On the Design and Fabrication of PATH-based In-Memory Computing Multipliers

Jinam J. Modasiya
University at Albany, SUNY
Albany, NY, USA
jmodasiya@albany.edu

Venkata Nithin Kamineni
University of Florida
Gainesville, FL, USA
vkamineni@ufl.edu

Muhammad Rashedul Haq Rashed
University of Texas at Arlington
Arlington, TX, USA
muhammad.rashed@uta.edu

Maximilian Liehr
NY CREATES
Albany, NY, USA
mliehr@ny-creates.org

Sumit Kumar Jha
Florida International University
Miami, FL, USA
sjha@fiu.edu

Rickard Ewetz
University of Florida
Gainesville, FL, USA
rewetz@ufl.edu

Nathaniel C. Cady
University at Albany, SUNY
Albany, NY, USA
ncady@albany.edu

***Abstract*—Processing in-memory is projected to reshape the computing landscape of accelerators for data-intensive applications. PATH-based computing is a promising in-memory computing paradigm for evaluating Boolean logic. Computation within the paradigm is performed by controlling the electrical flow of currents within 1 Transistor 1 ReRAM (1T1R) crossbar arrays using nonvolatile memory devices and access transistors. However, this paradigm has only been evaluated in simulation and needs to be verified in fabricated devices. In this paper, we propose an end-to-end framework for accelerating Boolean functions using PATH-based computing. The framework consists of a crossbar fabrication step, a logic synthesis step, and a circuit analysis step. The crossbar fabrication step involves fabricating 8x8 1T1R crossbar arrays using a 65 nm CMOS process. The second step maps the Boolean function into a crossbar design. The last step evaluates performance through crossbar testing and circuit simulation. The framework was evaluated with the fabricated crossbars using 2 and 3-bit multiplier designs. The experimental evaluation using both 8x8 arrays and Spice based simulation yielded similar results: 3-bit multiplier worked as designed while the 2-bit multiplier showed unexpected results. The resulting data will be useful to inform future logic designs for PATH-based computing.**

I. Introduction

The amount of available digital data is projected to reach 175 zettabytes by 2025 [1]. The access to vast amounts of digital data has powered the dominance of data-driven applications such as artificial intelligence algorithms, neural networks [2], and scientific modeling and simulations [3]. However, today's high-performance computing systems based on the traditional von Neumann architecture struggle with accelerating data-intensive applications [4] due to the separation of the computing and memory units, which introduces bandwidth constraints and expensive data transfers [5]. Moreover, there are diminishing returns from further technology scaling due to the end of Dennard scaling and the slowdown of Moore's law [6–8]. This has spurred investigations into alternative computing paradigms and technologies such as quantum computing [9], photonic computing [10], and in-memory computing [11]. Processing in-memory using emerging nonvolatile memory technology is a promising solution strategy to accelerate data intensive applications in future computing systems. The paradigm performs energy-efficient computation in-place, circumventing the need to transfer data back and forth between processor and memory [12].

Processing in-memory can be divided into the analog and digital domain. Analog in-memory computing is extremely energy efficient but lacks precision [13]. Digital in-memory computing offers high precision but is slightly less energy efficient. [14]. Digital in-memory computing can be performed using logic styles such as MAGIC [15], Flow [16], Majority [17], Imply [18], and PATH [19]. The limitation of many of these logic styles is that they require the state of the art nonvolatile memory (NVM) devices to be repeatedly switched. In contrast, each of the nonvolatile memory devices are only required to be switched once when implementing PATH-based computing.

PATH-based computing involves evaluating Boolean functions using the electrical flow of current within a 1T1R crossbar array. It is performed by first programming the nonvolatile memory devices in the crossbar. Next, the input operands are applied to the selector lines of the crossbar. Finally, a voltage is applied to the top-most wordlines and the output of the Boolean function is decoded from the bitlines. While PATH-based computing has been demonstrated to be energy-efficient using simulation on both the logic and system level, it has not been demonstrated in fabricated hardware. Although meaningful research can be performed using simulation, it is critical to verify that the underlying computational concepts can be realized in hardware. This is especially true for emerging technologies where the fabricated devices may have non-ideal properties that are not captured by simulations.

In this paper, an end-to-end framework for accelerating Boolean functions (such as multiplication) on fabricated 1T-1R crossbar arrays using PATH-based computing is proposed. The framework consists of a fabrication step, a logic synthesis step, and a circuit evaluation and simulation step. The primary outcomes can be summarized as follows:

