Memory Processing Unit for In-Memory Processing

Rotem Ben Hur and Shahar Kvatinsky, Member, IEEE
The Andrew and Erna Viterbi Faculty of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 3200003, ISRAEL

***Abstract*— Performance and energy of modern computers, usually built as von Neumann machines, are primarily limited by data transfer from the memory to the CPU and vice versa. Only a true non-von Neumann architecture, where data is processed and stored within the same unit can remove this bottleneck. Using emerging non-volatile resistive memory technologies (namely, memristors) enables the development of Memory Processing Unit (MPU) – a novel non-von Neumann architecture. MPU relies on adding computing capabilities to the memristive memory cells without changing the basic memory array structure, and is compatible with existing computing systems. This paper describes the MPU architecture and examines its controller.**

Keywords— Memristive systems, memristor, logic design, MAGIC, Crossbar memory, memory controller, CPU, MPU.

I. *INTRODUCTION*

Since the 1940s, computers have been built in a von Neumann architecture, where data processing and data storage are separated into different units (namely, CPU and memory). von Neumann architecture has become so popular that today almost any type of computing machine is a von Neumann machine (or an improvement thereof). While von Neumann architecture is relatively simple to program and design, it suffers from several disadvantages, usually collectively called *von Neumann bottleneck*. One of the primary sources to this bottleneck is the need in von Neumann machines to supply the CPU with data from the memory, result in lower performance and higher energy.

Today this bottleneck is even more severe since the speed of CPUs has scaled over the past decades according to Moore's law by a factor of two every two years, while the memory access time and bandwidth have scaled at a substantially slower pace. This phenomenon is sometimes called the *memory wall*. Additionally, the energy of modern computers is dominated by memory accesses and data transfer, as it is substantially higher than processing operations in CPU.

This paper show how emerging technologies enable development of novel non-von Neumann architectures that alleviate the von Neumann bottleneck. These emerging non-volatile memory technologies include RRAM, PCM, and STT-MRAM among others (for simplicity, we refer to all emerging non-volatile technologies that are based on resistance to store data as *memristors*) and are considered as attractive candidates to replace conventional memory technologies (*e.g.,* DRAM and Flash) due to their speed, low power, scalability, and high endurance [1]. Memristive technologies have also been explored for additional applications such as analog circuits, neuromorphic computing, and logic circuits.

For the proposed non-von Neumann architecture, memristive memory is used also to perform logical operations. Several logic families using the structure of a memristor-based memory have been proposed, thus enabling a unit that can perform both memory and processing within the same unit. We show such a unit called *Memory Processing Unit* (MPU) that can be dynamically changed from data processing to storage. MPU maintains the standard structure of the memory cells and the array. Thus the use of MPU is compatible with standard von Neumann architecture and existing operating systems as it can function as a standard memory or as a parallel processing element and the new capabilities of the memory are encapsulated by the memory controller.

II. STATEFUL LOGIC

One approach for memristor-based logic within a memory architecture is to treat resistance solely as the logical state, exactly the same way as storing data within a memristive memory. For this approach, the memristors are the primary building blocks of the logic gate. Memristors act as an input, output, computational element, and latch in different stages of the computing process. This approach is called *stateful logic* and is suitable for array architectures and can therefore be integrated within a standard memristor-based memory, adding computing capabilities to the memory without changing its regular functionality. Several stateful logic families have been proposed [2-4] based on application of different voltages across the rows and columns of the memory array. The applied voltages write the result to an output memristor based on the stored values in the input memristor.

An improved stateful logic family is Memristor Aided Logic (MAGIC) [5]. In MAGIC, only a single applied voltage is used to perform a NOR logic operation, while the basic principles of stateful logic are maintained. The basic schematic of a MAGIC gate within a memristive crossbar array is shown in Figure 1. Stateful logic provides an opportunity to explore non-von Neumann architectures, where the memory can perform logic operations on the same devices that store data.

III. MEMORY PROCESSING UNIT (MPU) ARCHITECTURE

A. MPU - General Structure

MPU (Memory Processing Unit) is a non-von Neumann architecture where the memory has inherent and independent processing capabilities. In MPU, retained data within the memory act as the input of the logical operations and the result of the operation is immediately stored to the memory cells without need to transfer data out of the memory array. The structure of the logic gates and memory cells is identical and the decision as whether an element is a data storage element or a processing element is done dynamically by the memory controller, according to the executed program.

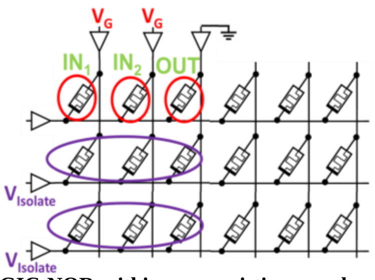

Figure 1: MAGIC NOR within a memristive crossbar is executed in the first row, where the data initially stored within memristors IN_1 and IN_2 is the input of the gate and the final state of memristor OUT is the result of the operation. Other rows are unselected.

