ArXrCiM: Architectural Exploration of Application-Specific Resonant SRAM Compute-in-Memory

Dhandeep Challagundla, *Graduate Student Member, IEEE*, Ignatius Bezzam, *Member, IEEE*, and Riadul Islam, *Senior Member, IEEE*

***Abstract*— While general-purpose computing follows von Neumann's architecture, the data movement between memory and processor elements dictates the processor's performance. The evolving compute-in-memory (CiM) paradigm tackles this issue by facilitating simultaneous processing and storage within static random-access memory (SRAM) elements. Numerous design decisions taken at different levels of hierarchy affect the figures of merit (FoMs) of SRAM, such as power, performance, area, and yield. The absence of a rapid assessment mechanism for the impact of changes at different hierarchy levels on global FoMs poses a challenge to accurately evaluating innovative SRAM designs. This article presents an automation tool designed to optimize the energy and latency of SRAM designs incorporating diverse implementation strategies for executing logic operations within the SRAM. The tool structure allows easy comparison across different array topologies and various design strategies to result in energy-efficient implementations. Our study involves a comprehensive comparison of over 6900+ distinct design implementation strategies for École Polytechnique Fédérale de Lausanne (EPFL) combinational benchmark circuits on the energy-recycling resonant CiM (rCiM) architecture designed using Taiwan Semiconductor Manufacturing Company (TSMC) 28-nm technology. When provided with a combinational circuit, the tool aims to generate an energy-efficient implementation strategy tailored to the specified input memory and latency constraints. The tool reduces 80.9% of energy consumption on average across all benchmarks while using the six-topology implementation compared with the baseline implementation of single-macro topology by considering the parallel processing capability of rCiM cache size ranging from 4 to 192 kB.**

***Index Terms*— Compute-in-memory (CiM), logic synthesis, memory bottleneck, resonant energy-recycling, static random-access memory (SRAM).**

I. Introduction

CACHE memory remains one of the critical components in our computing system, enhancing overall performance by bridging the speed gap between the main memory [random access memory (RAM)] and the central processing unit (CPU). In addition, in recent years, static random-access memory (SRAM)-based in-memory computing paved a promising direction to enable energy-efficient computation. However, the lack of design and automation tools to map computation on optimal SRAM architecture increases design time-to-market, resulting in higher engineering costs. This research resolves this issue by proposing an architectural exploration tool that efficiently maps logic computations to optimal cache architecture.

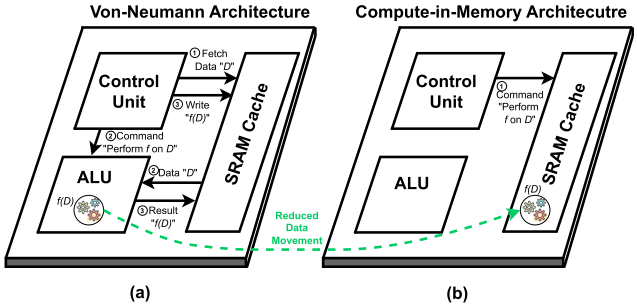

Fig. 1. (a) Conventional von Neumann architecture, where an operation f is performed on data D within the CPU, incurs high data movement overhead, which can be reduced using (b) CiM architecture, where f is computed directly within the memory, with the CPU primarily functioning as a control unit.

Computing-in-memory (CiM) architectures have emerged as highly promising solutions for data-intensive applications. They minimize data movement, enhance computational capabilities, and improve the system's overall energy efficiency by processing and storing data within cache memory. As shown in Fig. 1(a), the traditional von Neumann architecture relies on data communication between the arithmetic logic unit (ALU) and cache memory through address and data buses. However, as the CPU performance is significantly higher than the memory performance, the von Neumann architectures often create memory bottlenecks. CiM architectures, as shown in Fig. 1(b), mitigate the impact of large memory access latencies by performing the computations within the memory. By reducing the data movement and exploiting parallelism within the memory, CiM architectures significantly enhance computational efficiency and performance. SRAM-based CiM architectures have been heavily investigated for performing various operations, such as matrix–vector multiplication (MVM) [1],

Received 25 July 2024; revised 6 October 2024; accepted 9 November 2024. Date of publication 25 November 2024; date of current version 31 December 2024. This work was supported in part by the National Science Foundation (NSF) under Award 2138253, in part by Rezonent Inc. under Award CORP0061, and in part by the University of Maryland, Baltimore County (UMBC) Startup Fund. *(Corresponding author: Dhandeep Challagundla.)*

Dhandeep Challagundla and Riadul Islam are with the Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD 21250 USA (e-mail: vd58139@umbc.edu; riaduli@umbc.edu).

