Improving the Compatibility and Manufacturability of Digital Architectures for Processing Using Resistive Memory*

Minh S. Q. Truong† Liting Shen† Alexander Glass† Alison Hoffmann†
L. Richard Carley† James A. Bain† Saugata Ghose‡

†Carnegie Mellon University ‡University of Illinois Urbana-Champaign

Many modern applications require frequent data movement between the CPU and the main memory, which consumes a significant amount of energy. To mitigate the cost of data movement, recent works propose a new computing paradigm called *processing-using-memory* (PUM), which leverages electrical interactions between memory cells to realize useful computation within the memory arrays, *without* the need for additional CMOS logic. Compared to adding discrete logic *near* memory arrays, PUM eliminates significantly more data movement and can enable thousands to millions of ways of data-level parallelism [3, 10]. A PUM-capable memory cell can be used as either (1) a multi-bit device capable of analog operations such as multiplication (e.g., [1,2]), or (2) a single-bit device capable of digital operations such as Boolean algebra (e.g., [7, 9, 13]).

While PUM can be implemented using both conventional (e.g., DRAM, SRAM) and emerging (e.g., MRAM, PCM, ReRAM)[1] memories, prior works build PUM architectures around a specific memory technology. Unfortunately, this creates a significant barrier to the development of PUM-based systems. Due to DRAM scaling issues [8, 11], manufacturers are hesitant to alter the design of DRAM arrays in fear of hurting yield, and while SRAM-based PUM avoids such concerns, it significantly compromises storage density compared to other memory technologies. Emerging memory technologies offer the promise of high-density PUM, but issues remain with the at-scale production of each of them. Given the uncertainty of which technology will emerge as the dominant replacement for DRAM, it becomes unclear which of the previously-proposed PUM architectures will be viable, which in turn prevents systems developers from solving software-level challenges holding back the adoption of PUM [3].

In our JETCAS 2022 paper, we aim to decouple PUM architectures as much as possible from (1) a specific memory technology, as well as (2) a specific logic family (e.g., the MAGIC logic family [6] provides driving voltages and circuit-level support for either NOR or NAND, while the FELIX logic family [4] provides this for NOR, NAND, and NOT in the same array). For this work, we build upon our prior work on RACER [13], a highly-scalable PUM architecture that efficiently enables MAGIC-NOR-based processing inside small memory arrays. We start by developing *interface circuits* that allow the RACER architecture to be compatible with *any* PUM logic family. These circuits ensure that most of RACER's circuitry can stay the same as we adapt the architecture to new logic families and to other resistive memory technologies (including both 1S1R-based crossbars and 1T1R-based arrays).

Building upon our technology-agnostic version of RACER, we seek to design a new logic family that is practical for the types of memory devices that we can manufacture today. Unfortunately, the MAGIC and FELIX proposals require constraints for the device switching thresholds that are difficult to achieve with today's resistive memory technology prototypes (as was also recognized by prior work [5, 14]). We propose a new logic family, OSCAR, that enables NOR and OR using resistive memories with significantly-relaxed constraints, which are widely compatible with existing device technologies.

Background: RACER Architecture. RACER [13] uses *bit pipelining* to achieve high performance with small $n \times n$ ReRAM *tiles* (e.g., $n = 64$). A w-bit word is stored across w tiles as shown in Figure 1. RACER's bit-serial operations iteratively apply Boolean logic operations one tile at a time. To pass intermediate values (e.g., carry-out bits) between tiles, RACER uses a resistive-memory-based *buffer* that connects to a tile using programmable pass gates. This allows tiles to act as per-bit pipeline stages, while buffers act as pipeline registers between stages, enabling the concurrent computation of $w \times n$ bit-serial operations. Using MAGIC-based NOR operations in ReRAM arrays, RACER achieves a 107× speedup and 189× energy savings compared to a modern 16-core Xeon CPU for microkernels from a range of important domains (image processing, linear algebra, signal processing, neural-network-based classification, string matching).

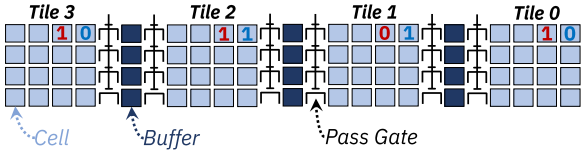

Figure 1: Tile and buffer design, showing two four-bit values (1101 in red, 0110 in blue) striped across the tiles.