- Fabrication: 8X8 crossbar arrays in 1T1R configuration

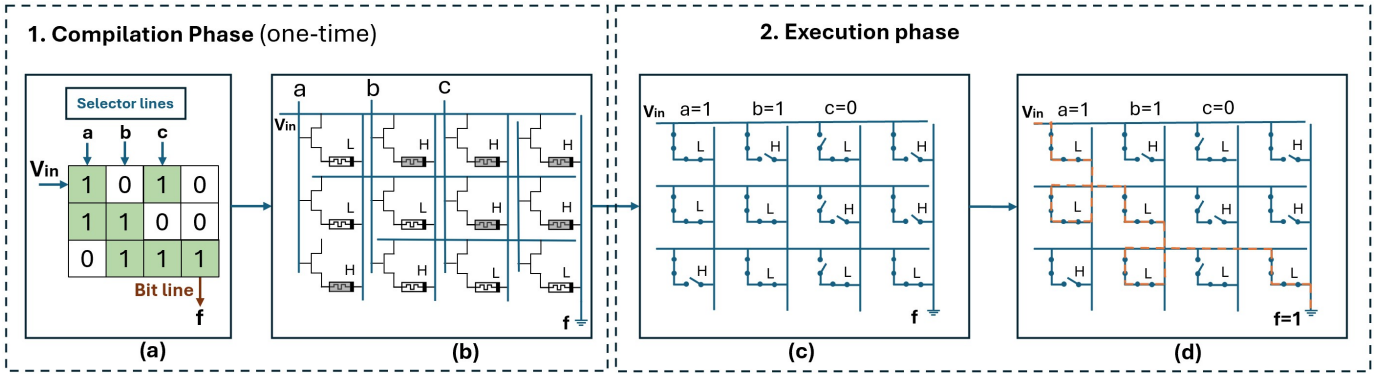

Fig. 1: Overview of the PATH logic style with synthesis and evaluation of Boolean expressions. (a) Resistive states of memristors in the 1TIR crossbar obtained from synthesis. (b) Programing of crossbar based on resistive states. (c) Setting the switch state (ON/OFF) of transistors based on the selector lines charged by literals. (d) The current finding path from Vin to Vout results in the Boolean function evaluating to true.

were fabricated using HfO2-based ReRAM devices and a 65 nm CMOS process, on a 300 mm wafer platform

- Logic Level: We created a synthesis tool for mapping Boolean functions to crossbar designs for PATH-based computing using binary decision diagrams (BDDs).
- Circuit Level: Spice simulations were performed to evaluate whether the logic-level designs can be realized within the crossbar arrays.
- Experimental Evaluation: Testing on the 8x8 arrays showed that our circuit models accurately captured the behavior of the fabricated crossbars. Hower, we observed deviations between the circuit and logic level modeling, which resulted in only one of the two implemented multipliers demonstrating functional correctness in hardware, which emphasizes the need for new logic level modeling and design techniques.

The remainder of the paper is organized as follows: background is given in Section II, methodology is detailed in Section III, fabrication process is presented in Section III-A, experimental evaluation is performed in Section IV, and the paper is concluded in Section VI.

II. Background

A. Memristive Crossbar Memory Arrays

Memristors (implemented in this work as resistive random access memory devices, or ReRAM) are terminal devices proposed by Leon O. Chua in 1972 as the fourth fundamental circuit element after resistors, capacitors, and inductors [20]. They can retain the information of the electric field applied previously in the form of a change in resistance state. Memristors, such as ReRAM, can retain their memory state in the absence of applied voltage, charge, or current, making them useful as nonvolatile memory (NVM) devices [21]. A crossbar memory array [22] is constructed using memristive elements, such as 1T1R cells, to perform the calculations. This comprises wordlines, bitlines, and selectorlines which are arranged as shown in Figure 1. Each Wordline is connected to bitlines via a series of connected memristors and associated access transistors. Each column of access transistors are vertically aligned which can be activated using a selectorline. Both series connected memristors and transistors connecting the bitline to the wordline act as switches to program the circuit. Having a high resistance state (HRS) in the memristor makes the switch OFF and a low resistive state (LRS) makes the switch ON. Applying voltage to the selectorline controls the access transistor.

B. PATH-based Computing

In this section, we discuss the flow of PATH-based computing, which comprises a slow and expensive compilation phase to program the crossbar based on the Boolean expression [19]. This is followed by a fast and inexpensive execution step that executes the Boolean operation based on its literals [19]. Figure 1 shows an overview of the framework for PATH-based computing. A crossbar memory array is programmed with a synthesized Boolean expression to apply the PATH logic style.

Compilation Phase We first consider a Boolean expression of (A + B) | C to be computed using PATH logic style in the crossbar. The input of PATH is a Boolean function represented in a hardware descriptive language (Verilog, VHDL) which is synthesized into the crossbar with several logic synthesis steps discussed in [19]. The logic synthesis outputs resistive states of the memristors that need to be programmed in the crossbar as shown in Figure 1(a) for Boolean expression (A + B | C). As discussed previously, memristors and access transistors can be connected in series for each wordline (horizontal lines) to all bitlines (vertical lines) in a crossbar as shown in Figure 1(b). The memristors in the crossbar are programmed to HRS or LRS based on the result of logic synthesis. The resistive states of memristors are programmed to LRS and HRS by applying precise voltage with controlled current [23], and the resulting resistance state is verified using a write-read-verify scheme that has been described previously [24]. The memristors in the crossbar are marked as 1 for LRS and 0 for HRS as shown in Figure 1(a). The programming of memristors is performed

in the compilation phase (Figure 1(b)) and the switching of transistors and application of read-out voltage is performed in the execution phase, as shown in Figures 1(c) and 1(d).

Execution Phase The transistors in the crossbar are marked as either on or off switches based on the transistor's state. After programming memristors, as shown in Figure 1(b), in conformity with the Boolean expression (A + B | C) the compilation phase is completed. Next, the input operands (a=1, b=0, c=1) of the Boolean expression (A + B | C) are applied to the transistor gates via selector lines, to toggle their state during the execution phase, as shown in Figure 1(c). After programming, the switch state of the transistors based on the operand input to the Boolean expression current is passed from the first bitline, as shown in Figure 1(d). The current now finds the path of least resistance in the crossbar array and flows in that direction. If the resulting current reaches the ground of the last wordline, the Boolean expression is evaluated to true; in the case where the current does not pass through the ground, the Boolean expression evaluates to false. In Figure 1(d) the current finds a path to flow from the start point of Vin to Vout in the crossbar, validating the Boolean expression to true. The PATH logic style is comprised of a slow compilation phase but has a fast execution phase. This makes the PATH logic style outperform other SOTA logic styles in terms of latency and energy usage [19].