Ignatius Bezzam is with Rezonent Inc., Milpitas, CA 95035 USA (e-mail: i@rezonent.us).

Digital Object Identifier 10.1109/TVLSI.2024.3502359

[2], multiply-and-accumulate (MAC) operations [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], Boolean logic operations [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], and content-addressable memory (CAM) [31], [32], [33], [34], [35], [36] operations for fast searching operations. However, none presents a generic energy-saving architecture that spans across various applications. This work utilizes a novel series-resonance-based resonant CiM (rCiM) architecture that reduces dynamic power consumption by recycling the wasted energy during writing operations.

This work proposes an agile architectural exploration tool to map various logical operations to an optimal SRAM macro cache size. The primary objective of the tool is to facilitate the development of novel energy-efficient SRAM-based energy-recycling rCiM implementations individually designed for specific Boolean logical applications.

In particular, the main contributions of this article are as follows.

1) A novel resonant rCiM structure that incorporates a series inductor to recycle energy dissipated during write operations.
2) An architectural exploration toolflow that integrates open-source synthesis tools (Berkeley-ABC [37] and YOSYS [38]) to identify the optimal SRAM configuration within a specified range of SRAM cache memory and map efficient logical operations tailored to an optimal rCiM macro size.
3) Comprehensive analysis of 6900+ distinct logical design implementations for École Polytechnique Fédérale de Lausanne (EPFL) combinational benchmark circuits [39] using 12 different SRAM topologies.

II. Background

In recent years, considerable efforts have been dedicated to addressing the memory bottleneck associated with conventional von Neumann architectures by adopting CiM architectures. This paradigm can be implemented using both SRAM and nonvolatile memories (NVMs) [40], [41], [42], [43], [44], [45], [46], [47], [48]. While CiMs utilizing NVMs address static power concerns, they encounter high write energy and latency challenges. Conversely, SRAM-based CiM provides faster processing speed and robust scalability [49]. In a recent study, Malhotra et al. [42] propose a ferroelectric field effect transistor-based CiM technique designed for executing a single two-operand Boolean function with a single-memory access. A different study [19] achieves the implementation of an arbitrary Boolean function using SRAM-based CiM. This work focuses on performing a whole combinational logic, which is crucial for SRAM-based CiMs to reduce the frequency of memory fetch operations. The diverse logical representations utilized for these combinational logic operations significantly influence the latency and overall performance of CiM architectures.

Logic synthesis takes a register transfer-level (RTL) implementation, typically in Verilog or VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (VHDL), and generates a gate-level representation of the design using a standard cell library. This work uses YOSYS synthesizer [38] and ABC logic synthesizer [37] to perform the RTL synthesis. The ABC takes Verilog input, and using a *"strash"* function converts the input RTL into an and-inverter-graph (AIG) graph represented as a directed acyclic graph (DAG). This AIG graph allows for structural optimizations to be performed [50]. This work uses four fundamental subgraph optimizations supported by ABC, namely, *"Refactor* (R_f)," "*Rewrite* (R_w)," *"Resubstitution* (R_s)," and *"Balance* (B_a)." The R_f optimization technique performs iterative collapsing and refactoring logic nodes in the AIG, aiming to reduce the AIG nodes and logic levels. Similarly, R_w performs DAG-aware rewriting of the AIG network to reduce the number of logic levels. These options are significant for CiM applications, as the proposed rCiM implementation aims to perform a single level of the design hierarchy within one computational cycle. The optimization with R_s is achieved by representing the logical function of a node using the existing nodes. A unique combination of these subgraph optimizations will yield distinctive AIG implementations—the proposed algorithm in Section III-D leverages these AIG implementations to map combinational workloads efficiently onto the rCiM architecture with diverse topologies.

