2022 20th IEEE Interregional NEWCAS Conference (NEWCAS) | 978-1-6654-0105-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/NEWCAS52662.2022.9841969

Operational Conditions Analysis for Memristive Stateful Logics - A Study on IMPLY and TMSL

Roya Rahimi Disfani*, Nima TaheriNejad†, and Mojtaba Valinataj*
* Babol Noshirvani University of Technology, Babol, Iran. E-mail: {royarahimidisfani, m.valinataj}@nit.ac.ir
† TU Wien, Vienna, Austria. E-mail: nima.taherinejad@tuwien.ac.at

***Abstract*—Memristive technology is a promising emerging technology, which can be used as storage and processing element. Memristors are non-volatile, compact, fast, and energy efficient. They can be used in logic designs to perform basic logical operations in memory and thus avoid the von-Neumann bottleneck. Among various possibilities, stateful logics stand out, since they can process the data with minimum data movement. Material Implication (IMPLY), Memristor-Aided Logic (MAGIC), Three Memristors Stateful Logic (TMSL) and Single-cycle In-memristor XOR (SIXOR) are the some of the main examples of memristor-based stateful logics. Given the maturing state of the memristive technology, in this paper, we evaluate the effect of non-idealities of this technology, especially the device variations, on the operations of stateful logic. In particular, we focus on two well-received logics, namely IMPLY and TMSL, and analyze the effect of various operational conditions and device variations on the functionality of these gates.**

***Index Terms*—Memristor, Logic design, Resistive RAM, Stateful Logics, IMPLY, TMSL.**

I. Introduction

Theory and the basis of the memristors was firstly presented by Leon Chua in 1971 [1] but it was only in 2008 that Hewlett Packard (HP) Laboratories connected the dots between their Titanium-dioxide with the theoretical memristors [2]. Since then, a substantial amount of research and development has been dedicated to memristive circuits and systems for storing data [3]–[7], logical operations [8]–[13] and calculations [14]–[17] and implementation of many emerging paradigms, such as stochastic computing [18]–[20]. In-Memory Computation (IMC) is one of the most important applications of memristors since it can omit the need of data transition to the processing unit. This reduces the delay and energy for the data movement and alleviates the Von-Neumann bottleneck problem [21]. Memristors can be used as the basic structure of logic gates such as Material Implication (IMPLY) [8], [9], Memristor-Aided Logic (MAGIC) [10], Fast and Energy-efficient Logic (FELIX) [12], Three Memristors Stateful Logic (TMSL) [11] and Single-cycle In-memristor XOR (SIXOR) [13]. In aforementioned logics, both input and output are represented as the state (resistance value) of the input and output memristors and therefore are categorized as **‘stateful’** logics [8], [9]. Stateful logics allow for computation inside the memory array, without any need for the data ever leaving the memory array (in-array computing). This is the minimum possible data movement, which maximizes the potential for improvements impeded by large data movements.

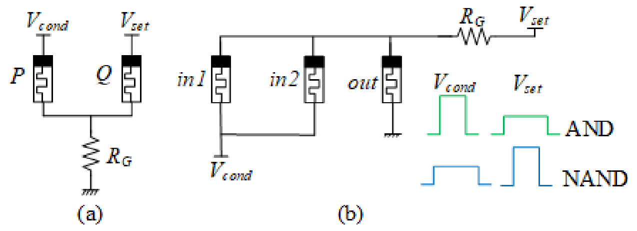

Fig. 1: Memristive logic gates. (a) IMPLY logic and (b) TMSL logic.

Design methodology of the IMPLY and TMSL gates are presented in [8], [9] and [11], respectively. There is a considerable degree of freedom in design choices of the applied voltages, pulse length and other parameter settings. Some of these choices depend on the specifications of the underlying technologies. However, current public models are often based on projections or reflect a limited properties of the actual technology. For instance, until very recently no model had incorporated device variations, even though it is a well-known effect with significant impact on the functionality of memristive circuits and systems [22]. A new memristor model presented in [23], however, does incorporate measurement-based device variations and its respective distribution for a commercially available technology. Hence, it allows us to conduct new studies on the interplay of these variations and various setting parameters of memristive logics. Our study provides an important and necessary insight into the compatibility of these logics and the target memristive technology, as well as suitable setting parameters (operational conditions) to improve the reliability of operations and reduce fault probabilities.

