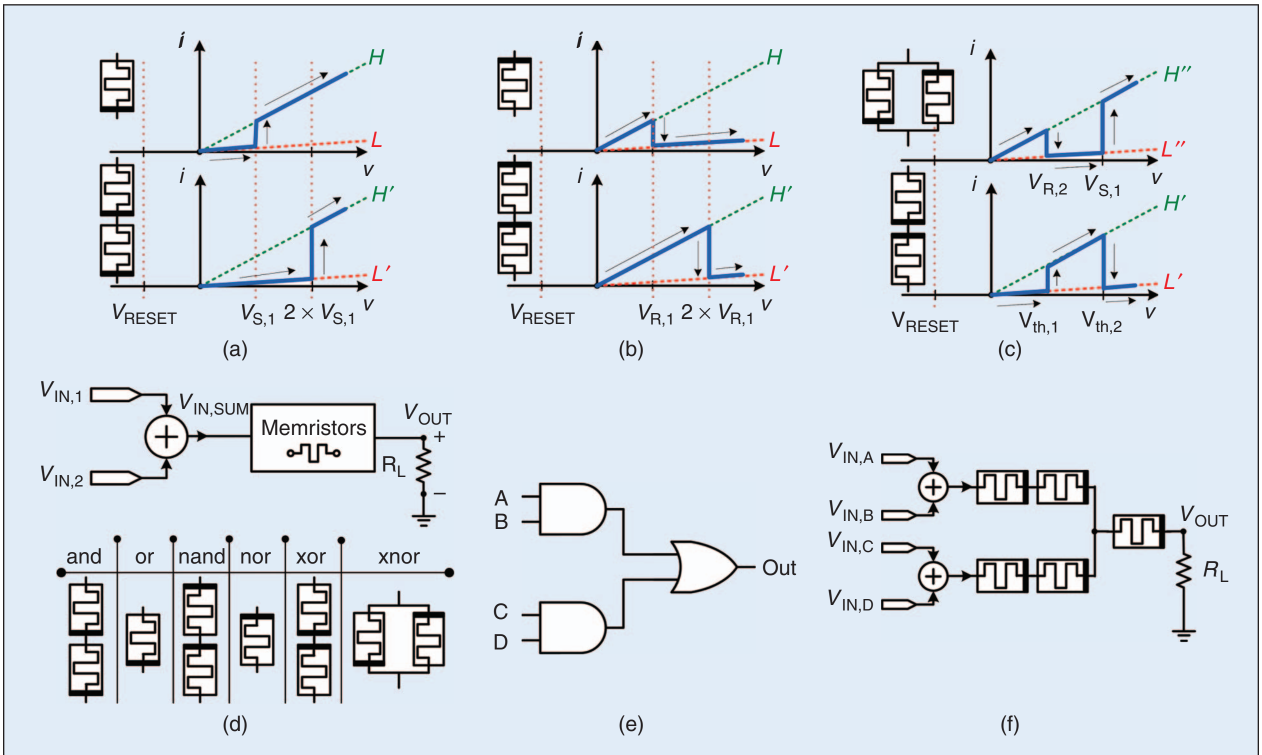


Figure 10. General concept for the construction of 2-input logic gates. (a-c) Qualitative $i-v$ characteristics corresponding to FPMs or RPMs and their series/parallel combination. (d) Circuit implementing any of the six provided logic operations with two variables. (e) Schematic of a typical sum-of-products logic function, and (f) its implementation with cascaded logic gates [42].

AND and NAND operations, an additional memristor has to be included for each additional input signal, to increase accordingly the cumulative threshold.

According to Fig. 10(c), with two reciprocal devices in series or in parallel, the circuit implements XOR and XNOR logic operations, respectively. In the series connection the composite memductance rises from a low level (L') to a high level (H') if any of the applied inputs is logic '1', i.e. exceeds the set threshold $V_{\text{th},1}$. If both inputs are logic '1', together they exceed the reset threshold $V_{\text{th},2}$ and the composite memductance collapses to level L'. In the same fashion, the equivalent memductance of the parallel configuration collapses from level H'' to L'' if only one input is logic '1', i.e. $> V_{\text{R},2}$, but rises to H'' if both inputs are logic '1', i.e. together exceed $V_{\text{S},1}$. The memductance in these cases is defined as:

$$XOR : G(V_{\text{IN, SUM}}) = \begin{cases} H', V_{th,1} < V_{\text{IN, SUM}} < V_{\text{th},2} \\ L', \quad \text{otherwiswe} \end{cases} \tag{4}$$

$$XNOR : G(V_{\text{IN, SUM}}) = \begin{cases} L'', V_{R,2} < V_{\text{IN, SUM}} < V_{S,1} \\ H'', \quad \text{otherwiswe} \end{cases} \tag{5}$$