Technology Interface Circuits. Figure 2a illustrates the necessary circuitry to enable bit-pipelining. The *pipeline controller* consists of one *micro-op queue* per tile. Each queue holds a sequence of micro-ops, which are commands that tell RACER which Boolean primitive to apply to a set of columns in the tile. RACER originally directly connected the pipeline controller to the tiles, and translated each micro-op to *predetermined* voltage signals according to the ReRAM-based MAGIC [6] logic family. Thus, RACER would need a significant redesign to integrate with a different logic family/technology. We address this inflexibility by decoupling the control circuitry from the tiles using modified *decode & drive* units. These units act as interface circuits that allow RACER to work with other logic families, by isolating the technology-agnostic aspects of the controller away from the technology-specific switching voltages.

We design the decode & drive units by observing that for all existing resistive PUM logic families (e.g., [4, 6, 12]), one of three voltages is asserted to each column of a tile depending on its "role": (1) one or two columns that serve as inputs of the Boolean operation are asserted with V_{in}; (2) one column that serves as the output is asserted with V_{out}; and (3) all other (i.e., idle) columns are asserted with V_{float}. The values of these assertion voltages depend on the specific logic family and on the Boolean operation (e.g., NOR, NAND). Once a decode & drive unit (Figure 2b) receives a micro-op (❶ in the

*__Full Paper:__ M. S. Q. Truong et al., "Adapting the RACER Architecture to Integrate Improved In-ReRAM Logic Primitives", JETCAS, June 2022, https://ghose.cs.illinois.edu/papers/22jetcas_oscar.pdf

[1]We use the term *resistive memory* to refer broadly to resistance-based non-volatile memories (e.g., PCM, MRAM, ReRAM), while we use ReRAM to refer specifically to oxide-based switches (often referred to as memristors).

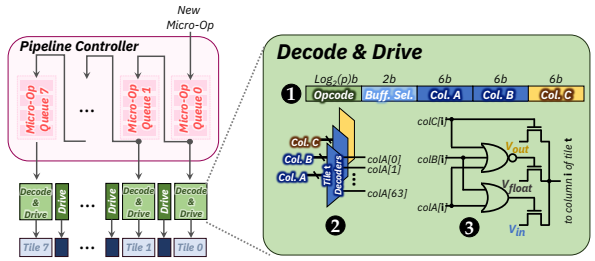

Figure 2: (a) CMOS circuits required to enable bit-pipelining; (b) the decode & drive interface circuit.

figure) from the micro-op queue, it can determine the assertion voltages using the micro-op's opcode field. Each column of the tile is then assigned a role based on the micro-op's operand address fields (❷). Once every column is assigned a role (i.e., input, output, or idle), the appropriate assertion voltages can be applied across the tile (❸). The decode & drive interface circuit enables the *parameterization* of different logic families (i.e., all logic families are treated in the same way from a control point of view). This effectively abstracts away any technology-dependent behavior from the control circuitry, and from any software building upon the architecture.

OSCAR Logic Family. With the ability to incorporate different logic families into RACER, we next explore limitations of existing logic families. One constraint for enabling logic in ReRAM is the ratio between V_{set} and V_{reset}, threshold voltages that when applied to a resistive device set it to a low-resistance or high-resistance state, respectively. Unfortunately, while the MAGIC [6] and FELIX [4] logic families require devices where $V_{set} < 2V_{reset}$, typical devices that can currently be manufactured instead require $V_{set} > 2V_{reset}$. Figure 3a illustrates the ratios between set, reset, and logic assertion voltages, showing how the constraints for MAGIC and FELIX do not cover the ratios achieved by today's real devices.

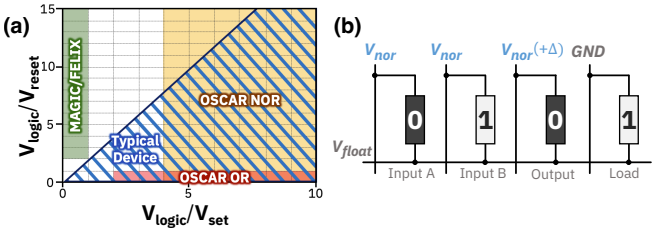

Figure 3: (a) Threshold voltage constraints of different logic families compared to typical devices; (b) voltage assertions on bipolar devices for OSCAR's NOR.