In the rest of this paper, we first present the fundamentals of the logics under study in Section II. In Section III, we present the simulations setup and the obtained results. We discuss the results in Section IV and conclude the paper.

II. Fundamentals

The resistance of a memristor (or its memristance) can take any value between R_{on} and R_{off}, which represent the minimum and maximum resistance of a memristor. These are also called Low Resistance State (LRS) or L and High Resistance State (HRS) or H, respectively. How these values are considered (assigned to) logical values (‘1’ and ‘0’, or True and False) is a choice that -as we see below- are different in different logics. To avoid potential confusions, in the rest of this paper, by default we refer to the resistance value of

TABLE I: The truth table of IMPLY and TMSL AND and NAND

Case No.	*in1*	*in2*	IMP	AND	NAND
0	H	H	L	H	L
1	H	L	L	L	H
2	L	H	H	L	H
3	L	L	L	L	H

memristors (e.g., H and L) and not the assigned logical values, unless necessary.

1) IMPLY: IMPLY is the first presented stateful logic which can be implemented stand-alone or inside a crossbar to build different circuits such as adders [24], [25] and multipliers [26], [27]. As shown in Figure 1(a), IMPLY has a simple structure consisting of two memristors connected to a resistor, R_G. The memristors P and Q are the inputs where initial memristance represents their logical value as inputs. After applying two voltage pulses called V_{cond} and V_{set} to P and Q, respectively, the final memristance of Q will represent the output state.

In logic function of $P \rightarrow Q$, the output state is always L except for the input logic combination of LH (Case 2), given in Table I. In IMPLY, conventionally L is considered as logical value of '1' (True) and H as '0' (False). Therefore, implying H (False) from L (True) is False (H). As elaborated in [27] and [28], there are some basic conditions for choosing the value of R_G, V_{set} and V_{cond} to ensure correct operations. R_G is considered as $R_{on}{<}R_G{<}R_{off}$. The values of V_{set} and V_{cond} should follow the constraints $V_{cond}{<}V_{th}{<}\mathrm{V}_{set}$, and $(V_{set} - V_{cond}) < V_{th}$ (V_{th} is threshold voltage of the memristor).

2) TMSL: The circuit diagram of TMSL logic is shown in Figure 1(b), which consists of three memristors and a resistor (R_G). TMSL can be used to implement AND and NAND logical operations [11]. The amplitude and pulse length (T) of the applied voltage sources (V_{set} and V_{cond}) can directly change the output state. However, the value of R_G can effect on the output state, and it should satisfy $\mathrm{R}_{on}{<}R_G{<}\frac{R_{off}}{2}$. So the selection of proper values is crucial for correct operation.

To perform NAND operation, as given in Table I, a wide low voltage pulse (V_{cond}), is applied to the input memristors and a narrow high voltage pulse (V_{set}), is applied to the output memristor. The output memristor is initialized to H (in this logic assigned to '1' or True logic). The output memristor should remain in H in all cases, except Case 0. Hence, the value of V_{set} should be high enough ($V_{cond} < V_{set}$) to switch the *out* memristor in this case. On the other hand, $V_{cond}{<}V_{th}$ is necessary to avoid changing the output state when one of the memristors is in L.

Implementation of the TMSL-based AND logic gate is similar to NAND. However, the difference is in the value and duration of voltage sources. Here, a narrow and high voltage pulse is applied to the input switches (V_{cond}), and a wide and low voltage pulse is connected to the output memristor (V_{set}) to perform AND operation as given in Table I.