All the memristors of a logic gate can be reset together via a single reset pulse, without having to be accessed separately. After the initialization of a gate via a reset pulse, any logic input combination will have an irreversible effect on the conductance of the memristors; e.g. if we apply "10" or "01" and the next input is "00", then unless an intermediate reset step is applied, the final output will be wrong because of the nonvolatility of the memristors. However, between "01" and "10" inputs there is no need for a reset step. The same applies if the next input (after a "10" or "01") is "11". Consequently, algorithmically the efficiency of the circuits could be improved by evaluating the number of '1' in the input combination. A similar analysis could improve efficiency of the MAGIC gates as well.

Regarding the composite conductance levels of the memristive ensembles, assuming $\{L, H\} = \{G_{\text{ON}}, G_{\text{OFF}}\}$, then for the parallel/serial compositions it is $\{L', H'\} = \{1/2 \times G_{\text{ON}}, 1/2 \times G_{\text{OFF}}\}$ and $\{L'', H''\} = .\ \{2 \times G_{\text{ON}}, 2 \times G_{\text{OFF}}\}$ Potential impact by variation in the switching thresholds could be overcome by optimally selected programming pulses. Regarding memristors with different polarities, a simulation-based exploration of the preferable relation between the V_{SET} and V_{RESET} thresholds in [13] showed that having $|V_{\text{RESET}}| > V_{\text{SET}}$ or $|V_{\text{RESET}}| = V_{\text{SET}}$ ($|V_{\text{RESET}}| < V_{\text{SET}}$) is preferable for the anti-series (for the anti-parallel) memristors; when $|V_{\text{RESET}}| = V_{\text{SET}}$ the parallel memristors initiate switching simultaneously ($R_{\text{OFF}} \parallel R_{\text{ON}} \leftrightarrow R_{\text{ON}} \parallel R_{\text{OFF}}$) and there is almost no intermediate switching stage with both of them in R_{OFF}.

Circuit output (V_{OUT}) of either single or cascaded logic gates could be read using a series load resistor R_{L}. Such resistor should have a small value (better smaller than R_{ON}) so as to draw a small voltage on its terminals and not to impede the complete (or almost complete) switching of the memristor(s) during the logic operations, unless it can be connected only during the output read-out. This is more crucial when logic gates are cascaded as shown in Fig. 10(e) where a typical sum-of-products digital circuit is presented, with the corresponding implementation in Fig. 10(f). Similarly to MRL, the presented gates lack signal restoration since they are built out of passive elements only. Output voltage levels degrade from input to output, thus these logic gates cannot be cascaded for many stages without intermediate signal amplification [42]. Therefore, CMOS inverter/buffer is necessary to make this logic family computationally complete and to provide signal inversion and amplification.

IV. Conclusions

While reaching and exploring the post-CMOS and beyond-von Neumann era, one thing is for sure; that memristor-based logic design and in-memory computing will attract more and more attention, giving rise to an ever-growing number of different computing approaches. Such approaches may depend either purely on the properties of single devices, or on the collective behavior of many interconnected devices. The memory property of the memristor ensures that the results of previous computations are maintained in the states of the employed devices.

Overall, the performance of any memristive logic design paradigm, in terms of processing speed and energy consumption, is expected to strongly depend on device material selection and operation schemes. The delay of logic operations involves the time that memristors require to switch their states, which also depends on the level of the applied voltage since higher voltage pulses will switch the devices more quickly. The memristors themselves are capable of fast (nanoseconds) and low-energy (picojoules) switching. So, in terms of speed and energy consumption, such circuits are very promising for future nanoelectronics. Circuit area, though, will depend on the type of integration with the silicon-based driving circuitry and/or on the utilization of the ultra high device density of the memristive nanowire crossbar architecture.

In this review we shortly presented some of the popular trends in memristive logic design, commenting on their most important characteristics, the performance goals, and their overall pros and cons. We also revealed the main focus of recent research efforts on making logic operations as parallel as possible. Defining fair comparison metrics for such emerging circuit design approaches is still difficult, so we avoided their strict

Table 1.
Summary of major properties of the studied memristive logic design concepts.

