SIXOR: Single-Cycle In-Memristor XOR

Nima TaheriNejad, *Member, IEEE*

***Abstract*— With the fast approach of the end of silicon scaling and existing problems, such as the Von-Neumann bottleneck, alternative computing paradigms are in demand. In-memory computation (IMC) is one of the most promising solutions, and memristive technology is one of the best platforms for that purpose. Many logic families have been proposed to enable memristive IMC, among which *stateful* logic family stands out due to its minimal power consumption and simplicity. In this work, to complement existing works, we propose the first stateful crossbar-compatible `XOR` atomic logic operation that requires only one cycle for its completion, which is two times faster than the current minimum required time for performing `XOR` (which is two cycles) using other atomic operations in comparable memristive stateful logic families. We show that, in an example case of an adder, by taking advantage of the proposed single-cycle in-memristor XOR (SIXOR), up to 4.5× speedup can be achieved compared to other SoA stateful adders. The gained speed-up scales up in more complex systems and calculations that use `XOR`.**

***Index Terms*— In-memory computing, memristors, resistive random access memory (ReRAM), single-cycle logic, stateful logic, `XOR`.**

I. Introduction

COMPUTER technology is currently facing severe challenges in keeping up with the demand for increased performance capabilities. Dennard's scaling has already stopped [1], and the speed-up gains due to silicon scaling have decreased. By approaching the size of a single atom, the threat of coming to the end of silicon scaling is more serious than ever. One of the frequently used solutions for avoiding these technological hurdles has been increasing the number of processing cores and their capabilities. However, that too is currently facing a critical challenge: "Von-Neumann Bottleneck" [2]. Transferring data from storage units to the processing cores takes up a significant portion of the system's energy, resources, and cost. This is the major bottleneck of high-performance and data-intensive applications since data cannot be provided to the processing units fast enough.

The abovementioned challenges have motivated science and industry to pay more attention to emerging technologies and innovative methods. Memristive technology and in-memory computation (IMC) are two of those promising emerging solutions. The infusion of the two, namely, IMC using memristors, is an ideal approach to which a substantial amount of research has been dedicated. Memristors are nonvolatile devices, often built in large arrays in a very compact fashion [3]. They can store data [4]–[8] and perform logical operations [9]–[13] and calculations [14]–[17], where the data are located. This reduces unnecessary data transfer and alleviates the Von-Neumann bottleneck [2]. This advantage is more significant when working with big data (e.g., sorting data [18]).

The nonvolatility of memristors renders them ultra-low power, and their compactness makes them economical. They are also compatible with complementary metal-oxide semiconductor (CMOS) technology, as they can be built in the Back end of line (BEOL) [19], [20] in additional steps and without requiring changes in the traditional CMOS technology. This ensures a smooth transition and minimal additional cost. These advantages have led to their incorporation in some products in the market (e.g., in certain Panasonic products [21] or crossbar memories [22]) and in major technology providers' processes, such as Taiwan Semiconductor Manufacturing Company (TSMC) [23].

Among the myriad of possibilities for near- and in-memory computing using memristors and memristive logics [9], [10], [10], [11], [14], [24]–[34], *stateful* logics seem to be one of the most promising solutions. In *stateful* logics, both input and output values are represented as the internal state of memory elements, which is the resistance of memristors, also known as memristance. A unique advantage of this kind of logic is that no read or write is necessary for performing logical operations and calculations. Saving reads and writes for when they are only intended as such (e.g., once at the beginning or end of a long process or function to transfer inputs and results of calculation) prevents unnecessary power consumption and delay in computations, which read and write operations impose. This simplifies the stateful logic operations and increases their efficiency. Many logical functions can be executed as atomic stateful operations, but there is currently no possibility of running `XOR` in a single cycle. Due to the importance and pervasive use of this logical function, creating a circuit that allows single-cycle execution of `XOR` will have significant benefits for a wide range of circuits and systems. This article tackles this challenge by proposing a new memristive circuit that can perform `XOR` in a single cycle.

The rest of this article is organized as follows. We begin with a brief literature review in Section II. In Section III, we present the proposed design. We verify the operation of single-cycle in-memristor XOR (SIXOR) and formulate the design constraints in Section IV. In Section V, using the example of a new adder, we showcase the advantages of using the proposed `XOR`. We discuss other design and usage considerations, including crossbar arrays, in Section VI.

Manuscript received November 1, 2020; revised January 4, 2021 and February 15, 2021; accepted February 21, 2021.

The author is with the Institute of Computer Technology (ICT), Technische Universität Wien (TU Wien), 1040 Vienna, Austria (e-mail: nima.taherinejad@tuwien.ac.at).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3062293.

Digital Object Identifier 10.1109/TVLSI.2021.3062293

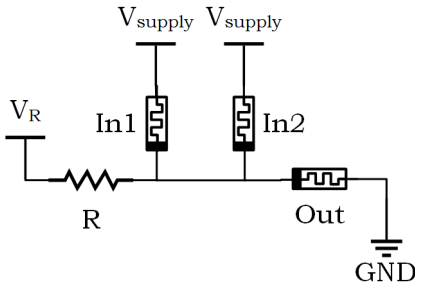

Fig. 1. TMSL AND [12].

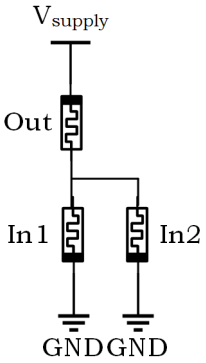

Fig. 2. FELIX OR [13].

Finally, we delve deeper into implementation aspects in Section VII before concluding this article in Section VIII.

II. Stateful Logics—Overview

The first stateful logic in the literature was material implication (IMPLY), proposed in the late 2000s [9], [10]. Afterward, memristor-aided logic (MAGIC) was proposed in 2014 [11], which included NOT, NOR, OR, NAND, and AND operations. Since only the first two were compatible with the memory production technology, namely, crossbar arrays, mainly those two were picked up by the community.

In 2016, crossbar compatible AND and NAND operations were proposed [12], which also go by the name of three memristors stateful logic (TMSL). Fig. 1 shows the TMSL AND that we have used in this article to construct our proposed adder. The need for a resistor and an additional voltage source ($V_R \lesseqgtr V_{\text{supply}}$) makes this circuit more complex than MAGIC and somewhat similar to IMPLY in terms of complexity. Nevertheless, it remains practical for implementation, including on crossbar arrays [12]. Performing AND and NAND operations atomically and in a single cycle renders TMSL efficient and important addition to stateful logic literature.

The contribution of fast and energy-efficient logic (FELIX) in 2018 [13] in basic logic was a modified version of MAGIC NOR, which enabled crossbar compatible OR operation. We have used FELIX OR in the adder that we propose in this article. As we see in Fig. 2, FELIX OR has a simple structure and is easy to implement. The list of stateful logics goes on (see [1], [35]); however, the abovementioned designs are a good representative of capital milestones and stateful logic design concepts in the current literature.

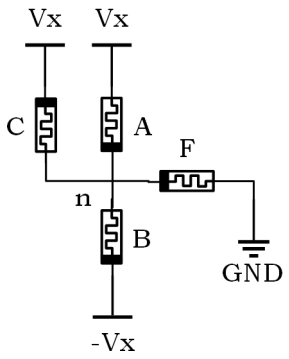

Fig. 3. Proposed single-cycle XOR: basic concept.

TABLE I
Truth Table of XOR

	A	B	F
Case 1	'0'	'0'	'0'
Case 2	'0'	'1'	'1'
Case 3	'1'	'0'	'1'
Case 4	'1'	'1'	'0'

We note that, in FELIX [13], they also proposed a method for performing XOR in two cycles using a particular combination of OR- and NAND-like operations. This way, FELIX reduces the delay of XOR operation by three cycles (from five cycles in [33] to two cycles in [13]). Another attempt to create an XOR/XNOR function is reported in [36]; however, it does not present a substantial advantage to FELIX since it needs two to three cycles to complete. More importantly, it cannot be implemented in homogenous memories since it requires a heterogeneous memristor array (with both unipolar and bipolar memristors).