III. Methodology

In this paper, we propose an end-to-end framework for accelerating Boolean function using PATH-based computing on 1T1R crossbar arrays. Our framework consists of a i) fabrication process step, ii) a logic synthesis step, and a iii) circuit analysis step. The fabrication process for the crossbar and ReRAM devices is explained in the Section III-A. The mapping of Boolean functions such as multiplication operations into crossbar designs is detailed in Section III-B. The functional correctness of the design is analyzed using circuit simulation in Section III-C. The experimental evaluation in hardware is performed in Section IV. An overview of the flow of the methodology is shown in Figure 2.

A. Fabrication Process

Fabrication and processing of hybrid ReRAM/CMOS 1T1R arrays was performed at the Albany NanoTech complex using a previously developed 65 nm CMOS/ReRAM process, on a 300 mm wafer platform [25]. For this work, HfO2-based ReRAM devices were implemented in the back end of the line (BEOL) at the interface of metal 1 (M1) and metal 2 (M2), and directly integrated with control transistors fabricated in the front end of the line (FEOL). As-fabricated ReRAM arrays are duplicated multiple times per die across the wafer [26], and have been extensively evaluated for electrical performance as a function of fabrication conditions [27]. In this work 8x8 1T1R arrays were implemented, where each of the 64 ReRAM structures are addressable through a combination of 8 source contacts, 8 drain contacts, and 8 transistor (gate) control contacts. This arrangement allows for either individual, single

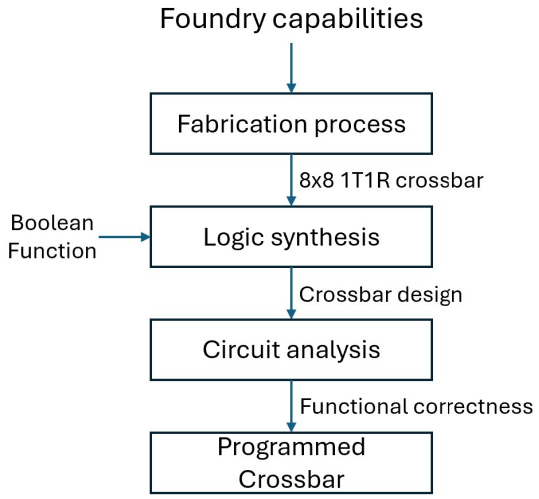

Fig. 2: Flowchart showing how PATH-based computing was implemented and evaluated in 1T1R arrays.

column, or multi-column measurements which are useful for in-memory computing operations such as vector matrix multiplication (VMM) or other forms of in-memory computing, such as flow-based computing [28]. The ReRAM devices used in this work were fabricated using a 70 nm thick TiN bottom electrode (BE) deposited using a subtractive etch approach. This was followed by a 6 nm thick HfO2 switching layer (SL) a 6 nm thick Ti-based oxygen scavenger/exchange layer (OEL), and finally a 40 nm thick TiN inert top electrode (TE), as shown in Figure 3. The ReRAM device active area where the filamentary switching kinetics take place, is defined by the patterned bottom electrode area (50 nm x 50 nm).

B. Logic Synthesis

The input to the synthesis step is a Boolean expression, and the output is the specification to program the crossbar. First, the Boolean expression is converted into a binary decision diagram (BDD) to initiate the PATH logic synthesis [19]. In path-based in-memory computing, a BDD graph $G = (V, E)$ is mapped directly to the 1T1R crossbar. Each node $v_i \in V$ is assigned a wordline, and each edge $e_{ij} \in E$ between nodes v_i and v_j, is implemented through a pair of bitline-selector lines. The selector lines control the access transistors, enabling or disabling paths based on the Boolean variables assigned to them.

This study aimed to investigate the working of an 8x8 1T1R memristive crossbar array to execute multiplexer operations by using PATH-based computing. This involves programming the crossbar based on the resistance values of the PATH-based logic determined during the synthesis phase. The resistance values are obtained from the process of PATH-based synthesis of the Boolean expression to implementation in the crossbar array, as shown Figure 2.

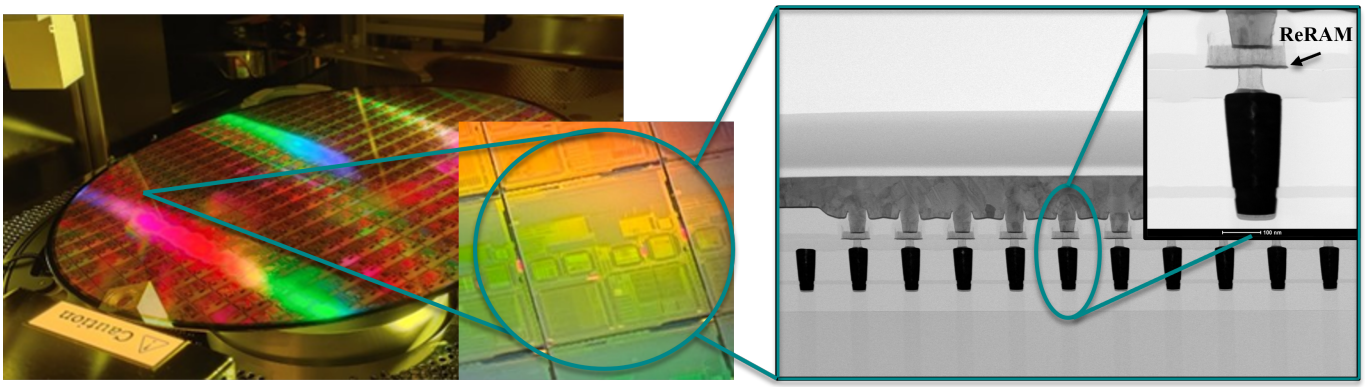

Fig. 3: Composite image showing a full 300mm wafer containing hybrid RRAM/CMOS cells, an image of an individual die within the wafer, and transmission electron microscope (TEM) cross sections of an array of RRAM cells and an individual close-up image of a RRAM cell.