III. Simulations and Evaluation

The explained logic gates are investigated using Spice simulation with LTSpice. To simulate the logics, a new and recently presented memristor model called BELIEVER [23] is used. The memristor model used here is a Self Directed Channel (SDC) ReRAM [29]. BELIEVER incorporates parameters such as device variations and leakage or drift [23], which enable more realistic evaluation of circuits and systems. In each simulation (leading to a point in each diagram) one parameter is changing over the given range and the other parameters are the same as the values in the respective reference. For each combination of parameters the circuit is run 200 times[1] to obtain a statistically relevant data regarding the effect of device variations and its impact on the correct operation of the respective logic with the given configuration (a specific combination of parameters). MATLAB is used to process the results. The output memristance (memristor state) is mapped to logical values and the error rate is calculated based on the Logic Threshold (LTh), which was selected to be in the middle of L and H.

The reported ER is the rate of incorrect outputs in percent in all simulated runs for each point. In Figures 2, 4 and 6, the mean value of all 200 runs in each point is shown as well as the minimum and maximum values of output states in that point.

[1]A larger number of runs did not show any significant change in the results.

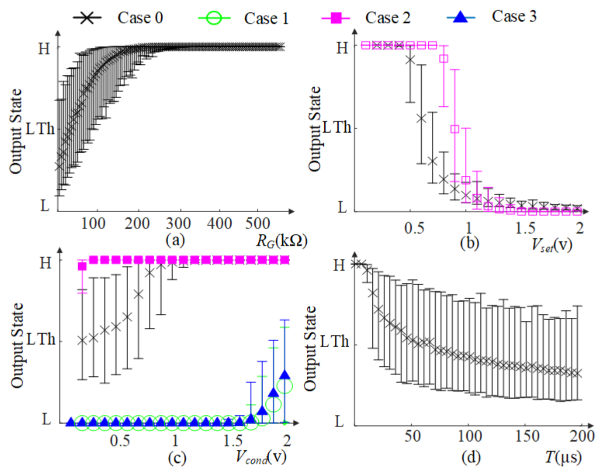

Fig. 2: Variation of output state for different values of (a) R_G, (b) V_{set}, (c) V_{cond} and (d) T for IMPLY logic.

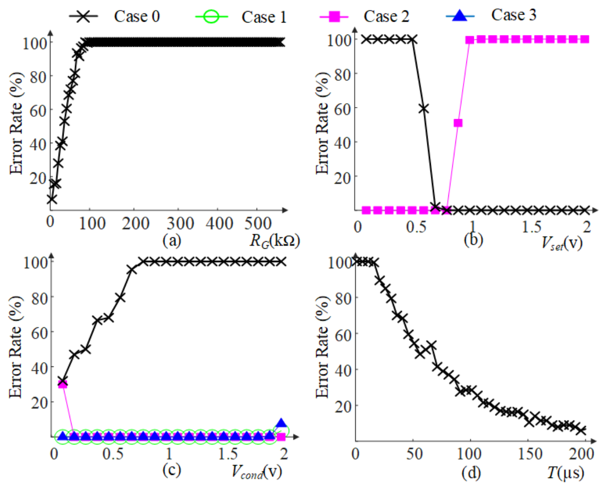

Fig. 3: Variation of error rate for different values of (a) R_G, (b) V_{set}, (c) V_{cond} and (d) T for IMPLY logic.

To make the diagrams more clear and easier to understand, the cases whose output states are correct with zero error rate are not shown in Figures 2 to 7.

1) IMPLY: According to [23], the default configuration for IMPLY in the technology modeled by BELIEVER includes V_{set}=0.6V, V_{cond}=0.4V, R_G=40kΩ and T=50μs. As mentioned before, the value of R_G should be selected between R_{on} (4.92kΩ) and R_{off} (545.54kΩ). The behavior of the *out* memristor for different values of the resistor in the range of 4kΩ to 550kΩ is shown in Figure 2(a) and only for Case 0 since the output memristance in other cases is independent from the value of R_G and there is no erroneous output state. In addition, in Case 0 a larger R_G (>30kΩ) can change the *out* memristance state to the values larger than the LTh. Figure 3(a) illustartes the output error rate for Case 0. As we see in this figure, $R_G > 60k\Omega$ leads to a completely incorrect output.