In continuation of this trend, we complement existing logics by proposing SIXOR, the first atomic stateful XOR operation, which requires only one cycle for completion, uses only common bipolar memristors, and is compatible with crossbar arrays. The proposed XOR can speedup IMCs that use this logic, such as additions, multiplications, search algorithms, image processing, neural networks (NNs), and many more. The speed-up scales up with the increase in the complexity of the overall system and the number of used XOR operations.

III. Proposed Single-Cycle In-Memristor XOR

A. Basic Version

Fig. 3 shows the basic concept of the proposed single-cycle XOR, where A and B are input memristors and F is the output memristor storing the result. Memristor C is an auxiliary memristor whose role we will explain later. Both F and C are initialized to high resistance state (HRS) or logic "0". Therefore, at the beginning of each operation, F and C are—virtually—open circuits.

During the operation, in Cases 1 and 4 of the XOR truth table shown in Table I, given that A and B are in the same state and have a similar resistance, the voltage of the common node, V_n, will be virtually ground, keeping F at its original state, i.e., "0". In Case 3, B is virtually an open circuit,

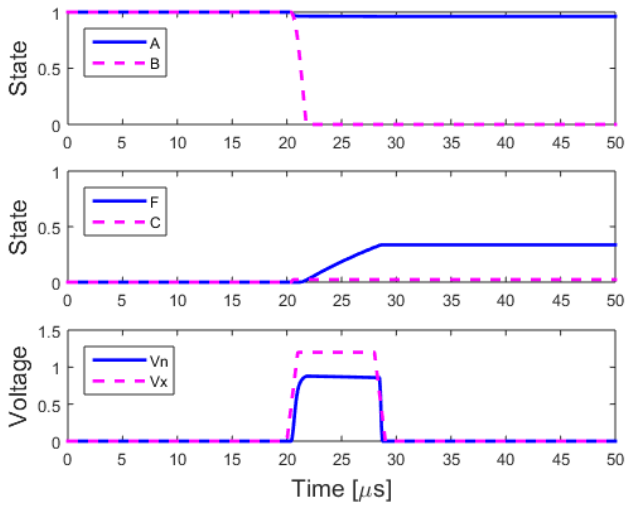

Fig. 4. Simulation result of Case 4 for basic XOR.

and A is virtually short; hence, F sees Vx across it and changes its state from "0" to "1". In Case 2, A is open, and B is short, which brings the common node, n, to $-Vx$. This should not affect F; however, since C sees $2Vx$ across itself, it changes status from "0" to "1". During this process, the current passing through B changes its status from "1" to "0", leaving C the only memristor in low resistance state (LRS) (logic "1"). Since C is now conducting, the voltage of the common node, V_n, rises to approximately Vx, which causes a state change in F, from "0" to "1". Hence, this circuit fulfills all the cases of the truth table of XOR. We note that, in Case 2, the auxiliary memristor C plays a crucial role, without which F would not correctly switch.

Even though this circuit is functional, in Case 4, there is a minor undesired effect. Since both A and B are in their LRS and conduct, the current going through them, as shown in Fig. 4, causes a state change in B. This loss of status brings the circuit practically to Case 3, which initiates a state change in F. Due to limited pulsewidth, this state change in F is not completed, as shown in Fig. 4, even though we observe a significant state drift. Tackling this matter is the topic of Section III-B.

B. Complete Version

To resolve the output drift issue in Case 4 of the circuit proposed in Fig. 3, one could reduce the duration of operation or change the drives. However, those lead to sensitivities to those parameters and their variations, as well as extra complexity or even potentially incompatibility with other timings in the system. Hence, we propose the circuit shown in Fig. 5, where another auxiliary memristor, namely D, which is initiated to "0" (HRS), is added to the circuit.

Given the circuit topology, memristor D can never switch its state and will always remain in HRS. Therefore, whenever memristor B is in LRS ("1"), D is an open circuit in comparison and plays—virtually—no role in the circuit. Whenever B is in its HRS, since B and D will be in the same order of magnitude, their parallel constellation leads to an equivalent

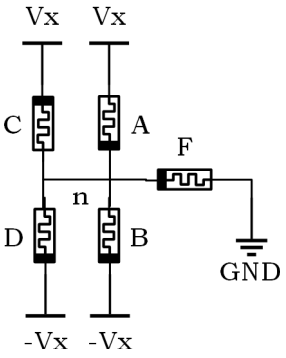

Fig. 5. Proposed single-cycle XOR: complete version.

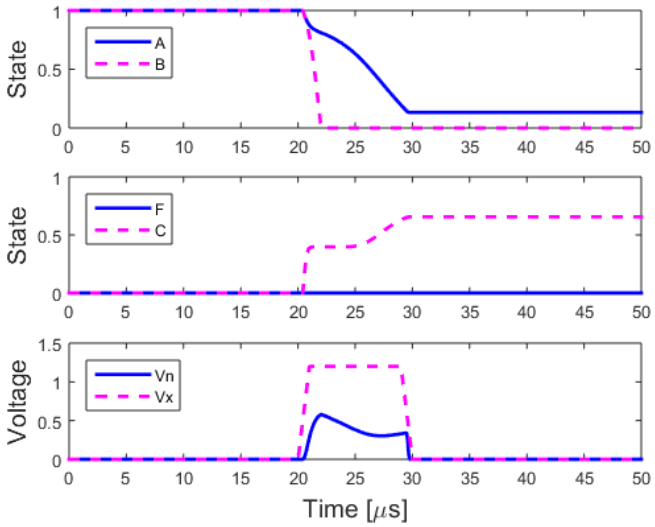

Fig. 6. Simulation result of Case 4 for the complete version of the proposed XOR.

resistance of approximately half their individual values. This lowers the voltage of V_n (particularly, in Case 4, after B changes from "1" to "0") to a voltage below the threshold voltage of F and prevents its drift. Since memristor B and D will share the current between themselves, the state change in B is slowed down, which further delays the undesired rise of the voltage of V_n. This effect contributes to preventing the undesired drift too. The result of this operation can be seen in Fig. 6, where no drift occurs.

We note that, in Case 2, a similar behavior should be observed. However, since in that case, B experiences a much slower change (A is open), F already begins a state change, which further delays the voltage drop at V_n. Thus, as shown in Fig. 7, Case 2 (similar to all other cases) provides a correct output, with no state drift in F. By reducing the duration of operation, as we can see in Fig. 7, an undesired state change in memristor A can be prevented in all cases. However, input memristor B loses its state after the operation in Case 2 and Case 4.