In addition, the innovative rCiM implementation employs a write driver based on series resonance and supply boosting, adopted from [51], [52], [53], [54], [55], [56], and [57], to significantly lower the dynamic power consumption when writing back the computational outputs. In a conventional CiM architecture, whenever a bitline discharges from a "1" to "0," energy gets dissipated through heat. Series LC resonance utilizes an on-chip inductor placed in the discharge path of the bitlines to store this dissipated energy and harvest it immediately into the design.

Fig. 2(a) illustrates a conventional SRAM write driver used to write data onto the SRAM using bitlines. Whenever the input data is "0," the corresponding bitline (BL) is driven from precharged value (VDD) to ground potential using the driver inverter. The resonant write driver, shown in Fig. 2(b), employs an inductor "L" to store this discharged energy. During the precharge phase, this energy is recycled back into the corresponding bitlines, the bitline (BL) or the complementary bitline (BLB) [51], [58]. At the start of the write operation, the "vsr" signal is turned "on," enabling the inductor to store the energy discharged from bitline. Subsequently, the "vdn" signal is turned "on" to ground the bitline fully. Once the write operation concludes and the precharge phase begins, the "vsr" signal is reasserted to recycle the stored energy onto the bitline.

Designing SRAM in scaled technologies necessitates a deep understanding of process variations, circuit dynamics, and architectural considerations. While technology scaling has facilitated the development of ever-larger cache memories, persistent challenges emerge from scaling issues. Open-source tools such as OpenRAM [59] and virtual prototyper (VIPRO) [60] contribute significantly by providing essential capabilities for estimating and generating SRAM architectures but do not apply to CiM architectures as they only generate SRAM memories for read and write operations and porting for another technology is nontrivial. Recently, researchers

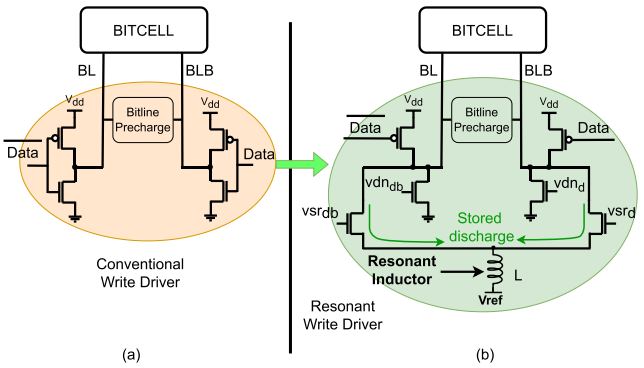

Fig. 2. (a) Conventional SRAM write driver exhibits high dynamic power consumption due to large bitline capacitance and (b) resonant write driver recycles this dynamic power using an inductor "L" placed in the discharge path, along with timed "vsr" and "vdn" signals.

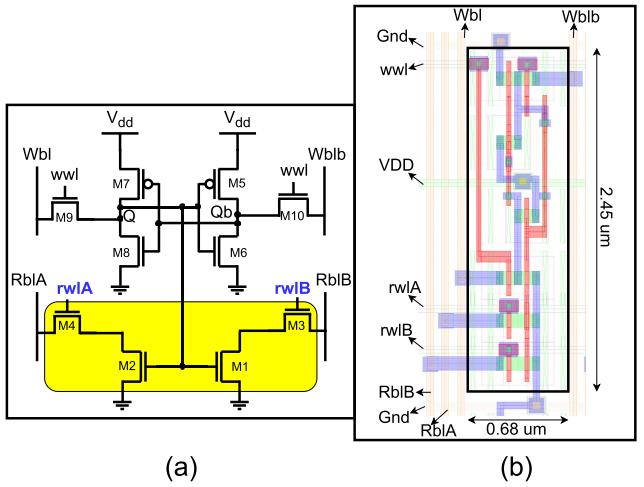

Fig. 3. (a) Schematic of the proposed 10T SRAM cell and (b) corresponding layout of the bitcell using up to M3 metal layers for horizontal wordlines and vertical bitlines with the area of the bitcell is 1.66 μm^2.