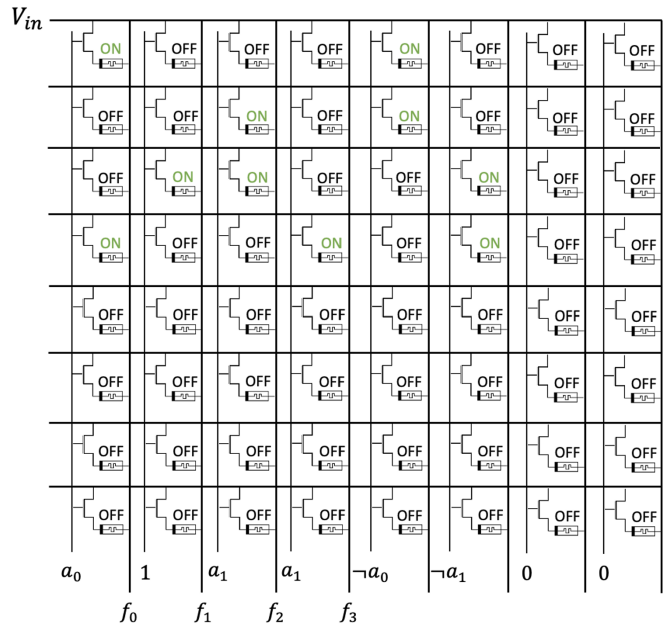

Fig. 4: Design for programmed device states for a 2-bit Multiplier, ON is LRS and OFF is HRS.

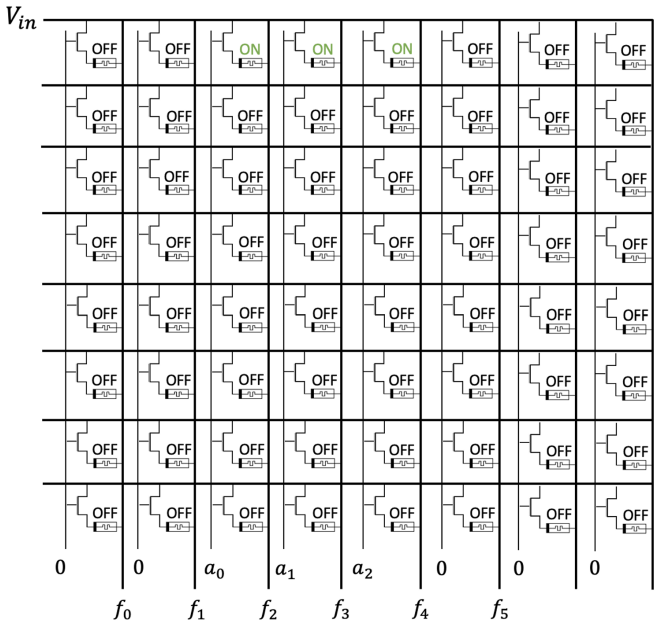

Fig. 5: Design for programmed device states for a 3-bit Multiplier, ON is LRS and OFF is HRS.

C. Circuit Analysis

A basic Spice model of the 8x8 array was developed to simulate the performance of the designed multiplier circuits. 1T1R cells were represented by a single resistor set to the value of the effective resistance measured during testing. The devices in the columns with gate 0, R_{open}, were set to 1 GΩ resistance and HRS and LRS states in the ON gates were set to their programmed values. These simulations were used to verify the results of the multiplier circuits.

IV. Experimental Evaluation

To better understand the transition from simulated PATH-based computing designs to fabricated device measurements, two small-scale designs were developed and tested. A 2-bit and

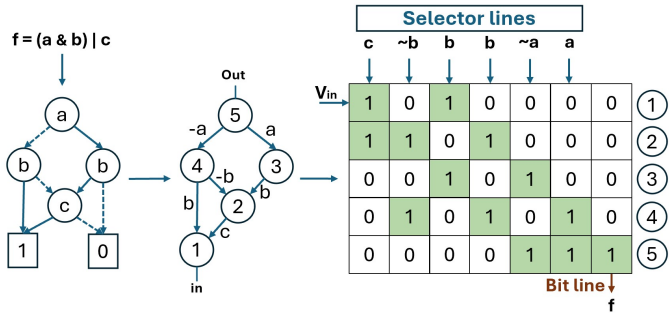

Fig. 6: Path logic synthesis.

Decimal			Binary							
A	B	F	A		B		F			
			a1	a0	b1	b0	f3	f2	f1	f0
0	3	0	0	0	1	1	0	0	0	0
1	3	3	0	1	1	1	0	0	1	1
2	3	6	1	0	1	1	0	1	1	0
3	3	9	1	1	1	1	1	0	0	1

Fig. 7: Truth table for 2-bit multiplier.

Decimal			Binary										
A	B	F	A			B			F				
			a2	a1	a0	b2	b1	b0	f5	f4	f3	f2	f1	f0
0	4	0	0	0	0	1	0	0	0	0	0	0	0	0
1	4	4	0	0	1	1	0	0	0	0	0	1	0	0
2	4	8	0	1	0	1	0	0	0	0	1	0	0	0
3	4	12	0	1	1	1	0	0	0	0	1	1	0	0
4	4	16	1	0	0	1	0	0	0	1	0	0	0	0
5	4	20	1	0	1	1	0	0	0	1	0	1	0	0
6	4	24	1	1	0	1	0	0	0	1	1	0	0	0
7	4	28	1	1	1	1	0	0	0	1	1	1	0	0

Fig. 8: Truth table for 3-bit multiplier.

