Defect-Tolerant Logic Synthesis for Memristor Crossbars with Performance Evaluation

Onur Tunali, M. Ceylan Morgül, and Mustafa Altun
Istanbul Technical University

In this paper, we study defect-tolerant logic synthesis of memristor-based crossbar architectures. We propose a hybrid algorithm, combining heuristic and exact algorithms, that achieves perfect tolerance for 10-percent stuck-at open defect rates. Along with defect tolerance, we also consider area, delay, and power costs of the memristor crossbars to elaborate on two-level and multi-level logic designs.

Due to the scaling issues of current CMOS-based technologies, research has been shifted to novel computing elements such as memristors. Even though the theoretical manifestation of the memristor goes back to the 1970s, physical realization of an actual circuit component is very recent and was established by HP. After this initial step, a variety of memristor-based logic circuit designs were proposed, such as Boolean logic, implication logic, and threshold logic. A comprehensive review and references of memristor-based logic circuits can be found in Vourkas and Sirakoulis.[1] In this paper, we mainly focus on defect-tolerance aspects of crossbar arrays using memristors as switching elements. We also perform comprehensive performance evaluation of the arrays by considering delay power and delay costs in relation to the used logic designs.

The first study using hysteric resistors as memristive components in crossbar architectures is demonstrated in Snider.[2] An integrated design approach is devised in Xie *et al.*,[3] showing the implementation of arbitrary Boolean logic functions on crossbar arrays. This work adopts a two-level logic design using NAND - AND planes, obtaining the negation of a logic function with a final inversion. However, authors overlooked the fact that the crossbar is able to produce the logic function and its negation as outputs, so considering both cases with different two-level logic design techniques would generate a potential optimization in terms of area and power costs, as shown in this paper. Furthermore, by modifying the computing states, we demonstrated in a previous paper[4] that it is possible to achieve multi-level logic design that uses the outputs of NAND gates computed in the previous level as inputs. An improvement of multi-level design is also presented.

Considering defect issues of memristor-based crossbars realizing logic functions, defects occurring in a crossbar degrade operational capacity of switches and complicate the logic-synthesis process severely. There are various studies such as Xie *et al.*[5] that examine the robustness of the crossbar, which disregards any operational or switch defects. Velasquez and Jha[6] perform defect-tolerant logic mapping in which the entire process is crossbar-dependent. To tackle this challenge, first we model defects borrowing the common terminology as stuck-at open and closed types based on their physical resistance characteristics.

Next, we devise a modular defect-tolerant logic-synthesis process consisting of generating a crossbar-independent description of the logic function and a fast hybrid mapping algorithm utilizing the combination of a heuristic matching and an assignment method. Indeed, this problem is very similar to the logic mapping of reconfigurable nano-crossbar arrays (using AND-OR logic) for which a quite mature literature exists.[7,8] However, most of the mentioned studies focus on a single plane of crossbar, particularly AND, and neglect the OR plane. Furthermore, the related algorithms operate using 1.5 times larger crossbars and show poor performance for optimum-size crossbars. Motivated by these shortcomings, we propose a hybrid algorithm, a combination of heuristic and exact algorithms, for optimum-size crossbars. Our algorithm covers the whole crossbar array by applying a heuristic approach for NAND and AND planes with an exact assignment technique for the output connections of given logic functions, which is more critical since a single defect might discard a whole output. Furthermore, we show that defect-tolerance performance and area, delay, and power costs of the crossbars are strongly linked with the used logic design technique.

BACKGROUND

Memristor Model

A memristor is a nonlinear electrical component that shows resistive switching properties. When it is SET or RESET, a memristor keeps its state unless the voltage difference between the terminals of the component is inside the defined constraints. Figure 1 shows I-V characteristics and switching behavior of an ideal memristor. In this paper, we use the Snider Boolean Logic model, which regards a lower-resistance R_{ON} as logic "0" and a high-resistance R_{OFF} as logic "1."[2]

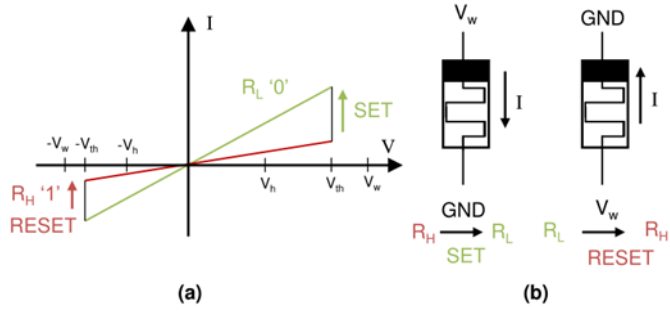

Figure 1. (a) I-V characteristics and (b) switching operations of a memristor.

