Making Real Memristive Processing-in-Memory Faster and Reliable

Shahar Kvatinsky

Viterbi Faculty of Electrical and Computer Engineering
Technion – Israel Institute of Technology
Haifa, Israel 3200003
shahar@ee.technion.ac.il

Abstract— **Memristive technologies are attractive candidates to replace conventional memory technologies, and can also be used to perform logic and arithmetic operations using a technique called 'stateful logic.' Combining data storage and computation in the memory array enables a novel non-von Neumann architecture, where both the operations are performed within a memristive Memory Processing Unit (mMPU). The mMPU relies on adding computing capabilities to the memristive memory cells without changing the basic memory array structure. The use of an mMPU alleviates the primary restriction on performance and energy in a von Neumann machine, which is the data transfer between CPU and memory. Here, the various aspects of mMPU are discussed, including its architecture and implications on the computing system and software, as well as examining the microarchitectural aspects. We show how mMPU can be improved to accelerate different applications and how the poor reliability of memristors can be improved as part of the mMPU operation.**

Keywords—memristor, stateful logic, memristive memory processing unit, memristor aided logic (MAGIC)

I. INTRODUCTION

Computing systems are typically designed in von Neumann architecture, or an ameliorated version of it, which separates the memory and processing space. In these systems, programs are executed by moving data between the processing unit and memory using specific operations (load/store). While this programming model is simple, the performance of the system is limited by the memory access time, which is substantially higher than the computing time itself. This performance bottleneck (known as the "memory wall") has become even more severe over the years because CPU speed has improved much more than memory speed and bandwidth [1]. Moreover, many modern workloads have high and unstructured data volumes with limited locality, reducing the effectiveness of data caching.

Processing-in-memory (PIM) is an attractive solution to alleviate the memory wall. One such architecture is the *memristive memory processing unit* (mMPU) [2-6]. In the mMPU, memristors are used to construct a dense nonvolatile memory that can also be used to perform *stateful logic* operations, using the same cells. At different times of the program execution, memristors can serve as input, output, latches, and memory cells, enabling real PIM.

In this work, we describe the mMPU architecture, based on a stateful logic [7-8] technique called memristor aided logic (MAGIC) [9-10]. Then, several approaches to improve the performance and reliability of the mMPU are presented.

II. MAGIC

Several techniques have been proposed to perform logic with memristors [7, 11-15]. *Memristor-Aided loGIC* (MAGIC) [9] is a stateful, in-memory, flexible logic family. In MAGIC, only a single voltage V_G is used to perform a specific function. The initial states (resistance) of the input memristors serve as the input of the logic gate, while the final state (resistance) of the output memristor is the result of the logical operation. Prior to the operation, the output is usually initialized to a known logical state. During the operation, the applied voltage forms a voltage divider and the exact voltage across the output memristor depends on the inputs. Therefore, the output device may switch and by that constitutes the desired logical operation.

The original MAGIC paper [9] presented several functions and concluded that NOR can be performed within the memristive crossbar array. Since then, several additional gates to be performed within the crossbar have been proposed for different memristive technologies [16-17, 21] and were experimentally realized [18, 37]. MAGIC gates can be performed simultaneously on multiple rows/columns within the array. This enables massive parallel execution of different vector operations. The operation of a vector of basic MAGIC NOR gates within a memristive crossbar array is illustrated in Figure 1.

III. MMPU

The mMPU [2-6] is a standard memristive memory with a few modifications that enable the support of MAGIC-based PIM instructions. In other words, the mMPU functions as a standard memory that supports memory operations (*i.e.*, read and write) with additional PIM capabilities, and thus it is backward compatible with the von Neumann computing scheme. The mMPU architecture is shown in Figure 2. To support PIM instructions, the memory controller [19, 22], the memory protocol [20], and the peripheral circuits (*i.e.*, voltage drivers and row/column decoders) must be modified to support MAGIC instructions [10]. The mapping of data is also modified to maintain persistency and coherence. Note, however, that the memory crossbar array structure itself is not modified and can be in different forms of memristive memory cells, such as 1R (single memristor per cell), 1S1R (1 selector, 1 memristor), and 1T1R (1 transistor, 1 memristor).