The variations of output state and error rate based on different values of V_{set} are presented in Figure 2(b) and Figure 3(b), respectively. There is an optimum point with zero error rate, which as shown in Figure 3(b) happens at V_{set}=0.8V. Cases 1 and 3 are independent from variation of V_{set} because Q is in L state and thus with every value of V_{set} it can remain its state. As expected, in Case 0 if the value of V_{set} becomes equal or smaller than V_{cond}, memristance of the *out* cannot change to L, and the output state will be larger than LTh. Hence, the error rate in Fig. 3b will be high. However, for V_{set}>0.5V there is no problem as the output state can reach the desired value. In Case 2, high voltages can change Q to L. Therefore, V_{set} should be lower than 1v to retain the output voltage larger than the threshold.

As shown in Figure 2(c) and Figure 2(d), Case 0 is the only one that is sensitive to both V_{cond} and pulse length (T). If the applied voltage to the p memristor becomes more than V_{set}, p changes to L and the output memristor stays in H. Thus, V_{cond} should be smaller than V_{set} to minimize the error rate. As shown in Figure 2(c), in all cases except Case 0 the output state is correct. However, larger V_{cond} can lead to more incorrect outputs. On the other hand, by increasing the pulse length, the output memristor has more time to change its state to L and reach the stable value in Case 0, which leads to a decrease in error rate, as seen in Figure 3(d) too. In other cases the output memristor does not change its state regardless of the values of V_{cond} and T.

2) TMSL: According to [11], [23], considered default values in this paper for simulation of the AND logic gate with TMSL are V_{set}=0.5V, V_{cond}=1V, R_G=40kΩ and T=100µs. Here, the maximum pulse width is 100 μs, which is the pulse width of V_{set}, as V_{cond} has a shorter pulse width. The variation of output state and the error rate of simulated AND gate are shown in Figures 4 and 5, respectively. There is no error in Case 0 for different values of R_G, V_{set}, V_{cond} and T. Plotted output states for Cases 1 and 2 in Figure 4(a) show incorrect outputs for different values of resistor. Moreover, the error rate of Case 3 is high. So, the assumed default values cause wrong outputs for any values of resistor. Similarly, based on Figure 5(b), the default values cause wrong outputs with 100% error rate regardless of the value of V_{set} in Cases 1 and 2. In Case 3, for $V_{set} \leq$ 0.4V the *out* memristor changes to L but larger V_{set} can maintain *out* mermistor in H. So error rate will increase as shown in Figure 5(b). In Figure 4(c) there are some points with zero percentage error rate ($V_{cond} <$ 0.4V) in all cases of V_{cond} sweep, which are the best choices to set the V_{cond} voltage source. As shown in Figure 4(c), by increasing the voltage source, the output state is also increasing more than LTh in all cases (except for Case 0) and produces wrong outputs with high error rate.

The results of pulse length sweep are illustrated in Figure 4(d) and Figure 5(d). With the default values, the output state will be wrong for each value of T in Cases 1 and 2. But in Case 3 smaller pulse length causes more correct answers with lower error rate, as illustrated in Figure 5(d). Here, like the other AND simulations, the output memristance in Case 0 is not influenced by the variations of T.

The TMSL-based NAND gate is simulated with the default values of V_{set}=1V, V_{cond}=0.5V, R_G=40kΩ and T=100µs. Fig-