Logic Synthesis of Memristor Crossbars

A crossbar array is constructed from two layers of orthogonal wires/lines. Every crosspoint/junction acts as a switching element, which is a memristor in this study. The memristor-based crossbar proposed by Xie *et al.*[3] is able to perform two-level synthesis for an arbitrary logic function in sum-of-products (SOP) form as $f = m_1 + ... + m_k + ... + m_n = \overline{\overline{m_1} \cdots \overline{m_k} \cdots \overline{m_n}}$, where m's are minterms of f. Note that minterm and product concepts are used interchangeably in the literature

(although they are fundamentally different), so we follow the same tendency in this paper, as well.

Using the same approach, one can also perform multi-level logic synthesis.[4] In short, the logic-synthesis process of a crossbar consists of choosing which switches to activate and disable to implement a given Boolean function. A memristor switch can be programmed into two operational ranges:[2,3]

- *Active*. Memristors can switch between two resistive states (low R_{ON} and high R_{OFF})
- *Disabled*. Memristors always stay in the R_{OFF} state regardless of voltage difference.

As an example, two-level and multi-level implementations of a Boolean function $f = \overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5}\overline{x_6}\overline{x_7}\overline{x_8}$ is shown in Figures 2(a) and 2(b), respectively. Horizontal and vertical lines represent the minterms and inputs, respectively. If an input is present in the minterm, the corresponding switch is activated; otherwise, it is disabled. Here, computation cycles are controlled by CMOS controllers, which either sets or resets the related memristor, sequentially. Since the multi-level design has more cycles, controllers tend to be more complex compared to the two-level design.

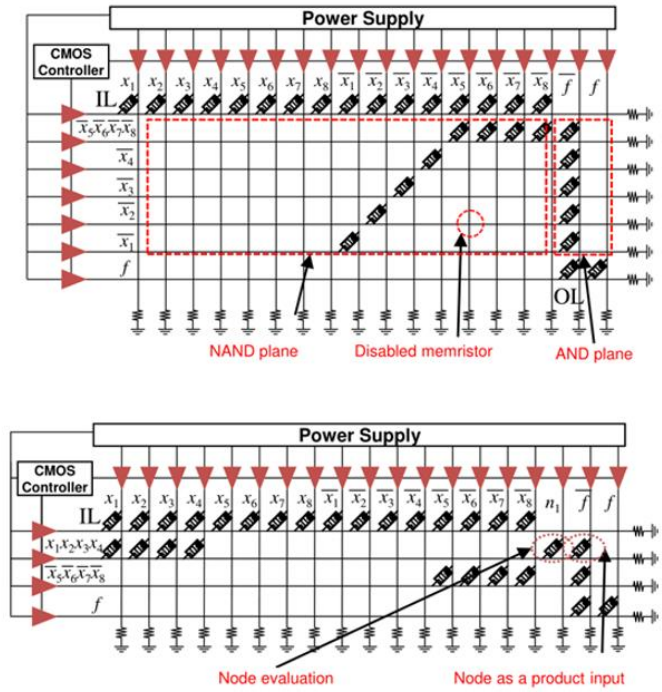

Figure 2. Logic mapping of a Boolean function on a crossbar with (a) two-level design and (b) multi-level design.

Readers are encouraged to refer to Xie *et al*.[3] and Tunali and Altun[4] for further information on logic synthesis of memristor crossbars.