IV. IMPROVING MMPU PERFORMANCE

The preliminary performance results of the mMPU have shown the potential to improve the throughput and execution time compared to conventional computing systems for different applications. Imani *et al.* demonstrated more than
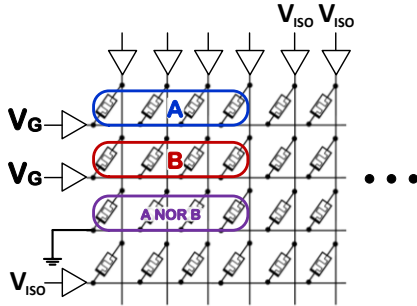
Figure 1. A MAGIC NOR operation between two row vectors *A* and *B* is performed within the memristive memory array by applying V_G to the wordlines of the input memristors, ground to the wordline of the output memristor, and V_{ISO} to isolate unselected bitlines and wordlines. The operation takes a single clock cycle regardless of the vector size of *A* and *B*.
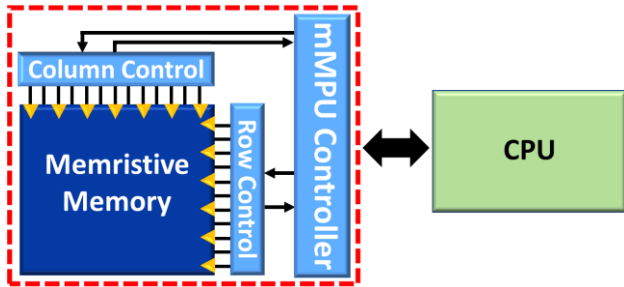

Figure 2. Memristive memory processing unit (mMPU). The mMPU is built from memristive memories orchestrated by the mMPU controller. The mMPU supports regular memory operations (read/write), as well as logical operations.
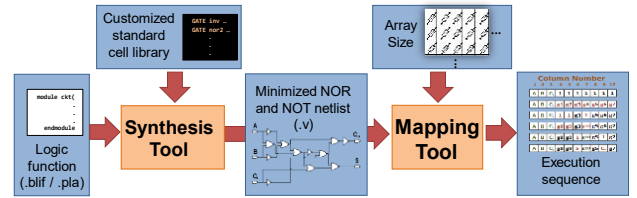

Figure 3. SIMPLER MAGIC flow. The desired function is represented as a .blif/.pla file and is generated as a netlist using a standard synthesis tool for a specific technology libraray. Then, the netlist goes into a mapping tool that maps the logic gates into specific memristors in a sequential manner, based on the physical memory array constraints.

100X speedup versus GPU for deep neural network training [23]. Haj Ali *et al.* have shown a similar speedup for image processing compared to previously proposed memristive accelerators [2, 24]. The Bitlet model [25] is an analytical model to determine whether a specific application will benefit from the mMPU compared to a von Neumann machine.

The efforts to improve the performance of the mMPU focus on several aspects. The mMPU controller orchestrates the mMPU [19, 26] based on the specific MAGIC operations that are supported. AbstractPIM [27-28] is the first effort to explore the software-hardware interaction of the mMPU, by defining different instruction set architectures (ISA) for different logic primitives and evaluating the performance. For the processor design, several synthesis tools have been proposed to support automatic generation of the mapping and execution sequence for any desired function. SIMPLE [29] is based on solving an optimization to minimize the latency of the execution, while its successor SIMPLER [30] takes a different approach. In SIMPLER, heuristics are used to improve the runtime of the tool. The target of SIMPLER is increasing the throughput, rather than lowering the latency. SIMPLER limits the execution to a single row in the memory array and by that enables execution of multiple operations concurrently. The flow of SIMPLER is shown in Figure 3.

V. Improving mMPU Reliability

Non-idealities influence the performance and reliability of the mMPU [31-33]. Wald *et al.* [34] explored the influence of process and environmental variation on the execution of MAGIC. Talati *et al.* [35] evaluated the cost of non-ideal interconnect and the cost of internal data movement within the mMPU. Hoffer *et al.* [18] showed experimentally that some memristive technologies fail to execute MAGIC operations due to their voltage threshold values and proposed new gates that are appropriate for the specific memristor properties.

Recently, Leitersdorf *et al.* [36] proposed using internal error correcting codes (ECC) to overcome soft errors in the memristive devices. The difference between standard ECC and the proposed technique is that in the standard manner, the encoding is performed during a read operation, while in the mMPU the aim is to perform the encoding within the memory, without reading the values prior to the logic operations. Leitersdorf *et al.* proposed a diagonal code that can be performed using MAGIC operations as part of the logical execution, and by that improve the reliability of the mMPU and increase the mMPU mean time between failures by nine orders of magnitude.

VI. Conclusion

Processing-in-memory is an attractive approach to overcome the memory wall. However, there are many challenges that need to be considered in order to make PIM systems, such as the mMPU, practical and efficient. These challenges include the entire design stack from the physical non-idealities of the technology up to the software definitions and the programming model.

References

[1] A. Pedram, S. Richardson, S. Galal, S. Kvatinsky, and M. Horowitz, "Dark Memory and Accelerator-Rich System Optimization in the Dark Silicon Era", *IEEE Design and Test*, Vol. 34, No. 2, pp. 39-50, April 2017.