3-bit design was developed for an 8x8 1T1R crossbar array configuration where each column has 8 transistors sharing a common gate node to help select specific ReRAM devices and reduce unwanted sneak currents (during programming). These two designs, Figure 4 and Figure 5 show the ReRAM devices programmed to LRS or HRS conductance state. Another representation used for later simulation results was R_{open}, this condition was used for ReRAM devices that had their transistor open. For a 2-bit multiplier, each column was assigned either a gate input of 0 or 1 using a combination of two Boolean variables while the 3-bit multiplier used a combination of three Boolean variables. By controlling the gate inputs and having pre-programmed conductance states for each device, the current was intended to flow to specific desired output nodes. Each word and bit line of the 8x8 array was measured using a source measure unit (SMU) to determine any sneak path current to other nodes.

The first design tested was the 3-bit multiplier case seen in Figure 5. All 8 possible cases represented in the truth table were tested. For example, in this design, case 8 (shown in the truth table in Fig 8) indicates a2 = a1 = a0 = 1. This means the common gates in columns 3 to 5 should be turned on. Based on the truth table, outputs f4, f3, and f2 should display outputs of 1 for this case. Hardware testing on the 8x8 arrays verified the designs and generated the correct expected output results, showing no significant sneak path current as was expected. From this experimental verification, a clear threshold of 1E-5 amps was observed and used to differentiate between a 0 and a 1 at the output nodes of the crossbar array. In the case of a2 = a1 = a0 = 1, the output current values for f4, f3, and f2 were above a threshold of 1E-5 amps, as shown in Figure 9. Using this threshold to determine a cutoff for determining 0 vs. 1 outputs, hardware testing for all 8 cases for the 3-bit multiplier agreed with expected results (from the truth table). Once the

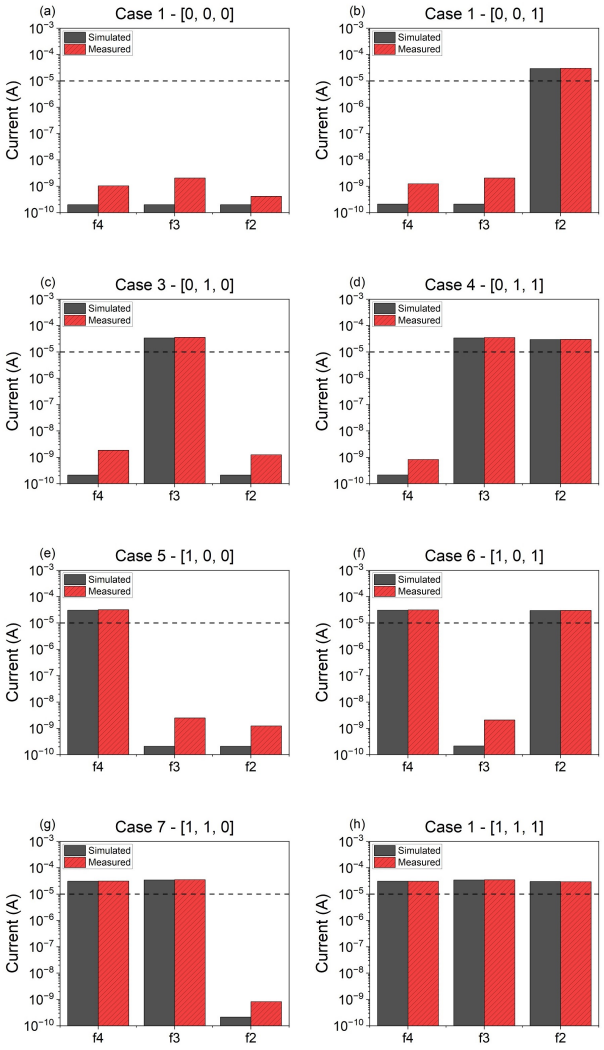

Fig. 9: Results of testing the 3-bit multiplier design vs. simulated results for different input cases. Grey bars represent Spice simulation results and red bars represent arrays test results.

device and array functionality were verified using this simple design, a 2-bit multiplier design was tested on the 8x8 arrays. In this design, the current is expected to follow trajectories outlined by certain devices set to the LRS and generate outputs of 1 via so-called sneak path currents that percolate through the 1T1R array; however, the output results as shown in Fig 10 did not match the expected outcomes in the 2-bit multiplier truth table shown in Figure 7. For example, in case 2 for the 2-bit multiplier, f1 received an output of 0 while a 1 was expected in the truth table. Another interesting phenomenon was that currents up to a few hundred nA were measured at the inputs of row 2 through 8. This suggests that there may be sneak path current that is diverted from the desired outputs and percolating to the inputs of the rows; however, the magnitude of these currents does not affect the total output of the node (0 or 1).

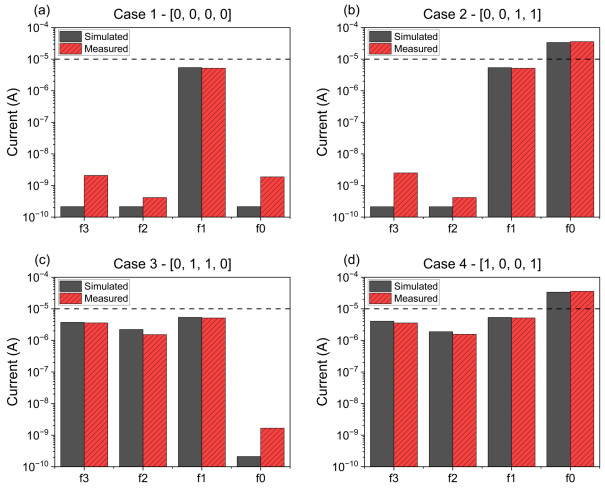

Fig. 10: Results of testing the 2-bit multiplier design vs. simulated results for different input cases. Grey bars represent Spice simulation results and red bars represent arrays test results.