IV. Verification and Formulization

A. Simulations

The proposed design was verified by SPICE simulations in LTSpice. To model the behavior of memristors, we used

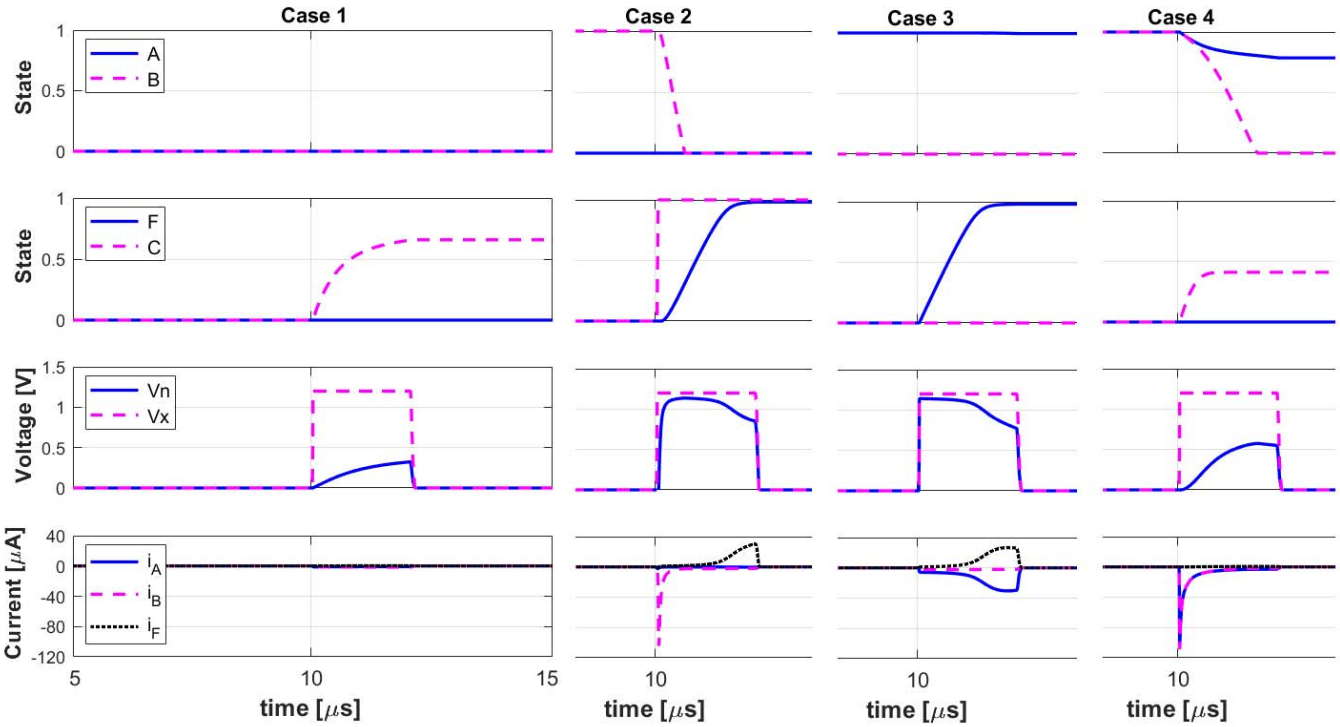

Fig. 7. Simulation results of all cases for the complete version of the proposed XOR.

TABLE II
PARAMETER SETUP USED IN THE VTEAM MODEL, OBTAINED BY FITTING IT TO OUR MEASUREMENTS [8] OF KNOWM "BS-AF-W" DISCRETE MEMRISTORS [40]

Parameter	v_{off}	v_{on}	α_{off}	α_{on}	R_{off}	R_{on}
Value	0.7 V	−10 mV	3	3	1 MΩ	10 kΩ
k_{on}	k_{off}	w_{off}	w_{on}	w_C	a_{off}	a_{on}
−0.5 nm/s	1 cm/s	0 nm	3 nm	100 pm	3 nm	0 nm

the voltage-controlled threshold adaptive memristor (VTEAM) model [37] implemented in SPICE [38], [39]. The parameters used for the model are inserted in Table II. We note that we obtained the model parameters by fitting the VTEAM model to our measurements [8] of Knowm "BS-AF-W" discrete memristors [40]. In all experiments, $V_x = 1.2$ V, and the pulse duration has been 20 μs for Fig. 4 and Fig. 6 and 2 μs for the rest of the simulations. The rise time and fall time of the pulses were set to 5% of the pulse duration. As shown in Fig. 7, the simulations lead to correct results for all cases.

For evaluating the power and energy consumption, we have used the built-in measurement tool of LTSpice. Since the power dissipation depends on the cases and states of input memristors, we have averaged the power consumption over all four combinations of the inputs. On average, for the 2-μs pulse duration, the proposed circuit has a power consumption of 20.25 μW and energy consumption of 44.55 pJ. These numbers include only the power consumption of memristors.

We note that the (minimum) operation delay (here 2 μs), power, and energy consumption of memristive circuits depend heavily on the model—or the technology—used for the implementation of the circuit. Therefore, they will be different and should be adjusted when considering other technology or models. Nonetheless, we provide those numbers to serve as a good base for comparison in the future. The models that we have used are available to the public, and others can use them to simulate their own circuits and obtain comparable numbers for the delay, power, and energy consumption.

B. Formulization of Design Constraints

In this part of the article, we analyze the constraints of each case and extract governing equations and considerations based on those constraints. We bear in mind that, to have any state changes in the F

$$V_x \geq V_{\text{th}}^{\text{on}} \tag{1}$$

must hold.

Other case-specific constraints are as follows.

Case 1 In this case, the main constraint is keeping F in its initial state by ensuring $V_n < V_{\text{th}}^{\text{on}}$, where $V_{\text{th}}^{\text{on}}$ is the threshold voltage of a memristor going from HRS to LRS. Given the symmetry of the circuit, $V_n = 0$ at the beginning and as long as $(V_x - V_n) < 2V_{\text{th}}^{\text{on}}$ or equivalently

$$V_x < 2V_{\text{th}}^{\text{on}} \tag{2}$$

none of the memristors should switch.

The only concern, in this case, is the slow rise of V_n due to a state change in C, which affects the timing. This rise can be faster if V_x has a value close to the maximum dictated by (2). In the proximity of 1.5 $V_{\text{th}}^{\text{on}}$ and the optimum timing (minimum time required for

full state change in F in all cases), this effect is negligible. Even in a few times larger periods, this has no functional effect on the proposed XOR operation.

Case 2 In this case, the primary constraint is

$$V_n \geq V_{\text{th}}^{\text{on}}. \tag{3}$$

To calculate V_n, we need to consider that memristor C sees 2 V_x, which, following (1), is significantly larger than $V_{\text{th}}^{\text{on}}$. Thus, C quickly changes its state to LRS, during which B also changes to HRS. If the parallel resistance of A and C is R_H/k_2, where R_H is the memristance in HRS and k_2 is a constant coefficient, we have

$$V_n \approx \frac{k_2}{k_2+3}V_x + \frac{2}{k_2+3}(-Vx) \geq V_{\text{th}}^{\text{on}}. \tag{4}$$

This gives us

$$Vx \geq \frac{k_2+3}{k_2-2}V_{\text{th}}^{\text{on}} \tag{5}$$

In a technology, such as tungsten chalcogenide resistive random access memorys (ReRAMs) of Knowm [41], where $R_H \approx 100R_L$ (R_L is the memristance in LRS), $k_2 = 101$, and (4) is estimated by V_x. By replacing this in (3), we have—with a very good approximation—the same constraint as (1). To be exact, $V_x \geq 1.05\ V_{\text{th}}^{\text{on}}$.

Case 3 In this case too, we consider the constraints and equations that govern Case 2. The reason is that again one memristor is short to V_x (this time, A itself is short, and C is open), and two open memristors (B and D) are in the path of node n to $-V_x$.

Case 4 The main constraint, in this case, is $V_n < V_{\text{th}}^{\text{on}}$, which is valid at the beginning of the cycle since $V_n = 0$ due to the symmetry of the circuit (the same value of resistance connects node n to V_x and $-V_x$, creating a virtual ground at that node). However, this symmetry is broken when, due to the passing current, B loses its state and draws current from A and C. Assuming that here $R_A||R_C = (R_H/k_4)$, where k_4 is a coefficient representing the drift of memristor A and C during the operation, we have

$$V_n \approx \frac{k_4}{k_4+3}V_x + \frac{2}{k_4+3}(-Vx) < V_{\text{th}}^{\text{on}} \tag{6}$$

or

$$V_x < \frac{k_4+3}{k_4-2}V_{\text{th}}^{\text{on}}. \tag{7}$$

We know that both A and C have drifted from LRS and HRS, and we cannot estimate k_4 with 100, especially given the highly nonlinear nature of memristors. In the example case of Fig. 7, $R_A = 0.18R_H$ and $R_C = 0.6R_H$, which gives $k_4 = 7.2$. This means that $V_x < 1.96\ V_{\text{th}}^{\text{on}}$ must hold.