[2] A. Haj Ali, R. Ben Hur, N. Wald, R. Ronen, and S. Kvatinsky, "Not in Name Alone: A Memristive Memory Processing Unit for Real In-Memory Processing," *IEEE Micro*, Vol. 38, No. 5, pp. 13-21, September/October 2018.

[3] N. Talati, R. Ben-Hur, N. Wald, A. Haj Ali, J. Reuben, and S. Kvatinsky, "mMPU – A Real Processing-in-Memory Architecture to Combat the von Neumann Bottleneck," Applications of Emerging Memory Technology, *The Springer Series in Advanced Microelectronics*, M. Suri (Ed.), Springer, Chapter 8, pp. 191-213, 2020.

[4] A. Eliahu, R. Ben-Hur, A. Haj-Ali, and S. Kvatinsky, "mMPU: Building a Memristor-Based General-Purpose In-Memory Computation Architecture," *Multi-Processor System-on-Chip 1*

Architectures, L. Andrade and F. Rousseau (Ed.), Chapter 6, pp. 119-132 March 2021.

[5] R. Ben-Hur and S. Kvatinsky, "Memory Processing Unit for In-Memory Processing," *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 171-172, July 2016.

[6] S. Kvatinsky, "Real Processing-in-Memory with Memristive Memory Processing Unit (mMPU)," *Proceeding of the IEEE International Conference on Application-Specific Systems, Archtiraectures and Processors*, July 2019.

[7] J. Reuben, R. Ben Hur, N. Wald, N. Talati, A. Haj Ali, P.-E. Gaillardon, and S. Kvatinsky, "A Taxonomy and Evaluation Framework for Memristive Logic," *Handbook of Memristor Networks*, L. O. Chua, G. Sirakoulis, A. Adamatzky, (Eds.), pp. 1065-1099, Springer 2019.

[8] J. Reuben, R. Ben Hur, N. Wald, N. Talati, A. Haj Ali, P.-E. Gaillardon, and S. Kvatinsky, "Memristive Logic: A Framework for Evaluation and Comparison," *Proceeding of the IEEE International Symposium on Power and Timing Modeling, Optimization and Simulation*, pp. 1-8, September 2017.

[9] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MAGIC – Memristor Aided LoGIC," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 61, No. 11, pp. 895-899, November 2014.

[10] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic Design within Memristive Memories Using Memristor Aided loGIC (MAGIC)," *IEEE Transactions on Nanotechnology*, Vol. 15, No. 4, pp. 635-650, July 2016.

[11] S. Kvatinsky, N. Wald, G. Satat, 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)*, Vol. 22, No. 10, pp. 2054-2066, October 2014.

[12] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based IMPLY Logic Design Flow," *Proceedings of the IEEE International Conference on Computer Design*, pp.142-147, October 2011.

[13] Y. Levy, J. Bruck, Y. Cassuto, E. G. Friedman, A. Kolodny, E. Yaacobi, and S. Kvatinsky, "Logic Operation in Memory Using a Memristive Akers Array," *Microelectronics Journal*, Vol. 45, No. 11, pp. 1429-1437, November 2014.

[14] S. Kvatinsky, N. Wald, G. Satat, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MRL – Memristor Ratioed Logic," *Proceedings of the International Cellular Nanoscale Networks and their Applications*, pp. 1-6, August 2012.

[15] E. Amrany, A. Drory, and S. Kvatinsky, "Logic Design with Unipolar Memristors," *Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1-5, September 2016.

[16] N. Peled, R. Ben-Hur, R. Ronen, and S. Kvatinsky, "X-MAGIC: Enhancing PIM with Input Overwriting Capabilities," *Proceedings of the IFIP/IEEE VLSI-SoC*, pp. 64-69, October 2020.

[17] J. Louis, B. Hoffer, and S. Kvatinsky, "Performing Memristor Aided Logic (MAGIC) using STT-MRAM," *Proceedings of the IEEE International Conference on Electronics Circuits and Systems*, November 2019.

[18] B. Hoffer, V. Rana, S. Menzel R. Waser, and S. Kvatinsky, "Experimental Demonstration of Memristor Aided Logic (MAGIC) using Valence Change Memory (VCM)," *IEEE Transactions on Electron Devices*, Vol. 67, No. 8, pp. 3115-3122, August 2020.

[19] R. Ben-Hur and S. Kvatinsky, "Memristive Memory Processing Unit (MPU) Controller for In-Memory Processing", *Proceedings of the IEEE International Conference on Science of Electrical Engineering*, pp. 1-5, November 2016.

[20] N. Talati, H. Ha, B. Perach, R. Ronen, and S. Kvatinsky, "CONCEPT: A Column Oriented Memory Controller for Efficient Memory and PIM Operations in RRAM," *IEEE Micro*, Vol/ 39, No. 1, pp. 33-43, January/February 2019.