While conventional memory technologies (*e.g.*, DRAM and SRAM) cannot be used for MPU, most memristive memory cells can also act as processing elements and therefore enable the proposed MPU architecture. The memristive memory architecture can be either a crossbar array structure [6] or include selectors in each memory cell [7]. Since this memory can also act as conventional memory in a von Neumann machine, MPU is compatible with standard computers and can have the functionality of either a conventional memory or hybrid memory-computing engine. The unique new capabilities of the memory are encapsulated by the memory controller and therefore the operating system does not change. The proposed MPU architecture is illustrated in Figure 2.

To achieve compatibility with standard von Neumann architecture, the instruction set architecture (ISA) of a standard computer (*e.g.*, X86, ARM) needs to be extended with instructions for logical and arithmetic operations within memory. These instructions are used to perform a specific computing task on known locations in the memory (*i.e.*, addresses). The extended ISA is mostly based on vector operations since they benefit most from logic within memory.

B. Controller - Memory and CPU Interface

In addition to the extended ISA, a new interface protocol between the CPU and the memristive memory is required. The memory controller which resides in the CPU sends commands to the memory through a dedicated controller, which controls both conventional read and write instructions as well as computational operations. To perform a certain instruction within the memristive memory, the controller breaks the instruction into micro-operations which are pipelined to the memory, thus maximize the processing efficiency. These micro-operations are built on several levels of abstraction where the lower level of abstraction is the basic logical operation (i.e., NOR operation in MAGIC). For example, a bitwise XOR operation between two 32-bit binary vectors from known addresses is divided into 38 write and NOR operations, whereas an instruction for adding two 32-bit numbers is broken into 385 operations [8]. We have developed different optimization algorithms to optimize the performance, energy, and area for different instructions. Note that there is a tradeoff between the complexity of the controller and the optimality of the algorithm.

C. MPU Contribution

The benefits from the MPU architecture come from the elimination of data transfer to/from the processor since the logic operations can be executed simultaneously on multiple rows within the memory (possibly all rows in the memory

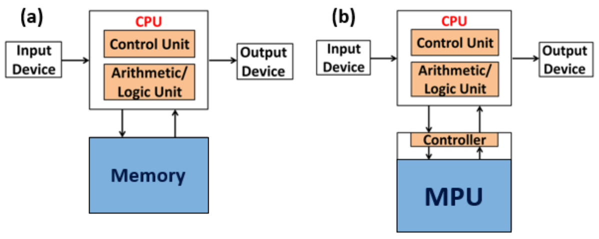

Figure 2: (a) von Neumann architecture consists of separate processing (CPU) and storage (memory) units. (b) The proposed architecture, where the memory is an MPU that also executes logical operations. Programs (or a fraction of them) are executed within the memory, without transferring data back and forth to the CPU.

array if desired). The potential drawbacks of this processing paradigm are due to the nature of the computation that relies on sequential execution of simple logical operations. In each computational step, only a single operation is executed (per row) and for certain applications, where there is low data-level parallelism, the latency of the execution can be longer than going to an off-chip processor. For example, a 32-bit addition takes 385 steps for MAGIC-based MPU [8].

IV. Conclusions and Future Work

MPU, a novel architecture for processing within a memristive memory is presented in this paper. MPU has a great potential for orders of magnitude improvement in both performance and energy efficiency for many types of applications [9], due to the significant reduction in data transfer and the massive parallel processing capability. Several implications of this new framework should be further researched. It is required to adjust the proposed scheme to various data-intensive programs that can benefit from it. Additionally, more algorithms for in-memory logic should be developed, while existing algorithms should be optimized in terms of processing time and power consumption.

V. Acknowledgements

This research was partially supported by Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI) and by the Viterbi Fellowship in the Technion Computer Engineering Center.

VI. References

[1] S. Kvatinsky *et al.*, , "The Desired Memristor for Circuit Designers," *IEEE CAS Magazine*, Vol. 13, No. 2, pp. 17-22, second quarter 2013.

[2] S. Kvatinsky *et al.*, "Memristor-based IMPLY Logic Design Flow," *ICCD 2011*, pp.142-147, October 2011.

[3] E. Lehtonen *et al.*, "Two Memristors Suffice to Compute All Boolean Functions," *Electronics Letters*, Vol. 46, No. 3, pp. 239-240, February 2010.

[4] E. Lehtonen *et al.*, "Recursive Algorithms in Memristive Logic Arrays," *IEEE JESTCS*, Vol. 5, No. 2, pp. 279-292, June 2015.

[5] S. Kvatinsky *et al.*, "MAGIC – Memristor Aided LoGIC," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 61, No. 11, pp. 895-899, November 2014.

[6] Z. Jiang *et al.*, "Analysis and Predication on Resistive Random Access Memory (RRAM) 1S1R Array," *IMW 2015*, pp. 1-4, May 2015.

[7] S.-S. Sheu *et al.*, "A 5ns Fast Write Multi-Level Non-Volatile 1K Bits RRAM Memory with Advance Write Scheme," *Proceedings of the Symposium on VLSI Circuits*, pp.82-83, June 2009.

[8] N. Talati *et al.*, "Logic Design within Memristive Memories Using Memristor Aided loGIC (MAGIC)," *IEEE Transactions on Nanotechnology* (submitted).

[9] S. Hamdioui *et al.*, "Memristor Based Computation-in-Memory Architecture for Data-Intensive Applications," *DATE*, pp. 1718-1725, March 2015.