In summary, all design constraints can be expresses as

$$V_{\text{th}}^{\text{on}} \leq \frac{k_2+3}{k_2-2}V_{\text{th}}^{\text{on}} \leq V_x < \frac{k_4+3}{k_4-2}V_{\text{th}}^{\text{on}} < 2V_{\text{th}}^{\text{on}} \tag{8}$$

where k_2 and k_4 are technology dependent values. As a rule of thumb, it is reasonable to assume roughly one order of magnitude difference between the two. In the example case of Knowm [41] and our simulation setup, $k_2 = 100$ and $k_4 = 7.2$, leading to

$$1.05V_{\text{th}}^{\text{on}} \leq V_x < 1.96V_{\text{th}}^{\text{on}}. \tag{9}$$

V. Showcase: Full-Adder

A. Design

To showcase the impact of SIXOR, we present a new stateful in-memory full-adder (FA) that uses the proposed XOR to calculate a single bit addition in only four cycles. In this adder, the sum (S_{HA}) and carry (C_{HA}) of the half-adder (HA) are the direct implementation of

$$1:\ C_{\text{HA}} = A \cdot B \tag{10}$$

$$2:\ S_{\text{HA}} = A \oplus B \tag{11}$$

using the proposed XOR and TMSL AND, taking two cycles only. From there to full addition, first, an interim value (Int) needs to be calculated in the third cycle

$$3:\quad Int = C_{\text{in}} \cdot S_{\text{HA}} \tag{12}$$

$$\text{Re-initialize } B \ \&\ C_{\text{aux}} \tag{13}$$

Where, again, the TMSL AND is used. At this step, in parallel, we reinitialize B to store the final sum in it. In addition, C_{aux}, the auxiliary C memristor in Fig. 5, may have lost its state during the first XOR and is reinitialized. In the fourth cycle, two independent operations can run in parallel

$$4:\ S_{\text{FA}} = C_{\text{in}} \oplus S_{\text{HA}} \tag{14}$$

$$C_{\text{FA}} = C_{\text{HA}} + Int \tag{15}$$

where FELIX OR is used in addition to our proposed SIXOR, therefore finishing a single-bit full-addition in four cycles only. This FA would need ten memristors; however, since memristor B loses its original state, B can store the output. Thus, we can reduce the number of required memristors to nine only.

Algorithm 1 shows how this can be extended to an n-bit addition, which requires $2n + 2$ cycles and $6n + 3$ memristors. To keep the number of memristors down, we propose reusing B to store the sum and using the same memristors for Int and C in each iteration of the loop.

B. Validation

We validated the proposed adder design through SPICE simulations using the VTEAM model [37] implemented in SPICE [38], [39]. Model settings are shown in Table II (fitted to our measurements [8] of Knowm "BS-AF-W" discrete memristors [40]). In these simulations, $V_{\text{supply}} = V_x = 1.3v$ and $t_{\text{pulse}} = 2\ \mu$s. For the TMSL AND, additional parameters are $V_R = 0.6$ and $R = 16\ k\Omega$. All input combinations led to correct outputs. Fig. 8 shows sample simulation results for $A = 1, B = 1$, and $C_{\text{in}} = 1$. For this experiment, every 10 μs, a 2-μs pulse has been applied to respective memristors to perform one of the steps of addition. That is, from 10 to 12 μs, memristors A and B have been ANDed and stored

Algorithm 1 Proposed n-Bit Full-Adder

Input: A, B, C_{in} =C_0
Output: S, C_{out}
Initialization
1: C_{HA} = A AND B
2: S_{HA} = A XOR B
LOOP Process
for $i = 0$ to n **do**
3: Int_i = $S_{HA,i}$ AND C_i
4: C_{i+1} = Int_i OR $C_{HA,i}$
S_i = C_i XOR $S_{HA,i}$
end for
return $S, C_{out} = C_{n+1}$

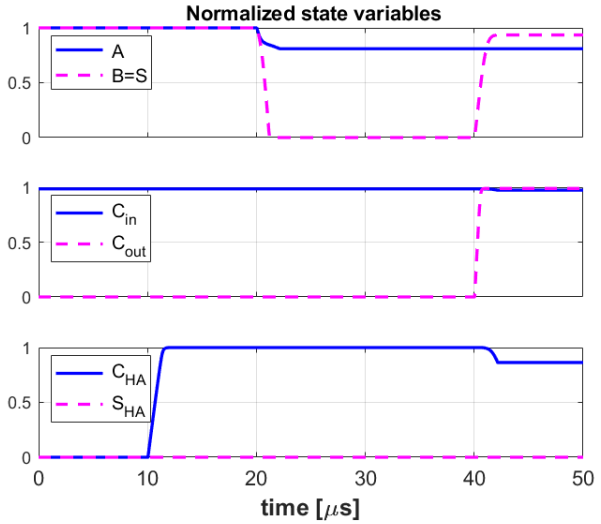

Fig. 8. Simulation results of addition for $A = 1, B = 1, C_{\text{in}} = 1$, leading to $S = 1$ and $C_{\text{out}} = 1$.

in C_{HA}, according to (10). From 20 to 22 μs, A and B have been XORed and stored in S_{HA}, according to (11). During the 30–32-μs period, B and C_{aux} are reinitialized, as in (13), while Int is calculated according to (12). During 40–42-μs, two operations, namely (14) and (15), are conducted simultaneously, leading to the calculation of the final sum and carry out.

C. Comparison: Single-Bit

Table III shows a list of stateful FAs published in the last three years. As we can see in this table, the proposed FA is the fastest adder with a significant margin compared to other FAs. Due to the XOR operations proposed here, the proposed adder is, on average, 2.94 times faster than other FAs. Compared to the fastest competitor, namely, FELIX FA [13], the proposed FA achieves a 50% improvement in speed while using the same number of memristors. On average, the proposed adder uses 43% more memristors than other designs, which is negligible compared to the 294% average speed-up gained.

Table III also provides the reported power consumption of different FAs. We note that, for a fair comparison, both circuits should be simulated (or implemented) using the same memristive technology since technological parameters affect

TABLE III
RECENT STATEFUL SINGLE-BIT FULL-ADDERS

Designs	# of Step	Number of Memristors	Power [μW]	Energy [nJ]
Iterative [42]	18	8	NA[?]	NA[?]
IMPLY Serial [43]	22	5	NA	NA
IMPLY Parallel [44]	21	5	NA	NA
FELIX [13]	6	9	NA	0.135[$]
Semi-Serial [39]	12	8	329	9.78
Semi-Parallel [45]	17	5	199	9.98
ORNOR [35]	17	8	133[@]	0.43[@]
Cascading Logic [46]	13	7	NA	NA
Proposed	4	9	25.31	0.22

[?] From the discussion provided on power, we could not understand what is the actual average power consumption of a single-bit adder in this design.
[@, $] These numbers are not directly comparable with others since they are based on different memristive technologies.
[@] The power consumption is calculated using the energy consumption, the number of cycles, and the pulse-width of each cycle.
[$] We could not find the pulse-width used in this design and hence, could not calculate the power consumption.

delay, power, and energy consumption to a considerable extent. For example, our model and, consequently, our simulations are based on Knowm discrete devices. Discrete devices have substantial parasitics that considerably increase their delay and energy consumption compared to integrated devices. Similarly, the different material composition of memristors leads to considerably different features (e.g., HRS, LRS, threshold voltages, and switching speed), as well as delay and power consumption behaviors. Hence, the absolute numbers of delay, power, and energy consumption of various circuits cannot be directly compared if they are not based on the same technology. In the case of here proposed SIXOR, these numbers will significantly decrease when implemented (or simulated) in an integrated fashion.