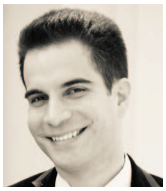
	IMPLY [36–39]	MRL [33]	MAGIC [41]	Threshold logic [34]	CMOS-like [40]	Parallel input processing [42]
Computation	sequential	parallel	sequential	parallel	sequential	parallel
Logic state variable	memristance	voltage	memristance	voltage	voltage	memristance
Main circuit structure	memristors + resistor	memristors†	memristors	memristors† + diodes + resistor	memristors	memristors† + resistor
Preferred memristor type	threshold-type	linear*	threshold-type	threshold-type	threshold-type	threshold-type
Complexity of driving/ auxiliary circuitry	high*	low	very high*	medium*	very high*	medium*
Cascading logic gates	possible	possible	difficult	possible	not possible	possible
Comments*	requires tri-state voltage drivers	longer expected switching latency	requires CMOS transistors to access every memristor	requires tri-state voltage drivers but only for programming	requires tri-state voltage drivers and CMOS transistors	requires CMOS transistors to reset every logic gate

†: CMOS components are used as well for signal restoration and amplification.

comparison, rather commenting on their underlying principles and major operational differences.

The major conclusions of our commentary are summarized in Table I and concern: (i) the main circuit structure; (ii) the preferred type of memristor devices; (iii) the way computation is realized (being either sequential or parallel); (iv) the circuit property which serves as the logic state variable; (v) the possibility to cascade several gate-stages; and (vi) the impact of the driving/auxiliary circuitry. So, observing Table I reveals the following:

- Among the *parallel processing circuit design concepts*, despite the fact that it assumes linear switching memristors which impose longer latency, the MRL concept generally outperforms as the concept with the simplest circuit structure and the lowest required complexity for the driving circuitry, supporting also gate-cascading. The parallel input-processing concept (last column of Table I) assumes a similar circuit structure with an additional load resistor, though when multiple gates are cascaded then CMOS transistors are required to properly access and reset altogether the memristors of every logic gate. Finally, compared to MRL, the linear threshold logic (LTG) implementation concept assumes additional diodes and a resistor along with the memristors, whereas the driving circuitry includes tri-state voltage drivers. The latter, however, are required only during the programming stage of the memristor states and not during logic computations, which by the way, are expected to last shorter than in MRL gates.
- Among the *sequential processing circuit design concepts*, the IMPLY logic assumes a relatively simple circuit structure (in spite of the large number of memristors) which, similar to MRL and LTG concepts, it is suitable to be implemented in crossbar architecture for high device-density designs. This is the main reason behind the current research efforts towards parallel IMPLY-based computing architectures, though at the cost of higher overall circuit complexity. Compared to IMPLY logic, the MAGIC and the CMOS-*like* sequential concepts require a much higher complexity for their driving circuitry, although the latter of those improves significantly the computation time of IMPLY logic. Moreover, they do not permit cascading logic gates (or pose challenges in doing so), and generally seem unlikely to accept significant further improvements, thus falling behind their IMPLY-based sequential counterpart.



Ioannis Vourkas (S'12-M'16) received the M. Eng. (Diploma) and the Ph.D. degrees in Electrical and Computer Engineering (ECE) from the Democritus University of Thrace (DUTh), Xanthi, Greece, in 2008 and 2014, respectively. Currently he is with the Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile, working as Post-doctoral Researcher and Principal Investigator in the project CONICYT FONDECYT Postdoctorado No. 3160042 (2016).

In 2006 he was summer-intern in the Cultural & Educational Technol. Institute in Xanthi, Greece. In 2009 he worked via the European vocational training program "Leonardo Da Vinci" in the Institute of Industrial & Control Eng. of the Polytechnic University of Catalonia—BarcelonaTech, Barcelona, Spain. In 2012 he was with the organizing committee of the 2012 *Int. Conf. on Cellular Automata for Research and Industry* (*ACRI*), where he met the "inventor" of the memristor, Prof. L. Chua. His current research emphasis is on novel nano-electronic circuits and architectures comprising memristors. His research interests include the design of nanoelectronic integrated circuits, unconventional computing, software and hardware aspects of parallel complex computational (bio-inspired) circuits and systems, Cellular Automata and their applications. In these areas, so far he is main author of one book, one book chapter, and more than 30 scientific journal and conference papers.

Dr. Vourkas distinguished as the 2014 *Best Ph.D. Student* in 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 Conf.* held in Istanbul, Turkey. He was *invited Tutorial speaker* in the 2015 *IEEE Int. Conf. on Electronics, Circuits, and Systems* (*ICECS*) held in Cairo, Egypt. He was scholar of the Greek BODOSSAKI foundation from 2011 to 2014.