Area-Power-Delay Analysis of Memristor Crossbars

To evaluate operational capacity of memristor crossbars, we introduce four parameters used in both two-level and multi-level designs and/or defect-tolerance performance of crossbars:

- *Area cost*. The size of the crossbar used to implement a given logic function. Area costs of two- and multi-level designs are calculated as AreaTwo = (# of minterms + # of outputs) × (2 × (# of inputs + # of outputs)) or AreaMulti = (# of total minterms nodes + 1 + # of nodes) × (2 × (# of inputs + # of outputs) + # of inverted nodes).
- *Delay cost*. Elapsed time to compute sequential states. It is calculated as (# of levels + 5), as in Xie *et al*.[3] Note that it is always 7 for two-level design.
- *Power factor*. Total memristor count divided by delay cost. We aim to approximate worst-case power. We assume that the memristor has zero static power.[2,3] Dynamic power depends on the power consumption of changing a memristor state (low to high and high to low). We assume that occurrence probabilities of input states are equal and, in the worst case, all memristor states are changing. Therefore, energy consumption becomes proportional to the number of memristors in the array,[3] and power consumption can be found as energy/delay. We encourage readers to look into Traiola, Barbareschi, and Basio[9] for more detailed information on estimating average power consumption of memristor arrays.
- *Logic inclusion ratio (IR)*. The ratio of the number of switches (memristors) used to realize a logic function to area cost.

Using the two-level design example in Figure 2(a), a crossbar has 7 horizontal lines and 18 vertical lines, so area cost is 126. There are 31 memristors used in implementation, so IR = 31/126 = 25 percent, and delay is 7. Power factor is 31/7 = 5.28. And for the multi-level design example in Figure 2(b), the crossbar has 4 horizontal lines and 19 vertical lines, so the area cost is 76. There are 29 memristors used in implementation, so IR = 29/72 = 38 percent, and delay cost is 8 (3 (# of levels) + 5 = 8). The power factor is 29/8 = 3.62.

DEFECT-TOLERANT MAPPING

First, we explain the used two-level and multi-level design techniques for a technology-independent description of a logic function for defect tolerance and performance evaluation. Then, we elaborate on defect models and, finally, propose the defect-tolerant algorithm.

Used Two-Level and Multi-Level Logic Designs

Memristor-based crossbars use minterms (more specifically, products) of the function in SOP form to obtain the function's negation, and then it obtains the function itself with a last inverter. We call forms of the original function and its negation Phase 1 and Phase 0, respectively. We calculated the function (Phase 1) and its negation (Phase 0), and then chose which one to select by using the PLA-based logic minimization tool Espresso. However, for multi-output functions (most of the benchmark functions), we don't have only two phases; rather, we have phase combinations. For n outputs, a function will have $2n$ numbers of phase combinations. Since minterms (or products) can be shared with outputs[3] as in PLA-like synthesis, we aim to increase the number of shared products among outputs with changing phases of outputs. Since a brute-force approach has a high time cost, we have written a greedy algorithm called the Greedy Two-Level Algorithm (GTLA). We first calculate product numbers of each output and its negation. The phase of an output is determined as the phase having the fewest products. We obtain the final phase combination according to the output phases.

In our previous work, for multi-level designs, we used the ABC logic-synthesis tool to acquire a gate-level technology mapping.[4] But the memristor delay-cost function is different than conventional technologies.[3] Therefore, in this work, we used the SIS logic-synthesis tool for technology-independent multi-level logic synthesis. We used the area-optimization script "script.rugged." Later in this paper, we evaluate the two-level and multi-level design techniques by investigating defect tolerance, as well as area, delay, and power costs of memristor crossbars.

Defect Model

A defective switch cannot operate properly (meaning no switching between stuck-at low or high resistance states, as reported in Li *et al.*[10]), and the cause is believed to be due to insufficient dopants or impurities.[11,12] Therefore, memristor defects might be modeled as stuck-at open and stuck-at closed types. Physical resistance equivalence of defect types can be defined as follows:

- *Stuck-at open*. The memristor is always in R_{OFF} mode, which means high resistance.
- *Stuck-at closed*. The memristor is always in R_{ON} mode, which means low resistance.

Stuck-at open defects show the same characteristics as programming a memristor as disabled in the mapping phase, so avoiding these defects during logic mapping is adequate for a valid mapping. However, stuck-at closed defects are always R_{ON}, so they disrupt the operational capacity of both horizontal and vertical lines. Since every computation step starts with initializing all the memristors to R_{OFF} and then copying the input values to minterms, a vertical line belonging to the defective switch cannot be used. Furthermore, R_{ON} is equivalent to logic "0" in the Snider Boolean logic model,[2] so every horizontal line that computes a NAND gate results in logic "1," independent of the other inputs. For this reason, tolerance of stuck-at closed defects is basically replacing the defective lines with defect-free ones, so it is not possible without any redundant crossbar lines. Finally, prior to the mapping process, we assume we will know the locations of defects, which are denoted with a defect map. After the fabrication of the memristor crossbar, it is possible to construct a defect map with testing techniques, as discussed in Tunali and Altun.[8]