Among the designs listed in Table III, TaheriNejad *et al.* [39] and Ganjeheizadeh Rohani *et al.* [45] use the same memristor model and use the same parameter values in the model. Therefore, these two designs can be fairly compared with the proposed design. We see that the proposed design has a significantly lower energy and power footprint than those other two designs. Two main contributing factors are the use of the new and different logics and the significantly smaller number of steps (which is a consequence of using the newly proposed XOR logic). We believe that, for the same reasons, particularly the reduced number of steps, other circuits and systems that use the proposed SIXOR will benefit from energy savings.

D. Comparison: Multibit

We note that a single-bit FA is not fully representative of the practical impacts of a design since it is rarely used as such. Therefore, in Table IV, we have inserted the number of steps and memristors for an n-bit adder, as it would be used in practice. We have also calculated the number of steps and memristors for a 32-bit addition as an example. Some designs in Table III do not explain how they would expand their design from a single-bit FA to an n-bit adder and how many steps

TABLE IV
SUMMARY OF STATEFUL ADDERS

Designs	Number of Steps		Number of Memristors	
	Total	$n = 32$	Total	$n = 32$
Iterative [42]	$21n - 3$	669	$8n$	256
IMPLY Serial [43]	$22n$	704	$2n + 3$	67
IMPLY Serial [44]	$23n$	736	$2n + 3$	67
IMPLY Parallel [44]	$5n + 16$	176	$4n + 1$	129
Semi-Parallel [45]	$17n$	544	$2n + 3$	67
Semi-Serial [39]	$10n + 2$	322	$2n + 6$	70
ORNOR [35]	$2n + 15$	79	$6n + 6$	198
Proposed	$2n + 2$	66	$6n + 3$	195

or memristors would be required in that case. For this reason, they are not included in Table IV.

Comparing Tables III and IV, we can see that the difference in performance has a considerable change from single-bit to n- or, in the example of Table IV, 32-bit adders. For instance, the Iterative design [42] is 3.5 times slower than the proposed FA and uses 12.5% fewer memristors. However, in a 32-bit addition, that design is 9.1 times slower while using 31% more memristor, thus having no advantage either in the number of steps or number of memristors. We observe that this is the general trend of performance change between single-bit FA and the n-bit adder proposed here. That is, the gap in the number of used memristors closes (by reducing the extra memristor usage by an average of 37%) and the gap in the number of steps increases (by improving the speedup by an average of six times) for the 32-bit adders in Table IV. These improvements happen due to the single-cycle XOR operation, which allows faster computations.

To have a uniform overview of the merit of the proposed design, we consider two figures of merit (FoMs), as defined in [47]. These figure of merits (FoMs) consider a combination to speed and area or the number of memristors to evaluate the merit of a design. In a balanced design, where the number of steps (n_S) and number of memristors (n_M) have the same importance, the FoM of all designs can be calculated using [47]

$$\text{FoM}_B = \frac{1}{n_M \cdot n_S}. \tag{16}$$

For a 32-bit adder of the proposed design, $\text{FoM}_B = 77\ \mu$, which is between 21% and 12× better than all other adders in Table IV. On average, FoM_B of the proposed design is three times better than other adders.

The main goal and advantage of the proposed XOR operation is to speedup the computation. Hence, if we consider the *speed-centered* FoM [47]

$$\text{FoM}_S = \frac{1}{n_M \cdot n_S^2} \tag{17}$$

the minimum and average improvements of the proposed 32-bit adder compared to other adders in Table IV are 45% and 35×, respectively. This demonstrates the merit of the proposed adder and, consequently, the advantages of using SIXOR.

VI. DISCUSSION

A. Design Considerations

One of the important advantages of the proposed XOR is its ease of design and robustness. This robustness was shown by the correct operation under various voltages (1.05 $V_{\text{th}}^{\text{on}}$ to 1.96 $V_{\text{th}}^{\text{on}}$) and timings (the operation length was changed for one order of magnitude from 2 to 20 μs. As we saw in Section IV-B, the design constraints are summarized into (8), further simplified for our target technology to (9). This allows for a relatively wide range of values to select V_x from, namely, 0.91 $V_{\text{th}}^{\text{on}}$. Here, we first discuss how this range should be calculated in other technologies. Next, we discuss the state loss in memristor B.

Regarding the design constraints in other technologies, we have

$$k_2 = \frac{R_H}{R_A || R_C} \tag{18}$$

where R_A and R_C are the resistance of memristors at the end of the operation in Case 2. At the end of Case 2, memristor A is in HRS, and memristor C is in LRS, which gives us

$$k_2 = \frac{R_H}{R_H || R_L} = \frac{R_H + R_L}{R_L}. \tag{19}$$

The same formulation as (18) can be used for k_4, i.e.,

$$k_4 = \frac{R_H}{R_A || R_C} \tag{20}$$

with the difference that R_A and R_C are the resistance of memristors at the end of the operation in Case 4. At the end of Case 4, neither of the memristors is in LRS or HRS. Memristor A is close to HRS, and memristor C is close to LRS. The value of k_4 is, thus, decided based on the acceptable drift or deviation from LRS and HRS. Therefore, k_4 presents itself as the main parameter that the design engineer should decide at design time. In doing so, they should consider that this drift limits the range of allowable supply voltages (V_x). However, as we saw in our example, targeting a very small drift allows for a reasonably large range of supply voltage. Therefore, no complex decision process is necessary here.

Another design consideration is the loss of state for input memristor B in Case 2 and Case 4. A known loss of state is acceptable since it can be taken into account and mitigated at design time. We see a similar concept in IMPLY, where at least one and, in many cases, both inputs lose their states [48]. As shown in Section V, a careful design (proper ordering of operations) can render this loss of state during the XOR operation inconsequential. In that example, XORs are performed when the inputs were not used in further steps anymore. If such an ordering is not possible in a circuit, or for any reason, the input is necessary for further calculations, a copy of B can be stored prior to applying XOR. However, in many cases (e.g., intermediate results), this is not necessary. This further use or lack thereof can be easily determined at the compiler level, thus preventing any extra load for the operations and respective performance.

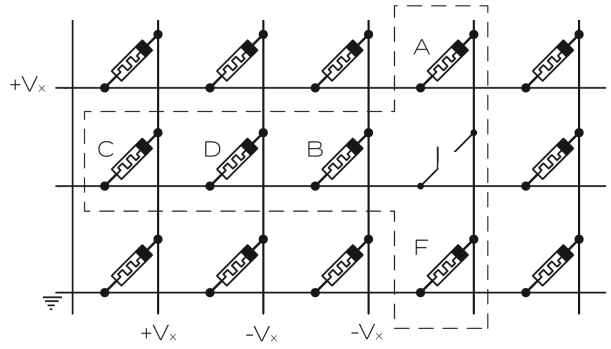

Fig. 9. Schematic of a 2-D implementation of the proposed XOR.

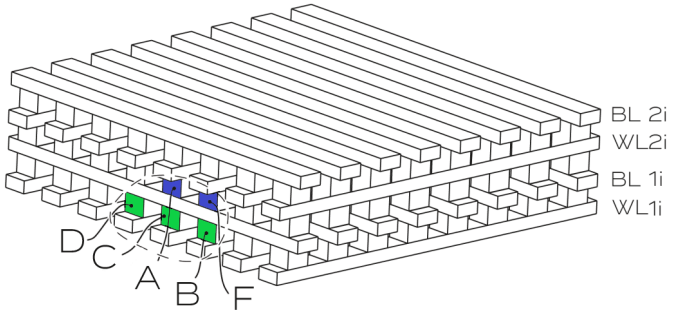

Fig. 10. Schematic of a 3-D implementation of the proposed XOR, where memristors A and F are highlighted in blue and others in green.