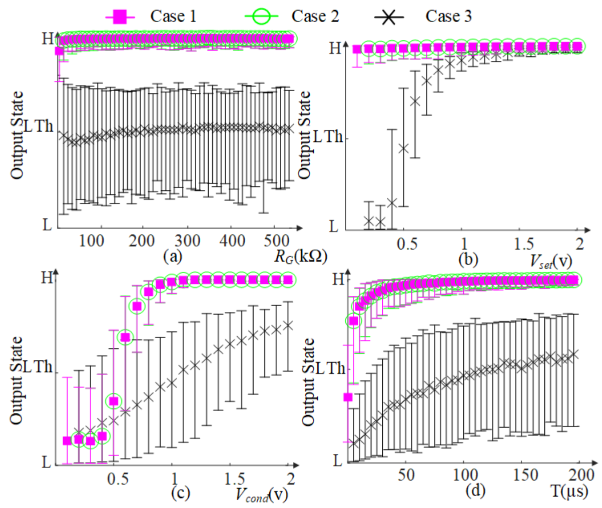

Fig. 4: Variation of output state for different values of (a) R_G, (b) V_{set}, (c) V_{cond} and (d) T for TMSL-based AND logic.

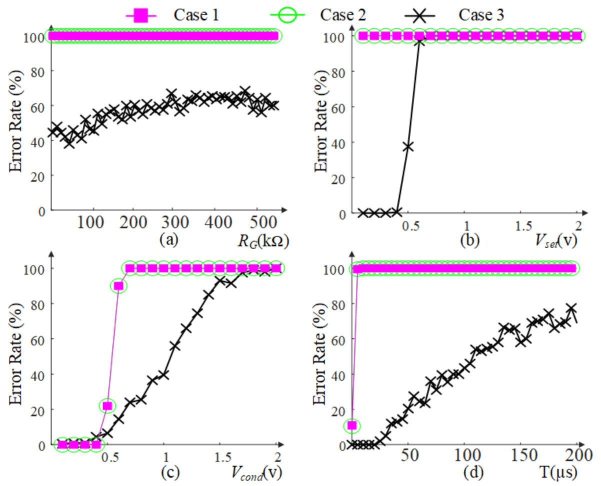

Fig. 5: Variation of error rate for different values of (a) R_G, (b) V_{set}, (c) V_{cond} and (d) T for TMSL-based AND logic.

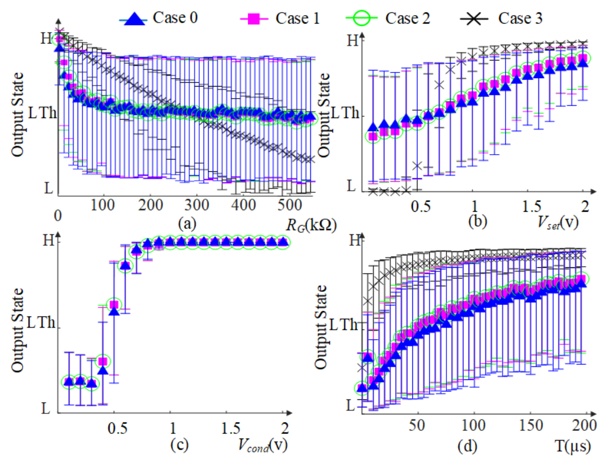

Fig. 6: Variation of output state for different values of (a) R_G, (b) V_{set}, (c) V_{cond} and (d) T for TMSL-based NAND logic.

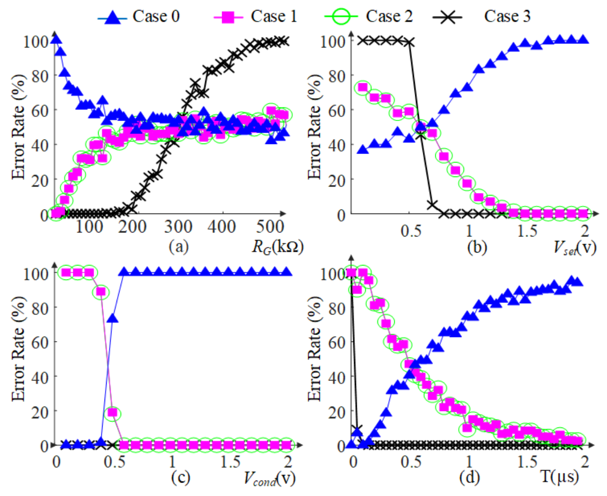

Fig. 7: Variation of error rate for different values of (a) R_G, (b) V_{set}, (c) V_{cond} and (d) T for TMSL-based NAND logic.