Georgios Ch. Sirakoulis (M'95) received the M. Eng. (Diploma) and Ph.D. degree in electrical and computer engineering (ECE) from the Democritus University of Thrace (DUTh), Greece, in 1996 and 2001, respectively. He is a Tenured Associate Professor with 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 at the EU IDEAS program. He has participated as a PI in more than 20 scientific programs and projects funded from the Greek Government and Industry as well as 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 the Vice President of the IEEE Student Branch of Thrace from 2000 to 2001. He serves as an Assoc. Editor of the *Microelectronics Jour.*, the *Integration, the VLSI Jour.*, the *Int. Jour. of Parallel, Emergent and Distributed Systems*, the *Jour. of Cellular Automata*, the *Jour. of Appl. Mathematics*, and the *Recent Advances on Electrical and Electronic Engineering.* He was General Co-Chair of the 2014 *Int. Conf. on Cellular Automata for Research and Industry* (*ACRI*), and several other conferences, including *ACRI* 2012, the 5th *Pan-Hellenic ECE Students Conf.*, of Special Sessions in the 2015 *IEEE Int. Conf. on Nanotechnology*, the 2015 *Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing* (*PDP*), *PDP* 2014, the 2014 *HiPEAC Computing Systems Week*, the 18th and 20th *Pan-Hellenic Conf. in Informatics*, the 20th *IEEE Int. Conf. on Electronics, Circuits, and Systems*, and many others.

References

[1] L. O. Chua, "Memristor: The missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[2] L. O. Chua, "If it's pinched it's a memristor," *Semicond. Sci. Technol.*, vol. 29, no. 10, pp. 104001, 2014.

[3] L. O. Chua and S. M. Kang, "Memristive devices and systems," *IEEE Proc.*, vol. 64, no. 2, pp. 209–223, 1976.

[4] T. Prodromakis, C. Toumazou, and L. O. Chua, "Two centuries of memristors," *Nat. Mater.*, vol. 11, no. 6, pp. 477–557, June 2012.

[5] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[6] S. D. Ha and S. Ramanathan, "Adaptive oxide electronics: A review," *J. Appl. Phys.*, vol. 110, no. 7, pp. 071101, 2011.

[7] C. H. Yang, et al., "Electric modulation of conduction in multiferroic Ca-doped BiFeO3 films," *Nat. Mater.*, vol. 8, pp. 485–493, 2009.

[8] S. Kim, et al., "Effect of scaling WOx-based RRAMs on their resistive switching characteristics," *IEEE Electron Device Lett.*, vol. 32, no. 5, pp. 671–673, 2011.

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

[10] ITRS. (2013). International Technology Roadmap for Semiconductors. [Online]. Available: http://www.itrs.net/

[11] Y. V. Pershin and M. Di Ventra, "Memory effects in complex materials and nanoscale systems," *Adv. Phys.*, vol. 60, no. 2, pp. 145–227, 2011.

[12] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "The desired memristor for circuit designers," *IEEE Circ. Syst. Mag.*, vol. 13, no. 2, pp. 17–22, 2013.

[13] A. Ascoli, F. Corinto, V. Senger, and R. Tetzlaff, "Memristor model comparison," *IEEE Circ. Syst. Mag.*, vol. 13, no. 2, pp. 89–105, 2013.

[14] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Memristor SPICE modeling," in *Advances in Neuromorphic Memristor Science and Applications*, 1st ed. The Netherlands: Springer, 2012, pp. 211–244.

[15] I. Vourkas and G. C. Sirakoulis, "Memristive crossbar-based nonvolatile memory," in *Memristor-Based Nanoelectronic Computing Circuits and Architectures*, 1st ed. Switzerland: Springer, 2016, pp 101–147.

[16] Panasonic. (2012, May 15). The new microcontrollers with on-chip non-volatile memory ReRAM. [Online]. Available: http://panasonic.co.jp/corp/news/official.data/data.dir/jn120515-1/jn120515-1.html

[17] Intel and Micron. (2015, July). 3D XPoint technology. [Online]. Available: https://www.micron.com/about/emerging-technologies/3d-xpoint-technology

[18] Bio Inspired Technologies. (2015, Dec. 15). The Neuro-bit: the world's first commercially available memristor. [Online]. Available: http://www.bioinspired.net/

[19] F. L. Traversa, C. Ramella, F. Bonani, and M. Di Ventra, "Memcomputing NP-complete problems in polynomial time using polynomial resources and collective states," *Sci. Adv.*, vol. 1, no. 6, pp. e1500031, July 2015.