B. Crossbar Compatibility

A critical aspect of memristive circuits is their compatibility with crossbar implementation. The main reason is that, as memory arrays, a crossbar implementation is the most-likely and most compact implementation for an integrated circuit. It needs no extra clarification on how the proposed circuit can be implemented using discrete devices. Therefore, we discuss the implementation of SIXOR on 2-D and 3-D memristive crossbar arrays.

Fig. 9 shows a schematic of how the proposed XOR can be implemented on a 2-D memristive crossbar array. As we see in this figure, due to the polarity of memristors (A and F, being different from the three other memristors), they cannot be implemented all in the same row (or the same column).[1] Therefore, a switch is needed to connected the row (or column) where B, C, and D are located to a column (or a row), where A and F are located. We have symbolically shown this switch on the crossbar. In practice, this switch will be implemented on the periphery of the crossbar, inside the control and read/write circuitry fabricated on CMOS. Although it is preferred to avoid additional switches, they have been used in many memristive circuits and systems (e.g., [13], [44], [47], [49]) since the benefit of using them can more than justify the imposed additional resources (in our case, the speed gain due to performing XOR in a single cycle only).

Given how most memristors, particularly ReRAMs, are fabricated, it is more likely that the crossbar arrays be fabricated in 3-D stacks (as opposed to 2-D, which implies only a single layer of memristors). This domain is little explored, and the literature on 3-D memristive circuits and logics is relatively sparse [1]. The proposed SIXOR is fully compatible with 3-D crossbars and achieves further efficiency on 3-D arrays. Fig. 10 shows how SIXOR can be implemented on a 3-D array, where memristors A and F are highlighted in blue, whereas B, C, and D are depicted in green. As we see in Fig. 10, the 3-D implementation of SIXOR does not need any external switches since the memristors in the blue layer and the green layer are by default connected with antiparallel polarities. In the 3-D crossbar, SIXOR occupies a more limited number of rows and columns, allowing for further parallelism and simultaneous computations.

[1]We note that the position of memristors in rows and columns can be exchanged. That is, B, C, and D could be located on the same column, while A and F sit on one row. From these two possibilities, we have shown only one and mention the other possibility inside parentheses.

VII. Implementation Aspects

To implement the proposed circuits on real memristors, additional aspects need to be taken into consideration. Some of these issues (such as sneak paths, memristance variation, and leakage) are general concerns in the community and not specific to this work only. For instance, statistical variation in memristance of the devices and the leakage phenomenon (state drift in the absence of stimuli) affect other existing stateful logics too (see [50] for an example) and in each case differently. However, they have not been thoroughly studied since they require new models that reflect these practical aspects. We are currently developing a model that will enable us to conduct such studies for all logics in the literature. Moreover, such an in-depth discussion requires its own venue, especially that it would be the first in the literature to evaluate all stateful logics in that regard. Therefore, we leave those two aspects for future publications.

A. Negative Voltage Supply

One implementation point to consider is that the proposed XOR requires both positive (V_x) and negative ($-V_x$) supply voltages. A simple solution for this requirement, using a single-pole input supply, is to have a supply voltage of $2V_x$ (or larger) and connect the positive supply to the positive pole and the negative supply to the negative pole. In this case, $V_{\text{supply}}/2$ will be the virtual GND node, and memristors should be able to switch with $(V_{\text{supply}}/2)$, i.e., $(V_{\text{supply}}/2) \geq V_{\text{th}}^{\text{on}}$. For memristors with smaller $V_{\text{th}}^{\text{on}}$ (e.g., 0.1–0.5 V), this would be no problem. However, with a memristive technology with a relatively high $V_{\text{th}}^{\text{on}}$ (e.g., 0.7 V), a relatively large voltage supply (in this example, 1.4–2.8 V) would be necessary. This requirement can be challenging.

Large voltage ranges limit the CMOS technology nodes that can be used for the peripheral and control circuits. Smaller technology nodes cannot be used since they cannot operate in such high voltage ranges (≥ 1.5). Using larger technology nodes can be acceptable for certain applications, such as memory chips, which generally operates at lower frequencies. Since they do not need high frequencies, they can use larger technology nodes, which are more economical. However, this is not ideal for the integration in processor chips, especially

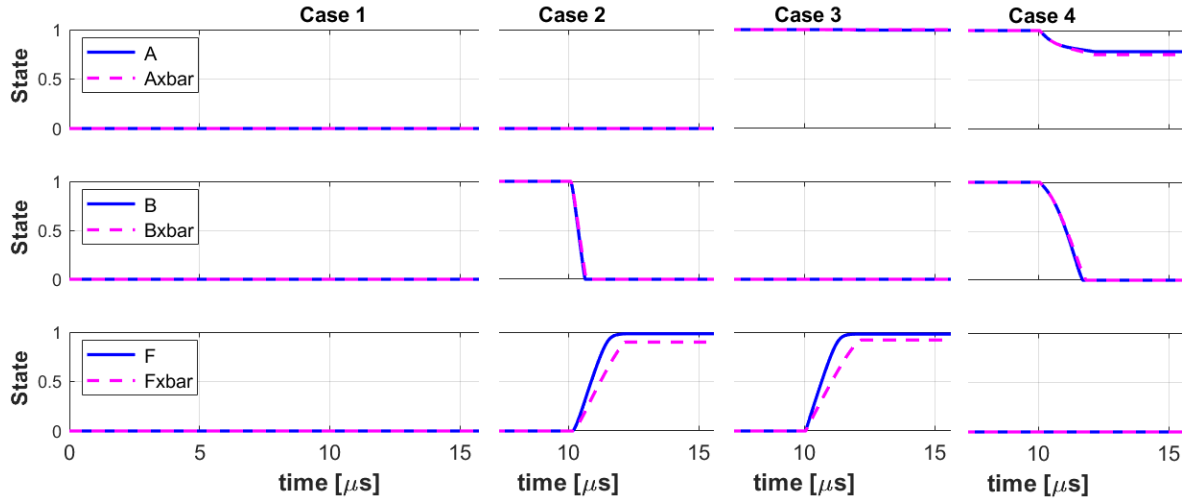

Fig. 11. Comparison of simulation results of all cases for the proposed XOR stand-alone (shown in solid blue lines) and in an 8×8 crossbar array (shown in dashed lines and marked with xbar).

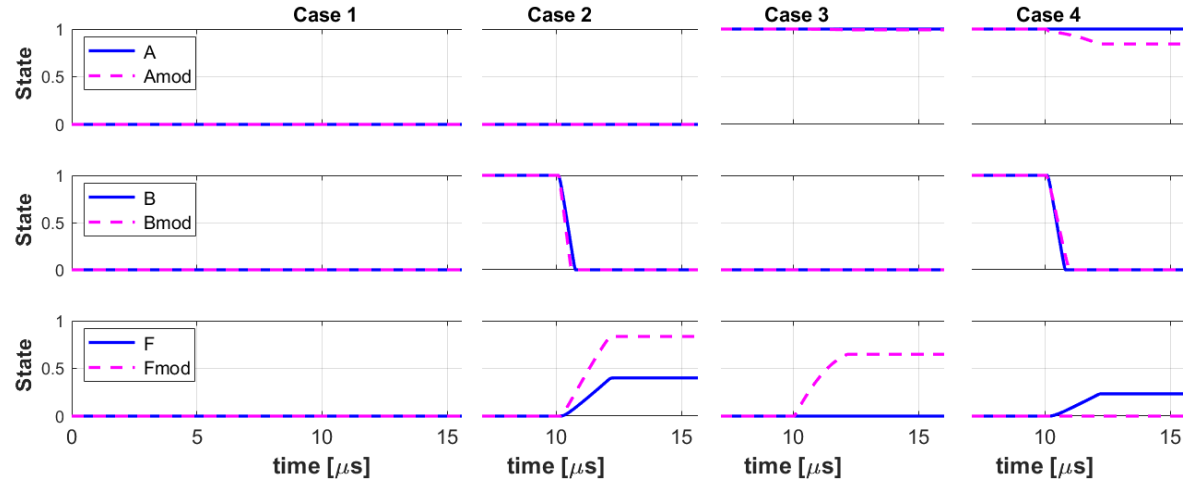

Fig. 12. Comparison of simulation results of all cases for the proposed XOR in a 16×16 crossbar array (shown in solid blue lines) and the 16×16 crossbar array with modifications (shown in dashed lines); that is, application of bias voltages to unused rows and columns as opposed to leaving them float.

high-end processors, which are fabricated on the smallest of technology nodes to meet the demand for speed.