Proposed Defect-Tolerant Logic-Mapping Method

To distinguish defect-tolerant mapping, we start with a naive mapping approach (see Figure 3(a)) disregarding defects for a given logic function, and an invalid mapping is obtained at the end. However, after careful consideration, a valid mapping is produced (see Figure 3(b)) with activating correct switches. In our methodology, we use the following concepts, which are thoroughly explained in Tunali and Altun:[4]

- Function matrix (FM) is a representation of a logic function in matrix form.
- Crossbar matrix (CM) is a representation of a defect map that shows the crossbar positions of either defective or functional switches.
- Row matching checks a row of FM or CM element by element.
- Matching matrix shows valid row matchings of FM and CM.

In short, our algorithm is composed of three parts:

1. First, the description of logic function (which is generated by two-level or multi-level logic design techniques) and the defect map of the crossbar (which is obtained through testing in advance) are converted into matrix form as FM and CM, respectively.
2. Second, a heuristic matching is applied to all minterm (product) rows of FM (denoted with FM_m), which performs row-by-row matching between FM_m and CM from top to bottom. During the process, matched rows of the CM are traced with an array showing which rows of the FM_m are assigned to them. At first, the matching searches only unmatched rows. If an FM_m row cannot be matched with the unmatched rows of the CM, then backtracking starts by considering the matched rows of the CM from top to bottom. If a matching is found, the previously assigned row of the FM_m is checked for whether it can be assigned to an unmatched row of the CM. If this check results in a mismatch, the algorithm continues with the next matched row of the CM and repeats the same process to find a valid matching.
3. As a final step, a matching matrix for output rows of FM and unmatched rows of CM (denoted with CM_u) is constructed. By using an assignment algorithm that chooses which output is mapped to unmatched rows of CM yielding a zero cost, we ensure that every output has a valid row matching. We use the classic Munkres' assignment algorithm for finding an assignment producing zero cost. This is an exact algorithm, which means if a zero cost is possible, it will be found.

We are using a hybrid algorithm due to runtime issues. Constructing a matching matrix and applying an assignment method to all rows of FM would increase computational load of the algorithm excessively for larger logic functions. We show drastic runtime differences in the next section.

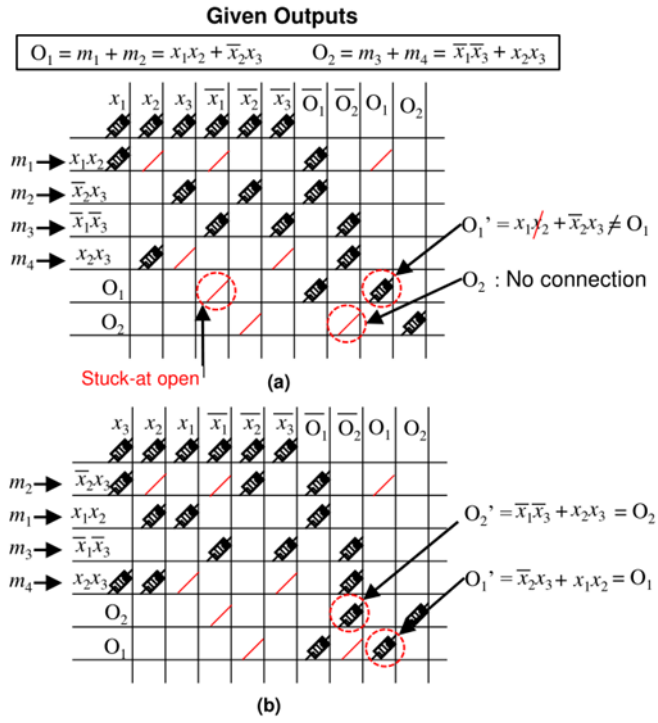

Figure 3. Logic mapping of a given Boolean function. Red diagonal lines represent a defect on the crosspoint. First, (a) mapping is applied by disregarding defects without a valid mapping and, second, (b) mapping is applied by considering defects with a valid mapping.