[20] T. Chang, Y. Yang, and W. Lu, "Building neuromorphic circuits with memristive devices," *IEEE Circ. Syst. Mag.*, vol. 13, no. 2, pp. 56–73, 2013.

[21] M. P. Sah, H. Kim, and L. O. Chua, "Brains are made of memristors," *IEEE Circ. Syst. Mag.*, vol. 14, no. 1, pp. 12–36, 2014.

[22] S. P. Adhikari, M. P. Sah, H. Kim, and L. O. Chua, "Three fingerprints of memristor," *IEEE Trans. Circuits Syst. I*, vol. 60, no. 11, pp. 3008–3021, 2013.

[23] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: Materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, pp. 382001, 2013.

[24] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, pp. 61–64, May 2015.

[25] 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. 30, pp. 305205, 2012.

[26] G. S. Rose, J. Rajendran, H. Manem, R. Karri, and R. E. Pino, "Leveraging memristive systems in the construction of digital logic circuits," *IEEE Proc.*, vol. 100, no. 6, pp. 2033–2049, 2012.

[27] M. Di Ventra and Y. V. Pershin, "The parallel approach," *Nat. Phys.*, vol. 9, pp. 200–202, Apr. 2013.

[28] A. Siemon, et al., "Realization of Boolean logic functionality using redox-based memristive devices," *Adv. Funct. Mater.*, vol. 25, no. 40, pp. 6414–6423, 2015.

[29] D. Sacchetto, P. Gaillardon, M. Zervas, S. Carrara, G. De Micheli, and Y. Leblebici, "Applications of multi-terminal memristive devices: A review," *IEEE Circ. Syst. Mag.*, vol. 13, no. 2, pp. 23–41, 2013.

[30] S. Balatti, S. Ambrogio, and D. Ielmini, "Normally-off logic based on resistive switches: Part I: Logic gates," *IEEE Trans. Electron Dev.*, vol. 62, no. 6, pp. 1831–1838, 2015.

[31] Y. V. Pershin and M. Di Ventra, "Neuromorphic, digital and quantum computation with memory circuit elements," *IEEE Proc.*, vol. 100, no. 6, pp. 2071–2080, 2012.

[32] J. Borghetti, et al., "A hybrid nanomemristor/transistor logic circuit capable of self-programming," *Proc. Nat. Acad. Sci. USA*, vol. 106, no. 6, pp. 1699–1703, 2009.

[33] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser and E. G. Friedman, "MRL: Memristor ratioed logic," in *Proc. Int. Workshop on Cellular Nanoscale Network and Application*, Turin, Italy, 2012, pp. 1–6.

[34] G. Ligang, F. Alibart, and D. B. Strukov, "Programmable CMOS/memristor threshold logic," *IEEE Trans. Nanotechnol.*, vol. 12, no. 2, pp. 115–119, 2013.

[35] J. Rajendran, H. Manem, R. Karri and G. S. Rose, "Memristor based programmable threshold logic array," in *Proc. IEEE/ACM Int. Symp. Nanoscale Architectures*, Anaheim, CA, 2010, pp. 5–10.

[36] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "Memristive switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, 2010.

[37] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. VLSI Syst.*, vol. 22, no. 10, pp. 2054–2066, 2014.

[38] E. Lehtonen, J. H. Poikonen and M. Laiho, "Implication logic synthesis methods for memristors," in *Proc. IEEE Int. Symp. Circuits and Syst.*, Seoul, South Korea, 2012, pp. 2441–2444.

[39] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *Proc. IEEE/ACM Int. Symp. Nanoscale Architectures*, San Francisco, CA, 2009, pp. 33–36.

[40] I. Vourkas and G. C. Sirakoulis, "Memristor–based combinational circuits: A design methodology for encoders/decoders," *Microelectron. J.*, vol. 45, no. 1, pp. 59–70, 2014.

[41] S. Kvatinsky, et al., "MAGIC: Memristor aided LoGIC," *IEEE Trans. Circuits Syst. II*, vol. 61, no. 11, pp. 895–899, 2014.

[42] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, and G. C. Sirakoulis, "Boolean logic operations and computing circuits based on memristors," *IEEE Trans. Circuits Syst. II*, vol. 61, no. 12, pp. 972–976, 2014.

[43] 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.