To address this incompatibility, we propose the OSCAR logic family, which can enable NOR and OR operations in ReRAM. For the non-destructive NOR primitive, we first preset the output cell to high resistance (i.e., logic 0). Then, we apply a voltage V_{nor} to the input cells, and $V_{nor}+\Delta$ to the output cell ($0 < \Delta < V_{set}$). We simultaneously ground a load cell set to low resistance (logic 1). This generates a current from the inputs and output to the load cell, resulting in a potential drop across the output. If one of the inputs has a low resistance (logic 1; shown in Figure 3b), the voltage drop across the output is negligible, and the output remains at logic 0. However, if both inputs are high resistance (logic 0), the voltage drop across the output is $V_{nor}/4$, and the output switches to logic 1 if $V_{nor} > 4V_{set}$. The addition of Δ ensures that the output switches before the inputs can, preventing the inputs from being destroyed. Our destructive OR primitive operates similarly to our NOR (see Section IV of the full paper). Unlike

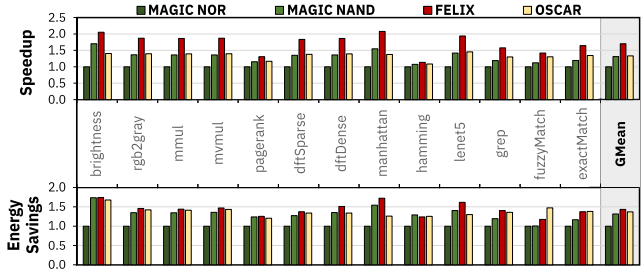

Figure 4: Speedup and energy savings of RACER for various logic families, normalized to MAGIC NOR.

existing logic families, OSCAR NOR does not require any constraint between V_{set} and V_{reset}, while OSCAR OR requires $V_{set} < 2V_{reset}$. As shown in Figure 3a, OSCAR's constraints are compatible with ratios achievable by typical devices.

With OSCAR, we improve upon RACER's performance and energy savings by 30% and 37%, respectively (Figure 4), compared to MAGIC NOR, while enabling broad compatibility with current devices. OSCAR-based RACER achieves a 142× speedup and 233× energy savings over a 16-core Xeon CPU. Notably, RACER now outperforms CASCADE [2], a state-of-the-art analog PUM dot product accelerator, for several matrix-based microkernels by an average of 3.16×. We conclude that by adapting RACER to other logic families, we can significantly improve the efficiency of Boolean PUM.

Significance. While we focus on improving the compatibility of RACER across a wide range of logic families and memory technologies, both our decode & drive circuit design and OSCAR can be adapted to other digital PUM architectures. This can allow others to design architectures that abstract away technology-specific details from the architecture and from software developers. We believe that this is a crucial step to enabling the development of the currently-missing software stack for non-dot-product PUM, which today is a significant barrier to widespread commercialization. Without this abstraction, there is little incentive to develop a toolchain for an architecture that may lose its relevance in a few years as device technologies and commercial processes evolve.

References

[1] P. Chi *et al.*, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *ISCA*, 2016.

[2] T. Chou *et al.*, "CASCADE: Connecting RRAMs to Extend Analog Dataflow in an End-to-End In-Memory Processing Paradigm," in *MICRO*, 2019.

[3] S. Ghose *et al.*, "Processing-in-Memory: A Workload-Driven Perspective," *IBM JRD*, Nov.–Dec. 2019.

[4] S. Gupta, M. Imani, and T. Rosing, "FELIX: Fast and Energy-Efficient Logic in Memory," in *ICCAD*, 2018.

[5] B. Hoffer *et al.*, "Experimental Demonstration of Memristor-Aided Logic (MAGIC) Using Valence Change Memory (VCM)," *TED*, Sep. 2020.

[6] S. Kvantinsky *et al.*, "MAGIC: Memristor-Aided Logic," *TCAS II*, Sep. 2014.

[7] S. Li *et al.*, "Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-Volatile Memories," in *DAC*, 2016.

[8] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.

[9] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.

[10] V. Seshadri and O. Mutlu, "Simple Operations in Memory to Reduce Data Movement," in *Advances in Computers*, 2017, vol. 106.

[11] S. Shiratake, "Scaling and Performance Challenges of Future DRAM," in *IMW*, 2020.

[12] S. Shirinzadeh *et al.*, "Logic Synthesis for RRAM-Based In-Memory Computing," *TCAD*, Jul. 2018.

[13] M. S. Q. Truong *et al.*, "RACER: Bit-Pipelined Processing Using Resistive Memory," in *MICRO*, 2021.

[14] D. J. Wouters *et al.*, "Reliability of Computing-In-Memory Concepts Based on Memristive Arrays," in *IEDM*, 2022.