ure 6 shows the impact of parameters sweep on the output state. As shown in Figure 6(a), there is a wide range of changes in the output state for Case 0. A larger resistor decreases the mean value of output state for this case, and thus the error rate decreases as well, as illustrated in Figure 7(a) . The value of 40kΩ for R_G is a good choice for minimum error rate in Cases 1, 2 and 3 based on Figure 7(a). However, there is a high error rate for small resistors in Case 0 because small resistors allow the *out* memristor to change its state to H. By setting R_G<200kΩ the output memristor can change its state from H to L but larger resistors increase the output error rate.

There is a range of points in Figure 6(b) and Figure 7(b) for tuning the value of V_{set} to have a smaller error rate. Based on the observations, 0.9V<V_{set}<1.1V has the minimum error rate in all cases. By increasing V_{set} the output state of *out* will be more likely correct in Cases 1, 2 and 3 but in Case 0 the output memristor cannot change to L and error rate increases. Sweeping V_{cond} (shown in Figure 6(c)) the behavior of output state is similar in Cases 0, 1 and 2. In Case 3, generated outputs are completely correct. Given that the output state in Case 0 is L, $V_{cond} > 0.5$V increases the error rate. There is just one point in which there is no case with 100% error rate. Therefore, V_{cond}=0.5V is the best choice since in all other voltages there is at least a case with 100% error rate. In Figure 6(d), the circuit is simulated with various pulse width in the range of 0.1μs<T<200μs. The mean value of *out* state increases in all cases when T increases in the range. Hence, the error rate in Cases 1, 2 and 3 decreases whereas it increases in Case 0 (as shown in Figure 7(d)). However, there is a crosspoint with minimum error rate that happens at $T = 55\mu s$ which has the error rate of 46.5% in Case 0 and a bit lower otherwise.

TABLE II: Reliable situations for parameter setting

Logic \ Parameter		Error Rate (%)	R_G ($k\Omega$)	V_{set} (V)	V$_{cond}$ (V)	T (μs)
IMPLY		0	40	0.8	0.4	50
		<5	40	0.7	0.4	50
		<10	4	0.6	0.4	50
			40	0.6	0.4	200
TMSL	AND	0	40	0.5	0.4	100
		<5	40	0.5	<0.4	100
		<10	-	-	-	-
	NAND	<10	-	-	-	-

IV. Discussion and Conclusions

Based on the results reported in the previous section, we can determine the suitable operation conditions of the investigated logics, IMPLY and TMSL, with respect to different design parameters. Table II represents the reliable situations in which the output memristance will be "correct" (i.e., has a limited error rate) in all input combinations and despite device variations. There is a point with zero error rate in IMPLY and AND logics. Situations with acceptable error rate (<10%) are also mentioned in this table. This means that these parameter values should produce a reliable operation (correct in more than 90% of cases) for these two gates in the technology modeled by the BELIEVER model [23]. Based on our experiments, NAND gate has at least 46.5% error rate. This means that we could not find any set of parameter values that lead to a reliable operation for this gate in the technology modeled by the BELIEVER model [23]. We plan to extend our search space in the future, to examine whether a reliable configuration in our technology can be found. Nonetheless, we note that in a different memristive technology, this gate may have a configuration that works reliably. This includes a potential future version of this same technology that may have a smaller device variation.

On the other hand, we need to note that we considered a LTh that divided the overall range to two halves for the two logic values (levels). In a more constringent logic, where there is a buffer zone between the logics (e.g., 30% buffer zone and each logic level constituting 35% of the overall range), the error rate for all above logic would increase and limit the configuration set that can lead to reliable operations. We plan to extend our study by expanding the range of design space explored for these logics and by conducting a similar study for other in-array logics such as FELIX, SIXOR and MAGIC.

REFERENCES

[1] L. O. Chua. Memristor—the missing circuit element. *IEEE Transactions on Circuit Theory*, CT-18(5):507–519, September 1971.