[44] D. B. Strukov and K. K. Likharev, "CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, no. 6, pp. 888–900, 2005.

[45] 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.

[46] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, pp. 035204, 2007.

[47] H. Yan, et al., "Programmable nanowire circuits for nanoprocessors," *Nat. Lett.*, vol. 470, pp. 240–244, 2011.

[48] Y. V. Pershin and M. Di Ventra, "Self-organization and solution of shortest-path optimization problems with memristive networks," *Phys. Rev. E*, vol. 88, pp. 013305, July 2013.

[49] F. Jiang and B. E. Shi, "The Memristive grid outperforms the resistive grid for edge preserving smoothing," in *Proc. European Conf. Circuits Theory and Design*, Antalya, Turkey, 2009, pp. 181–184.

[50] Y. V. Pershin and M. Di Ventra, "Solving mazes with memristors: A massively parallel approach," *Phys. Rev. E*, vol. 84, pp. 046703, 2011.

[51] I. Vourkas and G. C. Sirakoulis, "On the generalization of composite memristive network structures for computational analog/digital circuits and systems," *Microelectron. J.*, vol. 45, no. 11, pp. 1380–1391, 2014.

[52] A. Adamatzky and L. Chua, Eds. *Memristor Networks*, Switzerland: Springer, 2014.

[53] HP Cloud Source Blog. (June 17, 2014). The Machine, a view of the future of computing. [Online]. Available: http://h30507.www3.hp.com/t5/Cloud-Source-Blog/The-Machine-a-view-of-the-future-of-computing/ba-p/164568#.VB2Bw5qKDGg

[54] O. Kavehei, S. Al-Sarawi, K. R. Cho, K. Eshraghian, and D. Abbott, "An analytical approach for memristive nanoarchitectures," *IEEE Trans. Nanotechnol.*, vol. 11, no. 2, pp. 374–385, 2012.

[55] J. P. Strachan, et al., "State dynamics and modeling of tantalum oxide memristors," *IEEE Trans. Electron Dev.*, vol. 60, no. 7, pp. 2194–2202, 2013.

[56] R. Rosezin, E. Linn, L. Nielen, C. Kügeler, R. Bruchhaus, and R. Waser, "Integrated complementary resistive switches for passive high-density nanocrossbar arrays," *IEEE Elec. Dev. Lett.*, vol. 32, no. 2, pp. 191–193, 2011.

[57] R. K. Budhathoki, M. P. Sah, S. P. Adhikari, H. Kim, and L. Chua, "Composite behavior of multiple memristor circuits," *IEEE Trans. Circuits Syst. I*, vol. 60, no. 10, pp. 2688–2700, 2013.

[58] A. N. Whitehead and B. Russell, *Principia Mathematica*, vol. I, no. 7, Cambridge, UK: Cambridge University Press, 1910.

[59] E. Lehtonen, J. Tissari, J. Poikonen, M. Laiho, and L. Koskinen, "A cellular computing architecture for parallel memristive stateful logic," *Microelectron. J.*, vol. 45, no. 11, pp. 1438–1449, 2014.

[60] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nat. Nano.*, vol. 8, pp. 13–24, 2013.

[61] R. H. Wilkinson, "A method of generating functions of several variables using analog diode logic," *IEEE Trans. Electron. Comp.*, vol. EC12, no. 2, pp. 112–129, 1963.

[62] S. Muroga, *Threshold Logic and Its Applications*. Hoboken, NJ: Wiley, 1972.

[63] R. Zhang, P. Gupta, L. Zhong, and N. K. Jha, "Synthesis and optimization of threshold logic networks with application to nanotechnologies," in *Proc. Design Automation and Test in Europe Conf.*, Paris, France, 2004, vol. 2, pp. 904–909.

[64] V. Beiu, J. M. Quintana, and M. J. Avedillo, "VLSI implementations of threshold logic: A comprehensive survey," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1217–1243, 2003.

[65] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, pp. 075201, 2012.

[66] Y. Leblebici, H. Ozdemir, A. Kepkep, and U. Cilingiroglu, A compact high-speed (31, 5) parallel counter circuit based on capacitive threshold logic gates," *IEEE J. Solid-State Circuits*, vol. 31, no. 8, pp. 1177–1183, 1996.

[67] 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, Nov 2012.

[68] I. Vourkas and G. C. Sirakoulis, "Employing threshold-based behavior and network dynamics for the creation of memristive logic circuits and architectures," *Physica Status Solidi (c)*, vol. 12, no. 1-2, pp. 168–174, 2015.