OVERALL PERFORMANCE EVALUATION

For experimental results of our defect-tolerant logic-synthesis process, we use Monte Carlo simulations. We generate random defective crossbars by assigning an independent defect probability rate (10 percent) to each crosspoint showing a uniform distribution and attempt to map given benchmark logic functions using proposed defect-tolerant mapping algorithms. As opposed the common tendency of using 1.5 times larger crossbars for logic mapping,[7,8] we utilize optimum-size crossbars with no redundant lines present. We only include stuck-at open defects since our simulation regards optimum crossbars, so defect tolerance is not possible in terms of stuck-at closed types due to redundancy requirements. Furthermore, as reported in Li *et al.*[10] and Borghetti *et al.*,[13] defective memristors are dominantly observed to be stuck at high-resistance states, meaning that stuck-at open defect rates reach up to 10 percent. Considering the mentioned findings, our simulation setting depicts a fairly accurate representation of physical results. Finally, all algorithms are implemented in MATLAB, and experiments run on a 3.30-GHz Intel Core i7 CPU with 8 GB memory.

To evaluate operational capacity of memristor crossbars, we use area-cost, power-factor, and delay-cost parameters. In addition, we quantify the complexity of the CMOS control unit as a representation of its area and power. Since the control unit is functionally a demultiplexer, we define its complexity as the product of the number of its inputs and outputs. The number of inputs can be formulized as $\log_2(\#\text{ of states}) = \log_2(5 + \#\text{ of levels})$. The number of outputs can be formulized as $3 \times (\#\text{ of levels})$ by considering that, in each level, there are three memristor sets—

corresponding to inputs, products, and outputs—to be controlled successively. As a result, the complexity of the control unit can be formulized as $\log_2(5 + \#\text{ of levels}) \times 3 \times (\#\text{ of levels})$.

For the proposed hybrid algorithm (HBA), we consider success rate (SR) and runtime values compared to those of the exact algorithm (EA). Contrary to HBA, which applies the assignment method only to output rows, EA constructs the matching matrix for all minterms and output rows of FM, and then applies the assignment method.

Our simulations are performed as follows. Given a benchmark function, first, we use two-level or multi-level logic design to obtain the description of the function. Then, the logic function is converted into a matrix (FM). Randomly generated defective crossbars (200 samples) are in matrix form (CM), as well. Finally, proposed defect-tolerant mapping methods try to find a valid mapping of FM for each CM.

In the first simulation, we compared HBA and EA, employing original and the proposed (GTLA) two-level logic design techniques. As can be seen in Table 1, we have managed to decrease the area cost and power factor of most benchmarks using GTLA. For example, regarding ex5p, we have managed to increase the SR to 100 percent while decreasing the power factor by more than half (from 342.57 to 165.29). However, IR increases for certain benchmarks (rd53, rd73, and rd84), which returns as a performance degradation for both HBA and EA. The reason behind that is that higher IR means denser crossbar, so the mapping operation becomes difficult generally. In terms of runtime, HBA is superior to EA for all cases—at least one order of magnitude and at most two orders of magnitude for circuits such as apex4, alu4, and ex1010. In terms of SR, HBA underperforms EA in a few cases only for the benchmarks rd73 and ex5p.

In the second simulation, we performed a comprehensive performance evaluation by only using HBA for both two-level and multi-level logic approaches. As can be seen in Table 2, we have managed to decrease area cost, power factor, and runtime and increase SR for rd73, sqrt8, rd84, and alu4 using multi-level design, which is an absolute improvement regarding all parameters. In addition, power-factor values of multi-level design are lower for all benchmark functions, occasionally even down to an order of magnitude, such as rd73, rd84, and alu4. The main advantage of multi-level design is its ability to produce sparser (low-IR) logic-function descriptions, and it demonstrates a two to three times runtime performance increase when the area cost is lower than that of the two-level design. Even for the exception of apex4 and ex1010, the runtime difference is insignificant considering that mapping is executed for 16 to 30 times larger crossbars (area cost) for the same function designed with multi-level design than with its two-level counterpart. In terms of SR, if the original benchmark functions are designed with multi-level design, HBA is able to find a valid mapping in 100 percent of cases. It should also be noted that, for benchmarks such as ex5p, clip, and sao2 that require larger crossbars when synthesized with multi-level design, HBA is still able to maintain higher SRs (100 percent) than those of the two-level design. The main drawback of multi-level design is the complexity of its control unit. It is clear from Table 2 that multi-level design demands a fairly more complex control unit than its two-level counterpart. For certain cases such as ex1010, apex4, table3, and misex3c, complexity differences reach to an order of magnitude.