[21] N. Wald and S. Kvatinsky, "Design Methodology for Stateful Memristive Logic Gates," *Proceedings of the IEEE International Conference on Science of Electrical Engineering*, pp. 1-5, November 2016.

[22] R. Ben-Hur, N. Talati, and S. Kvatinsky, "Algorithmic Considerations in Memristive Memory Processing Units (MPU)," Proceedings of the *International Cellular Nanoscale Networks and their Applications*, pp. 1-2, August 2016.

[23] M. Imani, S. Gupta, Y. Kim, and T. Rosing, " FloatPIM: In-Memory Acceleration of Deep Neural Network Training with High Precision," *Proceedings of the IEEE International Symposium on Computer Architecture*, pp. 802-815, June 2019.

[24] A. Haj Ali, R. Ben-Hur, N. Wald, R. Ronen, and S. Kvatinsky, "IMAGING - In-Memory AlGorithms for Image ProcessiNG," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 65, No. 12, pp. 4258-4271, December 2018.

[25] R. Ronen, A. Eliahu, O. Leitesdorf, N. Peled, K. Korgaonkar, A. Chattopadhyay, and S. Kvatinsky, "Bitlet Model: A Parametrized Analytical Model to Compare PIM and CPU Systems," *ACM Journal on Emerging Technologies in Computing Systems*, (in press).

[26] A. Haj Ali, R. Ben-Hur, N. Wald, and S. Kvatinsky, "Efficient Algorithms for In-Memory Fixed Point Multiplication Using MAGIC," *Proceeding of the IEEE International Symposium on Circuits and Systems*, pp. 1-5, May 2018.

[27] A. Eliahu, R. Ben Hur, R. Ronen, and S. Kvatinsky, "abstractPIM: Bridging the Gap Between Processing-in-Memory Technology and Instruction Set Architecture," *Proceedings of the IFIP/IEEE VLSI-SoC*, pp. 28-33, October 2020.

[28] A. Elihau, R. Ben-Hur, R. Ronen, and S. Kvatinsky, "A Technology Backward-Compatible Compilation Flow for Processing-in-Memory," *VLSI-SoC: Open Source VLSI Technologies, IFIP Advances in Information and Communication Technology*, P.-E. Gaillardon, S. Kvatinsky, A. Calimerri, R. Reis, (Eds.), Springer, 2021 (in press).

[29] R. Ben Hur, N. Wald, N. Talati, and S. Kvatinsky, "SIMPLE MAGIC: Synthesis and Mapping of Boolean Functions for Memristor Aided Logic (MAGIC)," *Proceeding of the IEEE International Conference on Computer Aided Design*, pp. 225-232, November 2017.

[30] R. Ben-Hur, R. Ronen, A. Haj-Ali, D. Bhattacharjee, A. Eliahu, N. Peled, and S. Kvatinsky, "SIMPLER MAGIC: Synthesis and Mapping of In-Memory Logic Executed in a Single Row to Improve Throughput," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 39, No. 10, pp. 2434-2447, October 2020.

[31] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-Path Constraints in Memristor Crossbar Arrays," *Proceedings of the IEEE International Symposium on Information Theory*, pp. 156-160, July 2013.

[32] A. Doz, I. Goldstein, and S. Kvatinsky, "Analysis of the Row Grounding Method in a Memristor-Based Crossbar Array," *International Journal of Circuit Theory and Applications*, Vol. 46, No. 1, pp. 122-137, January 2018.

[33] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-Theoretic Sneak Path Mitigation in Memristor Crossbar Arrays," *IEEE Transactions on Information Theory*, Vol. 62, No. 9, pp. 4801-4814, September 2016.

[34] N. Wald and S. Kvatinsky, "Influence of Parameter Variations and Environment for Real Processing-In-Memory using Memristor Aided Logic (MAGIC)," *Microelectronics Journal*, Vol. 86, pp. 22-33, April 2019.

[35] N. Talati, A. Haj Ali, R. Ben Hur, N. Wald, R. Ronen, P.-E. Gaillardon, and S. Kvatinsky, "Practical Challenges in Delivering the Promises of Real Processing-in-Memory Machines," *Proceedings of the Design Automation and Test in Europe*, pp. 1628-1633, March 2018.

[36] O. Leitersdorf, B. Perach, R. Ronen, and S. Kvatinsky, "Efficient Error-Correcting-Code Mechanism for High-Throughput Memristive Processing-in-Memory," *Proceedings of the Design Automation Conference*, December 2021 (in press).

[37] B. C. Jang, S. Y. Yang, H. Seong, S. K. Kim, J. Choi, S. G. Im, and S.-Y. Choi, "Zero-static-power nonvolatile logic-in-memory circuits for flexible electronics," *Nano Research*, Vol. 700, No. 10, pp. 2459-2470, April 2017.