developed OpenSAR [61], a tool to design successive approximation register analog-to-digital converter (SAR ADC)-based analog building blocks, such as comparators and sample-and-hold circuits. Another noteworthy development is AutoD-CIM [62], a tool designed to generate CiM macros. These emerging tools inspire the development of an innovative architectural exploration tool that adeptly maps various logical optimizations, ensuring optimal utilization of SRAM cache architectures.

III. Proposed Methodology

The CiM architecture integrates a conventional SRAM cache, enabling additional computations within the same macro. This section presents a new energy-efficient CiM architecture specifically designed for performing Boolean logic operations. In this article, we proposed a novel methodology for selecting the optimal cache architecture, resulting in energy-efficient implementation tailored to a specific application.

A. Proposed 10-Transistor Cell

Fig. 3(a) shows the schematic of the proposed 10-Transistor (10T) SRAM cell, which builds upon a standard 6T cell

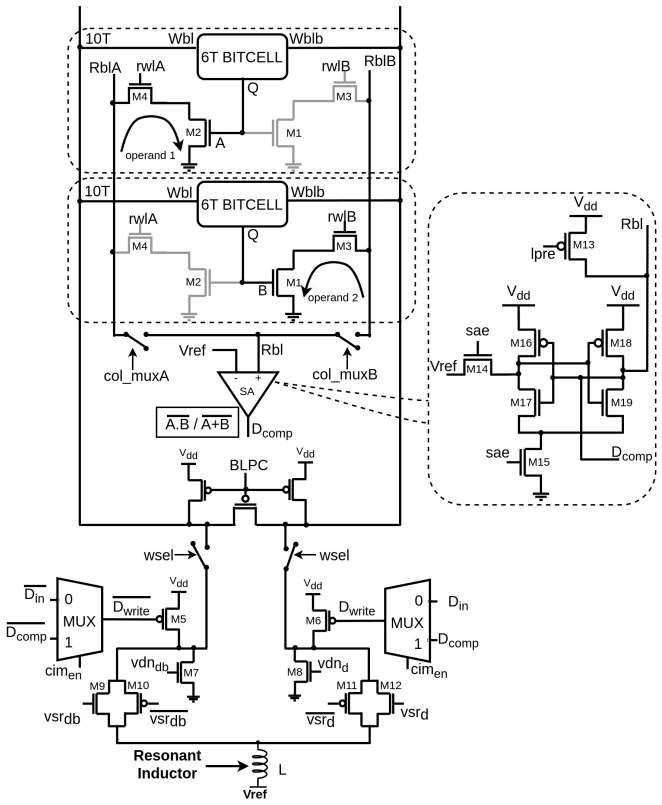

Fig. 4. 10T-SRAM bitcell along with resonant write driver implementing a single logical operation using the output from the SA and writeback using an energy-recycling resonant write driver.

architecture by incorporating four additional transistors ($M1$–$M4$). These extra transistors form a dedicated dual read-port, enhancing the cell's capability for single-bit logic operations. Fig. 3(b) illustrates the layout implementation of this 10T cell schematic. This layout occupies an area of 1.66 μm^2 and utilizes multiple fabrication layers, including $mpoly$ and metals. Specifically, the horizontal wordlines are routed using the M2 metal layer, and the vertical bitlines are constructed using the M3 metal layer.

B. rCiM Architecture

Fig. 4 shows the working principle of rCiM architecture. The rCiM performs Boolean logic using two 10T bit cells, as shown in Fig. 4. The transistors $M1$–$M4$ form a decoupled dual-read port, which allows for a large voltage swing during the conventional read operation and alleviates potential read disturb failures. Dedicated dual-read ports allow individual access to each vector operand, eliminating unidirectional computation restrictions in the SRAM array. This capability improves data retrieval efficiency, leading to enhanced system functionality and performance.