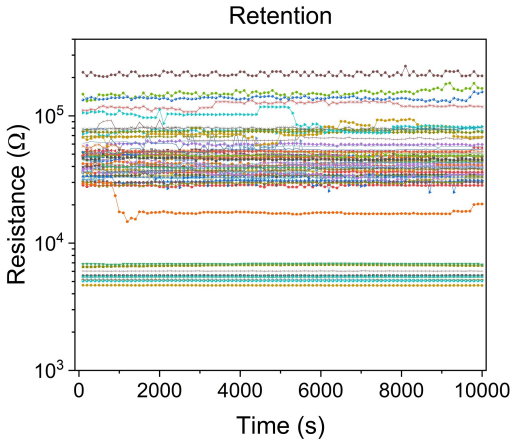

Fig. 11: Results of 10,000 sec duration retention testing for all 64 devices in 8X8 crossbar array.

To rule out the effect of resistance changes during experimental testing, ReRAM resistance drift was investigated. For this, the resistance values of each device in the 8x8 array were measured at an interval of 100 seconds for 10,000 seconds. As seen in the retention data in Figure 11, all devices in the LRS maintained stable resistance values for the duration of the test, while some devices in the HRS exhibited subtle resistance shifts. This is likely due to charge trap/detrap events that are occasionally observed in HfO2 ReRAM devices [29]. While this variation might cause minor current variations, the magnitude of the shifts are low and should not significantly affect the performance of the multiplier circuits.

Spice-based simulations were performed for the multiplier circuits. All devices in a column with gate OFF were set to R_{open} with a resistance of 1 GΩ while devices in columns with gate ON were set to their measured resistance values. As shown in Figure 10 and Figure 9, Spice simulations

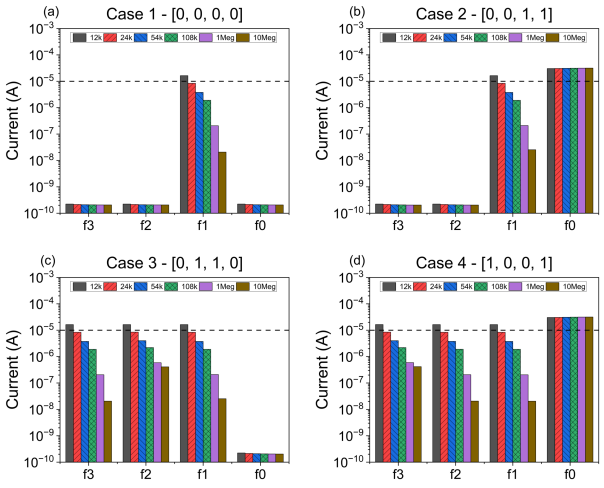

Fig. 12: Resistance values of devices in LRS and HRS during 2-bit multiplier testing, R_{off} is tested at (left to right on graph) 12k, 24k, 54k, 108k, 1 MΩ, and 10Ω.

agreed well with experimental results for both 2-bit and 3-bit multiplier designs. As shown in Figure 10, 2-bit multiplier simulations verified the deviation from expected circuit performance that was observed during hardware testing, namely that sneak path currents at certain output nodes were significantly lower than expected for an output value of "1". Simulations also verified that, in the case of 8x8 arrays, the first row of devices primarily determines the output currents. This suggests that by implementing an input at only one location, the 8x8 arrays effectively act as a current divider, and that future designs will need to be optimized to better suit the crossbar array architecture.

V. Analysis of Hardware Requirements using Circuit-Level Simulation

1) Memory Window Variability: To investigate the impact of ReRAM memory window on the performance of the multiplier circuit, the 2-bit multiplier was simulated with varying R_{off} (HRS) device resistance values. All ReRAM cells in OFF gate columns, R_{open}, were set to 1 GΩ. In columns with gate ON, resistance values for individual devices in the HRS were varied from 12 kΩ to 10 MΩ while devices representing the LRS were set to 6 kΩ. The resulting data are shown in Figure 12 where each of the 6 colors correspond to a different HRS value, the y-axis corresponds to the output currents in log-scale, and the dotted line shows the threshold at which the measured current was considered a "1" output in the truth table, for prior experiments.

Figure 12 shows that the outputs at node f0 were unaffected by changes in HRS values. This is likely because the device in row 1 column 1 is set to LRS, which allows it to pull significant amounts of current when the gate is set to 1 for column 1, regardless of the HRS values of other devices in row 1. For case 1, where the output of f1 should be 0, a 12 kΩ HRS value yields an f1 output of 1. As HRS is increased,

however, the current at f1 decreases, and the output trends towards 0. The opposite is observed in case 2, where based on the truth table in Figure 7, f1 should have the output of 1. At 12 kΩ HRS, f1 yields an output of 1, but f1 trends towards an output of 0 as HRS increases. Case 3 and case 4 are even more interesting because both of these trends can be observed. In case 3, f3 yields the expected output of 1 for HRS of 12 kΩ, but trends towards 0 as HRS increases; however, f2 and f1 should yield the expected output of 0 but yield a 1 for HRS of 12 kΩ, and trend towards 0 as HRS increases. In case 4, we observed the same pattern but with different expected outputs. In case 4, f3, f2, and f1 yield the output of 1 for an HRS of 12 kΩ, but f2 and f1 are expected to be 0 while f3 is expected to be 1. All 3 outputs trend towards 0 as HRS becomes larger. In general, columns where row 1 devices were set to LRS did not exhibit any meaningful differences with varying HRS values. As expected, in the columns where row 1 devices were set to HRS, the current outputs in the column decrease as the magnitude of the HRS increases.