To tackle this problem, an alternative solution can be used, namely, negative voltage generators and charge pumps that produce negative supply voltage on-chip (see [51]–[53]). This would allow using a power supply with a voltage as low as V_x and produce $-V_x$ from the same power supply on-chip. Using this approach, which is used for Dynamic Random Access Memorys (DRAMs) too [51], [52], smaller technology nodes can be used. We note that, for all logics (even those using a single-pole voltage), the technology node should support voltages higher than the $V_{\text{th}}^{\text{on}}$.

B. Sneak Paths

In larger integration of memristive structures with no dedicated switch for each memristor, such as 1R (1M) crossbar arrays, sneak path is an important concern [54], [55]. Therefore, many researchers in the community have tried to study [55]–[58] and tackle [26], [54], [56] this problem. To study this effect, we have simulated the proposed SIXOR in crossbar arrays for functionality validation. In the rest of this section, we present and discuss our simulations of 4×4, 8×8, and 16×16.1 R (1M) crossbar arrays. For these simulations, unused rows and arrays were left floating. In the 4×4 array, no considerable difference was observed, and only a slight difference was observed in the 8×8 array. As shown in Fig. 11, this difference is too small to affect the functional validity of the XOR operation. In the 16×16 array, however, as shown with solid blue lines in Fig. 12, the sneak paths already affect the functional validity of the XOR operation. To counter this effect, we slightly increased the V_x (from 1.2 to 1.3 V), connected the unused rows to $V_x/2$, and unused columns to the ground. This way, unused memristors see $V_x/2$, which, following (2), is smaller than $V_{\text{th}}^{\text{on}}$ and does not change their states. These actions reduce the undesired contributions of unused memristors to sneak paths and thus, as shown with dashed magenta lines in Fig. 12, make the XOR functional again.

As shown above, in cases where the sneak paths negatively affect the functional validity of the operation of a circuit, some strategies can be deployed to help. Strategies such as the judicious application of voltages to unused columns and rows can limit the sneak path effect. Our proposed circuit uses HRS as the initial value of memristors, which reduces the sneak paths current compared to those with LRS as their initial value. Nevertheless, 1R (1M) crossbar arrays are very difficult to handle [59]–[61], and in many cases in the literature, sneak paths have disrupted the functionality of implemented systems. Hence, even though we have not verified it, there is a chance that, in larger array sizes, the sneak paths may affect the functional validity of the proposed XOR, despite the above-used strategies. This common issue in the field has led to 1T1R (1T1M) being currently the dominant and preferred implementation method [59]–[61]. In 1T1R (1T1M) crossbar arrays, each memristor is in series with a transistor as the select switch. Hence, if a memristor is not participating in an operation, the transistor switch is in the off state, is virtually open circuit, and prevents sneak path currents going through that memristor. Following the current state-of-the-art (SoA), we propose 1T1R as the preferred method for implementing our circuit in larger crossbar arrays.

VIII. Conclusion

In this article, we proposed SIXOR: a new single-cycle XOR for in-memory computations using memristors. We presented the design constraints for SIXOR and discussed various aspects of design and implementation. By saving on the number of cycles necessary to run an XOR operation, the proposed SIXOR contributes to speeding up many IMC applications. We showcased this effect by designing a new adder using the proposed SIXOR. This adder proved to be the fastest stateful adder thus far, achieving a minimum of 50% and up to 4.5× improvement in that regard compared to other SoA stateful adders. Moreover, we observed that the proposed adder is significantly more energy efficient in contrast to other comparable designs due to the faster calculation and reduced number of steps made possible by SIXOR. We expect such an impact on speed and energy consumption to be achievable in a large variety of other circuits and systems to be designed in the future using the proposed single-cycle XOR.

References

[1] N. Xu *et al.*, "A stateful logic family based on a new logic primitive circuit composed of two antiparallel bipolar memristors," *Adv. Intell. Syst.*, vol. 2, no. 1, Jan. 2020, Art. no. 1900082.

[2] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electron.*, vol. 1, no. 1, pp. 22–29, Jan. 2018.

[3] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnol.*, vol. 8, pp. 13–24, Jan. 2013.

[4] H. Kim, M. P. Sah, C. Yang, and L. O. Chua, "Memristor-based multilevel memory," in *Proc. 12th Int. Workshop Cellular Nanosc. Netw. Their Appl. (CNNA)*, Feb. 2010, pp. 1–6.

[5] M. Zangeneh and A. Joshi, "Design and optimization of nonvolatile multibit 1T1R resistive RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 8, pp. 1815–1828, Aug. 2014.

[6] N. Taherinejad, P. D. S. Manoj, and A. Jantsch, "Memristors' potential for multi-bit storage and pattern learning," in *Proc. IEEE Eur. Modeling Symp. (EMS)*, Oct. 2015, pp. 450–455.

[7] N. Taherinejad, M. P. D. Sai, M. Rathmair, and A. Jantsch, "Fully digital write-in scheme for multi-bit memristive storage," in *Proc. 13th Int. Conf. Electr. Eng., Comput. Sci. Autom. Control (CCE)*, Sep. 2016, pp. 1–6.

[8] D. Radakovits and N. TaheriNejad, "Implementation and characterization of a memristive memory system," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–5.

[9] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *Proc. IEEE/ACM Int. Symp. Nanosc. Architectures*, Jul. 2009, pp. 33–36.

[10] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. Stanley Williams, "'Memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, pp. 873–876, Apr. 2010.

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

[12] P. Huang *et al.*, "Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits," *Adv. Mater.*, vol. 28, no. 44, pp. 9758–9764, 2016.

[13] S. Gupta, M. Imani, and T. Rosing, "FELIX: Fast and energy-efficient logic in memory," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 1–7.

[14] S. Li *et al.*, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proc. 53rd Annu. Design Automat. Conf.*, Jun. 2016, pp. 1–6.

[15] P.-E. Gaillardon *et al.*, "The programmable logic-in-memory (PLiM) computer," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2016, pp. 427–432.

[16] G. Papandroulidakis, I. Vourkas, A. Abusleme, G. C. Sirakoulis, and A. Rubio, "Crossbar-based memristive logic-in-memory architecture," *IEEE Trans. Nanotechnol.*, vol. 16, no. 3, pp. 491–501, May 2017.

[17] M. R. Alam, M. H. Najafi, and N. T. Nejad, "Exact stochastic computing multiplication in memristive memory," *IEEE Des. Test*, early access, doi: 10.1109/MDAT.2021.3051296.