To execute a NAND2/NOR2 operation, we start by decoding the input operand addresses and simultaneously enabling the corresponding read wordlines (rwlA and rwlB). The initially precharged read bitlines, RblA and RblB, will eventually discharge to 0 V if either of the operands corresponds to a "1." The discharge rate of RblA/RblB is dependent on whether the one-bit cell is storing a "1" or if both the bit cells are storing a "1." The pulse widths of read word-

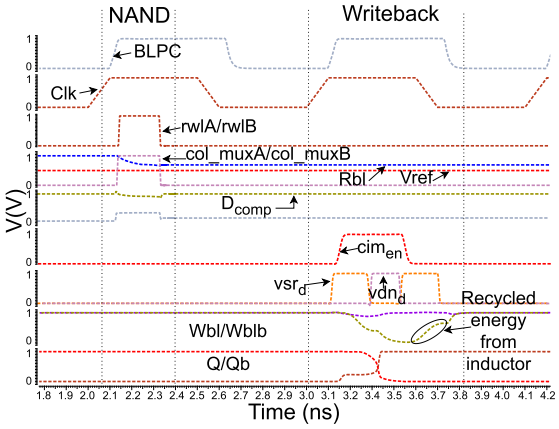

Fig. 5. SPICE simulation confirms the correct in-memory computation considering logical NAND2 operations with "01/10" data and a conventional energy-recycling writeback operation.

lines are adjusted to leverage this varying discharge rate to ensure that the RblA/RblB does not completely discharge for cases"10/01" during a NAND2 operation. For a NOR2 operation, enabling the rwlA/rwlB for a higher time allows the read bitlines to be driven to 0 V for cases"10/01," outputting a "0." A programmable buffer-based pulse generator circuit is integrated with the system clock to generate the necessary rwlA/rwlB pulses for performing a NAND2 or NOR2 operation. The discharge time for the NAND2 operation is approximately 150 ps, while the NOR2 operation has a discharge time of around 350 ps. The notable difference in discharge times contributes to the observed voltage difference between NAND2 ("01/00") and NOR2 ("01/00") operations, allowing for reliable distinction between these logic states. The rCiM architecture operates under a global clock frequency of 1 GHz, with all operations triggered on the rising edge of the clock. The pulsewidths required for the discharge operations generated using the programmable buffer are based on the rising edges of the clock signal and a delayed clock signal. This approach ensures that the pulsewidth remains constant at any lower frequencies below 1 GHz, as the delay introduced by the buffer does not change.

Fig. 5 shows the transient simulation of performing a single NAND2 operation for cases "10/01." The read bitlins, RblA and RblB, are connected to one end of a single-ended sense amplifier (SA) through the column mux switches (col_muxA and col_muxB), as shown in Fig. 4. The SA is formed using the transistors $M13$–$M19$, adapted from [63]. The other end of the SA is connected to a reference voltage (Vref) which is lower than the discharge of RblA/RblB during a NAND2 operation for cases "10/01," as shown in Fig. 5. Thus, the output of the SA (D_{comp}) will result in the output imitating a NAND2 operation by resulting a logical "1" for all three cases ("00" and "10/01"). The Vref signal is positioned at VDD/2 and the rwlA/rwlB pulsewidths are characterized such that the Rbl discharge is greater than the $Vref$ voltage during the NAND2 "10/01" cases. While performing a NOR2 operation, the SA output produced a logical "0" for all three cases ("11" and "10/01"). When a single vector operand is applied to both rwlA and rwlB, the operation only considers two different cases ("00" and "11"). Thus, performing a NAND2 operation with a single vector operand results in an inversion, effectively performing a NOT operation.