2) Resistance Variability: Lastly, the resistance variability of ReRAM cells within the crossbar array was assessed. Figure 13a shows the R_{on} and R_{off} spread for the 2-bit measurements performed using the 8x8 crossbar array. Overall, a median value of 5.72 kΩ and 57.45 kΩ was found for R_{on} and R_{off} respectively showcasing a MW of ≈10 which is similar to other ReRAM devices found in literature [30, 31]. While this MW is sufficiently large for most applications, the variability observed, particularly in R_{off}, indicate possible variable current fluctuation compared to ideal simulation operations. A heat map of the device resistance values in the 8x8 array is shown in Figure 13b, further highlighting the variability in R_{off} values. The effect of this variability should be investigated in the future with a focus on simulations, highlighting the impact of cell-to-cell resistance variability on digital in-memory computing performance.

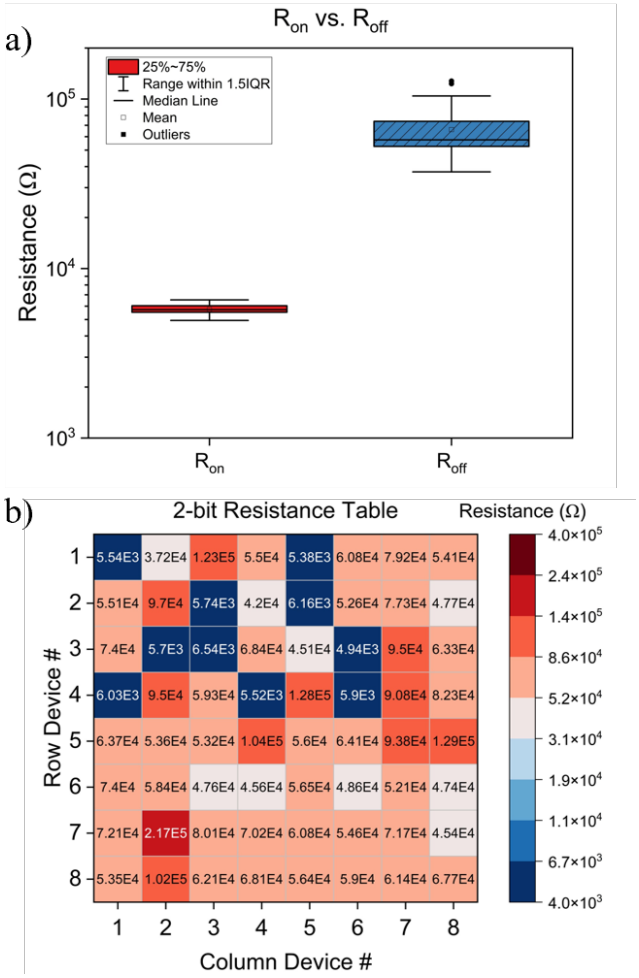

Fig. 13: a) Box plot of resistance values of devices in LRS and HRS during 2-bit multiplier testing. b) Heat map of resistance values of devices in LRS and HRS during 2-bit multiplier testing.

VI. Summary and Conclusions

PATH-based in-memory computing using 1T1R arrays has the potential to achieve high levels of efficiency and speed in digital in-memory computing. This work demonstrated the design and implementation of 2-bit and 3-bit multipliers using PATH-based in-memory computing within 8X8 1T1R crossbar arrays. The digital in-memory computing for these multipliers relies on current flow percolating through the crossbar array via the least resistive path (sneak paths) based on the pre-programmed resistance values of the 1T1R cells. The 3-bit multiplier testing within the arrays produced results in agreement with the PATH-based design, while the 2-bit multiplier testing showed discrepancies between the PATH-based design and the test results. These discrepancies were later confirmed via Spice simulations which produced similar results as the arrays testing did for 2-bit multiplier, implying that the design for 2-bit multiplier will need to be modified in the future. To further investigate the causes for the 2-bit multiplier testing results, the impact of 1T1R cell resistance drift on multiplier performance was evaluated, revealing no significant drift or impact on multiplier performance. In addition, the effects of varying memory window (R_{off}/R_{on}) for 1T1R cells was evaluated through simulation by varying R_{off} values. Memory window variation via simulations showed significant performance differences, and these data will be evaluated to inform future PATH-based in-memory compute designs. Simulation results confirm array testing results, showing that the resistance level of devices in the first row of the crossbar array heavily impacts percolation of the current flow through the array, and hence the performance of the multiplier computation results. Thus, future designs need to better compensate for the impact of current shunting through the array, especially with respect to the first row of memory cells. Further, repeated measures of 1T1R cell resistance show higher variability in HRS than LRS, as has been observed previously, and could directly impact in-memory compute operations. In response to this cell-to-cell variability, simulations and design efforts for PATH-based in-memory computation architectures must be adjusted to better compliment these 1T1R crossbar arrays.

REFERENCES

[1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," tech. rep., International Data Corporation, Framingham, 2018.

[2] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (pinns) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021.

[3] J. S. Sims, W. L. George, S. G. Satterfield, H. K. Hung, J. G. Hagedorn, P. M. Ketcham, T. J. Griffin, S. A. Hagstrom, J. C. Franiatte, G. W. Bryant, *et al.*, "Accelerating scientific discovery through computation and visualization ii," *Journal of Research of the National Institute of Standards and Technology*, vol. 107, no. 3, p. 223, 2002.

[4] G. Pedretti *et al.*, "A spiking recurrent neural network with phase-change memory neurons and synapses for the accelerated solution of constraint satisfaction problems," *JXCDC*, vol. 6, no. 1, pp. 89–97, 2020.