[18] M. Riahi Alam, M. Hassan Najafi, and N. TaheriNejad, "Sorting in memristive memory," 2020, *arXiv:2012.09918*. [Online]. Available: http://arxiv.org/abs/2012.09918

[19] Neuro-Bit. (2017). *Bio Inspired Technologies LLC*. [Online]. Available: http://www.bioinspired.net/home.html

[20] Knowm. (2017). *Knowm Inc*. [Online]. Available: https://knowm.org

[21] G. Hilson. (2015). *Panasonic Push Progress on ReRAM*. eeTimes. [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1327307

[22] ReRAM. (2017). *Crossbar*. [Online]. Available: https://www.crossbar-inc.com/en/

[23] P. Clarke. (2020). *TSMC Offers 22 nm RRAM, taking MRAM to 16 nm*. eeNews. [Online]. Available: https://www.eenewsanalog.com/news/tsmc-offers-22nm-rram-taking-mram-16nm

[24] I. Vourkas and G. Sirakoulis, *Memristor-Based Nanoelectronic Computing Circuits and Architectures: Foreword by Leon Chua* (Emergence, Complexity and Computation). Cham, Switzerland: Springer, 2015. [Online]. Available: https://www.springer.com/gp/book/9783319226460

[25] S. Muroga, *Threshold Logic and Its Applications*. Hoboken, NJ, USA: Wiley, 1971.

[26] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature Mater.*, vol. 9, no. 5, pp. 403–406, May 2010.

[27] 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, Feb. 2009.

[28] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Memristor-based IMPLY logic design procedure," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Oct. 2011, pp. 142–147.

[29] E. Linn *et al.*, "Beyond von Neumann—Logic operations in passive crossbar arrays alongside memory operations," *Nanotechnology*, vol. 23, no. 30, 2012, Art. no. 305205.

[30] L. Gao, F. Alibart, and D. B. Strukov, "Programmable CMOS/Memristor threshold logic," *IEEE Trans. Nanotechnol.*, vol. 12, no. 2, pp. 115–119, Mar. 2013.

[31] G. Snider, "Computing with hysteretic resistor crossbars," *Appl. Phys. A, Solids Surf.*, vol. 80, no. 6, pp. 1165–1172, Mar. 2005.

[32] A. Siemon, S. Menzel, R. Waser, and E. Linn, "A complementary resistive switch-based crossbar array adder," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 1, pp. 64–74, Mar. 2015.

[33] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided loGIC (MAGIC)," *IEEE Trans. Nanotechnol.*, vol. 15, no. 4, pp. 635–650, Jul. 2016.
[34] L. Xie *et al.*, "Scouting logic: A novel memristor-based logic design for resistive computing," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 176–181.
[35] A. Siemon *et al.*, "Stateful three-input logic with memristive switches," *Sci. Rep.*, vol. 9, no. 1, p. 14618, Dec. 2019.
[36] M. Abu Lebdeh, H. Abunahla, B. Mohammad, and M. Al-Qutayri, "An efficient heterogeneous memristive xnor for in-memory computing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2427–2437, Sep. 2017.
[37] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.
[38] D. Radakovits, M. Jungwirth, S. M. Laube, and N. TaheriNejad. (Sep. 2019). *Second (v2.0) LTSpice Implementation of VTEAM*. [Online]. Available: https://www.ict.tuwien.ac.at/staff/taherinejad/projects/memristor/files/vteam2.asc
[39] N. TaheriNejad, T. Delaroche, D. Radakovits, and S. Mirabbasi, "A semi-serial topology for compact and fast IMPLY-based memristive full adders," in *Proc. 17th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2019, pp. 1–5.
[40] Knowm. (2017). *Knowm BS-AF-W Memristor Datasheet*. [Online]. Available: https://knowm.org/wp-content/uploads/DM8-16DIP-BS-AF-W.pdf
[41] (Jun. 2018). *Datasheet of Knowm BS-AF-W Memristor*. [Online]. Available: https://knowm.org/wp-content/uploads/dm8-16dip-bs-af-w.pdf
[42] K. C. Rahman *et al.*, "Memristor based 8-bit iterative full adder with space-time notation and sneak-path protection," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 695–698.
[43] S. G. Rohani and N. T. Nejad, "An improved algorithm for IMPLY logic based memristive full-adder," in *Proc. IEEE 30th Can. Conf. Elect. Comput. Eng. (CCECE)*, Apr./May 2017, pp. 1–4.
[44] A. Karimi and A. Rezai, "Novel design for a memristor-based full adder using a new IMPLY logic approach," *J. Comput. Electron.*, vol. 17, no. 3, pp. 1303–1314, Sep. 2018, doi: 10.1007/s10825-018-1198-5.
[45] S. Ganjeheizadeh Rohani, N. Taherinejad, and D. Radakovits, "A semi-parallel full-adder in IMPLY logic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 297–301, Jan. 2020.
[46] K. M. Kim and R. S. Williams, "A family of stateful memristor gates for complete cascading logic," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4348–4355, Nov. 2019.
[47] D. Radakovits, N. Taherinejad, M. Cai, T. Delaroche, and S. Mirabbasi, "A memristive multiplier using semi-serial imply-based adder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, pp. 1495–1506, May 2020.
[48] N. TaheriNejad and D. Radakovits, "From behavioral design of memristive circuits and systems to physical implementations," *IEEE Circuits Syst. Mag.*, vol. 19, no. 4, pp. 6–18, 4th Quart., 2019.
[49] M. R. Alam, M. H. Najafi, and N. TaheriNejad, "Exact in-memory multiplication based on deterministic stochastic computing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.
[50] S. Michael Laube and N. TaheriNejad, "Device variability analysis for memristive material implication," 2021, *arXiv:2101.07231*. [Online]. Available: http://arxiv.org/abs/2101.07231
[51] K.-S. Min and J.-Y. Chung, "A fast pump-down V_{BB} generator for sub-1.5-V DRAMs," *IEEE J. Solid-State Circuits*, vol. 36, no. 7, pp. 1154–1157, Jul. 2001.
[52] Y. Tan, C. Zhan, and G. Wang, "A fully-on-chip analog low-dropout regulator with negative charge pump for low-voltage applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 8, pp. 1361–1365, Aug. 2019.
[53] C. Lee, T. Yim, and H. Yoon, "A negative charge pump using enhanced pumping clock for low-voltage DRAM," *Electronics*, vol. 9, no. 11, p. 1769, Oct. 2020.
[54] M. A. Zidan, A. M. Eltawil, F. Kurdahi, H. A. H. Fahmy, and K. N. Salama, "Memristor multiport readout: A closed-form solution for sneak paths," *IEEE Trans. Nanotechnol.*, vol. 13, no. 2, pp. 274–282, Mar. 2014.
[55] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 156–160.
[56] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Trans. Electron Devices*, vol. 60, no. 4, pp. 1318–1326, Apr. 2013.
[57] S. Shin, K. Kim, and S.-M. Kang, "Analysis of passive memristive devices array: Data-dependent statistical model and self-adaptable sense resistance for RRAMs," *Proc. IEEE*, vol. 100, no. 6, pp. 2021–2032, Jun. 2012.
[58] S. Shin, K. Kim, and S.-M. Kang, "Data-dependent statistical memory model for passive array of memristive devices," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 12, pp. 986–990, Dec. 2010.
[59] H. J. Wan *et al.*, "*In situ* observation of compliance-current overshoot and its effect on resistive switching," *IEEE Electron Device Lett.*, vol. 31, no. 3, pp. 246–248, Mar. 2010.
[60] C. Li *et al.*, "In-memory computing with memristor arrays," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2018, pp. 1–4.
[61] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, no. 1, p. 52, Jan. 2018.

Nima TaheriNejad (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, BC, Canada, in 2015.

He is currently a "Universitätsassistent" at the TU Wien (formerly known as Vienna University of Technology as well), Vienna, Austria, where his areas of work include in-memory computing, self-awareness in resource-constrained cyber–physical embedded systems, computer architecture, systems on chip, memristor-based circuit and systems, and health care. He has authored two books and more than 60 peer-reviewed articles.

Dr. TaheriNejad has also served as a reviewer and an editor for various journals, conferences, as well as an organizer and the chair for various conferences and workshops. He has received several awards and scholarships from universities and conferences he has attended.