As an overall evaluation, our performance findings can be summarized as follows:

- GTLA is able to decrease area cost and power factor, but occasionally increases IR, generating denser (higher-IR) logic functions; this is the key parameter in defect-tolerant mapping. Regarding defect-tolerance performance, the fewer memristors to match (low IR), the better the algorithm performs.
- Multi-level design produces a sparser (low-IR) logic-function description and naturally better defect-tolerance performance, but the area-cost trend is not predictable and its CMOS control unit complexity is much higher than that of the two-level design.
- HBA overwhelms EA for runtimes with a slight or no decrease in SRs.

CONCLUSION

The inherent power efficiency and integration potential of memristors have set forth an interesting and alternative solution path to the ongoing CMOS shrinkage issue. Although large-scale integration of memristor-based crossbars is rather recent and problematic in terms of defect rates, fundamental computation is shown to be feasible with the right combination of fabrication and electronic design automation techniques, as demonstrated in this paper. Currently, the main bottleneck is lack of data regarding physical and operational characteristics of memristor-based systems. However, the increase in the physical realization of memristor-based circuits will generate more in-field data related to delay, power consumption, defect rates, and so on; researchers will soon be able to fine-tune both their instruments and computational approaches.

Table 1. SR and runtime values of HBA and EA for optimum-area crossbars with a 10-percent defect rate.

Bench Name	Area Cost		Power Factor (worst-case)		IR		HBA				EA			
	Org	GTLA	Org	GTLA	Org	GTLA	Org		GTLA		Org		GTLA	
							SR	T	SR	T	SR	T	SR	T
rd53	544	400	26.29	21.43	33%	35%	98	1	94	1	98	1	94	1
squar5	858	806	22.00	21.71	16%	15%	100	1	100	1	100	1	100	1
inc	1,248	1,248	36.43	37.29	17%	17%	100	1	100	1	100	1	100	1
rd73	2,600	1,920	130.86	132.00	34%	40%	83	2	79	2	92	13	79	6
misex1	570	570	21.00	23.00	19%	19%	100	1	100	1	100	1	100	1
sqrt8	1,008	624	30.29	21.29	21%	20%	99	1	100	1	99	1	100	1
ex5p	19,454	18,744	342.57	165.29	10%	7%	65	6	100	7	80	21	100	10
rd84	6,216	4,560	299.00	295.71	33%	38%	71	6	47	4	79	84	57	40
clip	3,500	3,360	118.57	119.00	23%	23%	100	5	100	5	100	30	100	20
sao2	1,736	1,344	75.00	62.43	29%	28%	97	1	99	1	97	2	99	1
ex1010	11,760	11,760	400.86	402.57	23%	23%	100	3	100	2	100	54	100	50
alu4	25,652	16,500	740.57	468.57	19%	19%	100	10	100	6	100	305	100	138
apex4	25,480	25,368	796.86	790.00	21%	21%	100	8	100	7	100	152	100	150
bw	3,300	3,300	69.14	69.71	12%	12%	100	2	100	2	100	2	100	2
table3	10,584	10,584	403.71	403.71	25%	25%	100	3	100	4	100	24	100	32
misex3c	11,816	11,760	248.86	250.71	13%	13%	100	3	100	3	100	41	100	34

Org: Original two-level logic synthesis
GTLA: Two-level logic synthesis with our GTLA
SR: Success rate in percentage
IR: Logic inclusion ratio (% of activated memristors)
T: Average runtime in milliseconds
Delay cost for two-level design is 7 for all benchmarks.

Table 2. SR and runtime values of HBA with two-level and multi-level designs for optimum-area crossbars with a 10-percent defect rate.