[2] D. B. Strukov *et al.* The missing memristor found. *Nature*, 453:80–83, May 2008.

[3] H. Kim et al. Memristor-based multilevel memory. In *CNNA2010*, pp. 1–6, Feb 2010.

[4] M. Zangeneh and A. Joshi. Design and optimization of nonvolatile multibit 1T1R resistive RAM. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(8):1815–1828, Aug 2014.

[5] N. Taherinejad *et al.* Memristors' potential for multi-bit storage and pattern learning. In *2015 IEEE European Modelling Symposium (EMS)*, pp. 450–455, Oct 2015.

[6] N. Taherinejad *et al.* Fully digital write-in scheme for multi-bit memristive storage. In *2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–6, Sept 2016.

[7] D. Radakovits and N. TaheriNejad. Implementation and characterization of a memristive memory system. In *2019 IEEE 32nd Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–5, May 2019.

[8] E. Lehtonen and M. Laiho. Stateful implication logic with memristors. In *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, 2009.

[9] J. Borghetti *et al.* 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature*, 464:873–876, April 2010.

[10] S. Kvatinsky et al. MAGIC; memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):895–899, Nov 2014.

[11] P. Huang *et al.* Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits. *Advanced Materials*, 28(44):9758–9764, 2016.

[12] S. Gupta *et al.* FELIX: Fast and energy-efficient logic in memory. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, November 2018.

[13] N. TaheriNejad. SIXOR: single-cycle in-memristor XOR. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 29(5):925–935, 2021.

[14] S. Li et al. Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. In *DAC2016*, pp. 1–6. IEEE, 2016.

[15] P.-E. Gaillardon et al. The programmable logic-in-memory (PLiM) computer. In *DATE2016*, pp. 427–432. Ieee, 2016.

[16] G. Papandroulidakis *et al.* Crossbar-based memristive logic-in-memory architecture. *IEEE Transactions on Nanotechnology*, 16(3):491–501, May 2017.

[17] S. G. Rohani and N. TaheriNejad. An improved algorithm for IMPLY logic based memristive full-adder. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, April 2017.

[18] M. R. Alam *et al.* Exact in-memory multiplication based on deterministic stochastic computing. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2020.

[19] M. R. Alam *et al.* Exact stochastic computing multiplication in memristive memory. *IEEE Design Test*, pp. 1–8, 2021.

[20] M. R. Alam *et al.* Sorting in memristive memory. *ACM Journal on Emerging Technologies in Computing Systems*, pp. 1–22, 2022.

[21] M. A. Zidan *et al.* The future of electronics based on memristive systems. *Nature electronics*, 1:22–29, 2018.

[22] N. TaheriNejad and D. Radakovits. From behavioral design of memristive circuits and systems to physical implementations. *IEEE Circuit and Systems (CAS) Magazine*, 19(4):6–18, Fourthquarter 2019.

[23] D. Radakovits and N. Taherinejad. Behavioral leakage and inter-cycle variability emulator model for rerams (BELIEVER). *arXiv*, 2103.04179, 2021.

[24] S. Ganjeheizadeh Rohani *et al.* A semiparallel full-adder in imply logic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(1):297–301, Jan 2020.

[25] N. TaheriNejad *et al.* A semi-serial topology for compact and fast IMPLY-based memristive full adders. In *2019 IEEE New Circuits and Systems symposium (NewCAS)*, pp. 1–5, 2019.

[26] D. Radakovits *et al.* A memristive multiplier using semi-serial imply-based adder. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(5):1495–1506, 2020.

[27] A. Karimi and A. Rezai. Novel design for a memristor-based full adder using a new imply logic approach. *Journal of Computational Electronics*, 17(3):1303–1314, Sep 2018.

[28] S. Kvatinsky *et al.* Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(10):2054–2066, October 2014.

[29] E. Gale. Tio$_2$-based memristors and reram:materials, mechanisms and models (a review). *Semiconductor Science and Technology*, 29(10):104004, 2014.