[5] J. Backus, "Can programming be liberated from the von neumann style?: A functional style and its algebra of programs," *CACM*, vol. 21, no. 8, pp. 613–641, 1978.

[6] W. Haensch, "Scaling is over—what now?," in *2017 75th Annual Device Research Conference (DRC)*, IEEE, 2017.

[7] L. Eeckhout, "Is moore's law slowing down? what's next?," *IEEE Micro*, vol. 37, no. 04, pp. 4–5, 2017.

[8] S. Petrenko, *Big Data Technologies for Monitoring of Computer Security: A Case Study of the Russian Federation*, ch. Limitations of Von Neumann Architecture, pp. 115–173. Springer, 2018.

[9] A. Steane, "Quantum computing," *Reports on Progress in Physics*, vol. 61, no. 2, p. 117, 1998.

[10] B. J. Shastri *et al.*, "Photonics for artificial intelligence and neuromorphic computing," *Nature Photonics*, vol. 15, no. 2, pp. 102–114, 2021.

[11] B. Li, B. Yan, and H. Li, "An overview of in-memory processing with emerging non-volatile memory for data-intensive applications," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pp. 381–386, 2019.

[12] D. Bhattacharjee and A. Chattopadhyay, "Synthesis and technology mapping for in-memory computing," in *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann*, pp. 317–353, Springer, 2022.

[13] S. Channamadhavuni, S. Thijssen, S. K. Jha, and R. Ewetz, "Accelerating ai applications using analog in-memory computing: Challenges and opportunities," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pp. 379–384, 2021.

[14] M. R. H. Rashed *et al.*, "Logic synthesis for digital in-memory computing," in *Proceedings of the 41st IEEE/ACM ICCAD*, pp. 1–9, 2022.

[15] S. Kvatinsky *et al.*, "Magic—memristor-aided logic," *TCAS-II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.

[16] S. Thijssen, S. K. Jha, and R. Ewetz, "Compact: Flow-based computing on nanoscale crossbars with minimal semiperimeter," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 232–237, IEEE, 2021.

[17] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *2014 51st ACM/EDAC/IEEE DAC*, pp. 1–6, IEEE, 2014.

[18] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2013.

[19] S. Thijssen, S. K. Jha, and R. Ewetz, "Path: Evaluation of boolean logic using path-based in-memory computing," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1129–1134, 2022.

[20] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, pp. 507–519, Sep 1971.

[21] A. Ascoli, R. Tetzlaff, L. O. Chua, J. P. Strachan, and R. S. Williams, "History erase effect in a non-volatile memristor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 3, pp. 389–400, 2016.

[22] M. Wang *et al.*, "A selector device based on graphene–oxide heterostructures for memristor crossbar applications," *Applied Physics A: Solids and Surfaces*, vol. 120, no. 2, pp. 403–407, 2015.

[23] J. Solanki, K. Beckmann, J. Pelton, N. Cady, and M. Liehr, "Effect of resistance variability in vector matrix multiplication operations of 1t1r reram crossbar arrays using an embedded test platform," in *Proceedings of the IEEE 32nd Microelectronics Design Test Symposium (MDTS)*, pp. 1–5, 2023.

[24] M. Liehr, J. Hazra, K. Beckmann, S. Rafiq, and N. Cady, "Impact of switching variability of 65nm cmos integrated hafnium dioxide-based reram devices on distinct level operations," in *Proceedings of the IEEE International Integrated Reliability Workshop (IIRW)*, pp. 1–4, 2020.

[25] K. Beckmann, H. Manem, and N. Cady, "Performance enhancement of a time-delay puf design by utilizing integrated nanoscale reram devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 3, pp. 304–316, 2016.

[26] N. Cady, K. Beckmann, W. Olin-Ammentorp, G. Chakma, S. Amer, R. Weiss, S. Sayyaparaju, M. Adnan, J. Murray, M. Dean, *et al.*, "Full cmos-memristor implementation of a dynamic neuromorphic architecture," in *GOMACTech Conference*, 2018.

[27] M. Liehr, J. Hazra, K. Beckmann, W. Olin-Ammentorp, N. Cady, R. Weiss, S. Sayyaparaju, G. Rose, and J. Van Nostrand, "Fabrication and performance of hybrid reram-cmos circuit elements for dynamic neural networks," in *Proceedings of the International Conference on Neuromorphic Systems*, pp. 1–4, 2019.

[28] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, pp. 52–59, Jan 2018.

[29] N. Raghavan, R. Degraeve, A. Fantini, L. Goux, D. J. Wouters, G. Groeseneken, and M. Jurczak, "Modeling the impact of reset depth on vacancy-induced filament perturbations in hfo2 rram," *IEEE electron device letters*, vol. 34, no. 5, pp. 614–616, 2013.

[30] Y. Y. Chen, B. Govoreanu, L. Goux, R. Degraeve, A. Fantini, G. S. Kar, D. J. Wouters, G. Groeseneken, J. A. Kittl, M. Jurczak, *et al.*, "Balancing set/reset pulse for $10^{1}0$ endurance in hfo2/hf 1t1r bipolar rram," *IEEE Transactions on Electron devices*, vol. 59, no. 12, pp. 3243–3249, 2012.

[31] Y.-T. Su, H.-W. Liu, P.-H. Chen, T.-C. Chang, T.-M. Tsai, T.-J. Chu, C.-H. Pan, C.-H. Wu, C.-C. Yang, M.-C. Wang, *et al.*, "A method to reduce forming voltage without degrading device performance in hafnium oxide-based 1t1r resistive random access memory," *IEEE Journal of the Electron Devices Society*, vol. 6, pp. 341–345, 2018.