Bench Name	Control Unit Cmp.**	Area Cost		Power Factor (worst-case)		Delay*	IR		HBA			
	Multi	Two	Multi	Two	Multi	Multi	Two	Multi	Two		Multi	
									SR	T	SR	T
rd53	54	544	784	26.29	7.73	11	33%	10%	98	1	100	1
squar5	108	858	1,088	22.00	7.57	14	16%	9%	100	1	100	1
inc	72	1,248	2,160	36.43	14.46	13	17%	7%	100	1	100	1
rd73	120	2,600	2,544	130.86	10.67	15	34%	6%	83	2	100	1
misex1	54	570	1,440	21.00	10.18	11	19%	8%	100	1	100	1
sqrt8	24	1,008	1,064	30.29	15.89	9	21%	11%	99	1	100	1
ex5p	144	18,744†	39,900	165.29	32.88	17	7%	1%	100	7	100	6
rd84	54	6,216	4,588	299.00	21.82	11	33%	5%	71	6	100	2
clip	108	3,500	6,468	118.57	22.57	14	23%	4%	100	5	100	4
sao2	108	1,344†	3,168	62.43	18.07	14	27%	8%	99	1	100	1
ex1010	315	11,760	341,960	400.86	140.23	26	23%	1%	100	3	100	10
alu4	72	25,652	12,800	740.57	33.08	13	19%	3%	100	10	100	3
apex4	315	25,480	407,232	796.86	145.08	26	21%	1%	100	8	100	10
bw	156	3,300	8,580	69.14	16.61	18	12%	3%	100	2	100	2
table3	345	10,584	73,367	403.71	51.61	28	25%	1%	100	3	100	3
misex3c	180	11,816	21,728	248.86	41.05	20	13%	3%	100	3	100	1

SR: Success-rate percentage

IR: Logic inclusion ratio (% of activated memristors)

T: Time in milliseconds

†: Designed with **GTLA**

***: Delay cost** for two-level design is 7 for all benchmarks.

****: Control unit complexity** for two-level design is 18 for all benchmarks.

ACKNOWLEDMENTS

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 691178. This work is supported by the TUBITAK-Career project #113E760.

REFERENCES

1. I. Vourkas and G.C. Sirakoulis, “Emerging memristor-based logic circuit design approaches: A review,” *IEEE Circuits and Systems*, 2016.
2. G. Snider, “Computing with hysteretic resistor crossbars,” *Applied Physics A: Materials Science & Processing*, 2005.
3. L. Xie et al., “Fast Boolean logic mapped on memristor crossbar,” *33rd IEEE International Conference on Computer Design (ICCD)*, 2005.
4. O. Tunali and M. Altun, “Logic Synthesis and Defect Tolerance for Memristive Crossbar Arrays,” *Design, Automation, and Test in Europe Conference (DATE)*, 2018.
5. L. Xie et al., “On the robustness of memristor based logic gates,” *IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2017.
6. A. Velasquez and S.K. Jha, “Fault-tolerant in-memory crossbar computing using quantified constraint solving,” *33rd IEEE International Conference on Computer Design (ICCD)*, 2015.
7. O. Tunali and M. Altun, “Permanent and transient fault tolerance for reconfigurable nano-crossbar arrays,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.
8. O. Tunali and M. Altun, “A Survey of Fault-Tolerance Algorithms for Reconfigurable Nano-Crossbar Arrays,” *ACM Computing Surveys*, 2017.
9. M. Traiola, M. Barbareschi, and A. Basio, “Estimating dynamic power consumption for memristor-based CiM architecture,” *Microelectronics Reliability*, 2018.
10. C. Li et al., “Efficient and self-adaptive in-situ learning in multilayer memristor neural networks,” *Nature Communications*, 2018.
11. O. Kavehei et al., “The fourth element: characteristics, modelling and electromagnetic theory of the memristor,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2010.
12. J.J. Yang, D.B. Strukov, and D.R. Stewart, “Memristive devices for computing,” *Nature Nanotechnology*, 2013.
13. J. Borghetti et al., “A hybrid nanomemristor/transistor logic circuit capable of self-programming,” *Proceedings of the National Academy of Sciences*, 2009.

ABOUT THE AUTHORS

Onur Tunali has a master’s degree in nanoscience and nano-engineering from Istanbul Technical University. Previously, he studied mathematics at Istanbul University. Contact him at onur.tunali@itu.edu.tr.

M. Ceylan Morgül is a PhD student in electronics engineering at Istanbul Technical University. He has a master’s degree in electronics engineering from the same university. Contact him at morgul@itu.edu.tr.

Mustafa Altun is an associate professor of electronics engineering at Istanbul Technical University. He has a PhD in electrical engineering with a PhD minor in mathematics from the University of Minnesota. He is an author of more than 50 peer-reviewed papers and a book chapter, and he is the recipient of the TUBITAK Success, TUBITAK Career, and Werner von Siemens Excellence awards. Contact him at altunmus@itu.edu.tr.