The D_{comp} output is latched and utilized as input data (D_{write}) to be written in the subsequent clock cycle by a resonant write driver. During a conventional write operation, the multiplexer selects the CPU data input (D_{in}). While performing CiM computations, the cim_{en} signal goes high, selecting the D_{comp} signal to be written into a bit cell, as shown in Fig. 5. The energy-recycling write driver and supply boosting, which is adapted from [51] and [52], uses a series resonant inductor to recycle the dissipated energy from write bitlines (Wbl/Wblb) during write operation and the precharge phase. The resonant inductor is connected to Wbl/Wblb on one end, and a reference voltage ($Vref$) on the other end. To maximize the savings from the resonant inductor, the Vref value is chosen to be (VDD/2). Whenever Wbl/Wblb transitions from a logic "1" to a logic "0," the energy dissipated is stored in the Vref node. During the precharge phase, this stored charge is emptied from the Vref node, resulting in zero net currents for the whole cycle.

The resonant write driver circuit transistors $M9$–$M12$ shown in Fig. 4, enable resonance by conditionally connecting the Wbl/Wblb to the inductor controlled by vsr_d and vsr_{db} signals derived from the system clock. Depending upon the data, either Wblb is discharged, if the input data is "1," or Wbl is discharged. For the case shown in Fig. 5, vsr_d signal is enabled to discharge the Wblb signal for writing the NAND2 output of "1" for input case "01/10." The vdn_d and vdn_{db} signals ensure full voltage swing by completely discharging one of the write bitlines. After a successful write operation, the same transmission gates ($M11$–$M12$) as before are enabled to recycle the stored energy from the inductor. Hence, when the active-low bitline precharge (BLPC) signal is activated, there is no need to precharge the write bitlines from "0," resulting in a decrease in the overall power consumption. The product of the bitline capacitance and the resonant inductor remains constant for a given resonant frequency.

Utilizing a shared inductor for all the write drivers significantly minimizes the inductor's size as the bitline capacitance increases N times for N write drivers.

C. Overall Architecture of rCiM Topologies

Fig. 6 illustrates various SRAM topologies for implementing rCiM architecture. The overall architecture of rCiM includes a 10T SRAM array, a readout circuit using single-ended SA's, two-row decoders enabling concurrent operands access, energy-recycling write drivers for low-power writing operations, and a central control block responsible for generating internal signals.

When considering the memory size for rCiM implementation, one can choose between a single large SRAM macro, as shown in Fig. 6(a) or multiple smaller SRAM macros, as shown in Fig. 6(b). The latter allows parallel execution of various logical operations, which proves beneficial for smaller designs with fewer operations in each stage, resulting in enhanced performance. However, the optimal approach for larger designs is yet to be determined—whether to increase the number of operations per stage or divide them for minimal

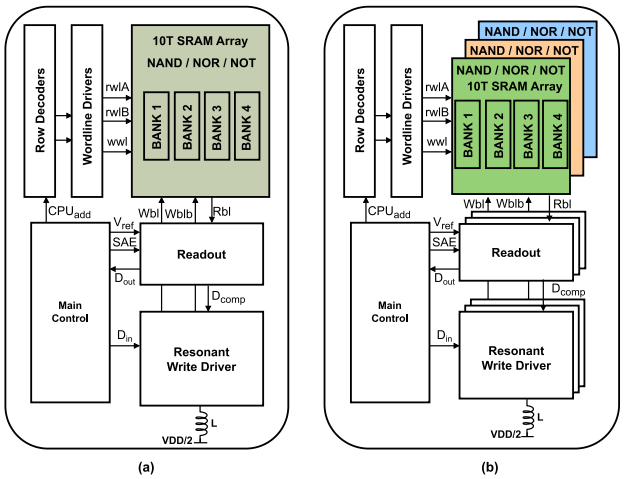

Fig. 6. Comparison of memory topology considerations for rCiM architecture, showcasing (a) single large SRAM macro or (b) multiple smaller SRAM macros.

energy consumption. The analysis in Section IV-B explains this particular aspect.

This design assigns one SA for each pair of columns in the bit cell array, facilitating the execution of both conventional read operations and efficient computational processes. Consequently, the resulting architecture exhibits the capability of executing $(M/2)$ logic operations of the same kind for an SRAM bank column size of M. For example, a 2-kB SRAM bank with 128×128 SRAM bit cells can perform 64 logical operations in a single computational cycle.

Within each SRAM macro, there are several SRAM banks. By activating only one selected SRAM bank, the remaining SRAM banks enter a standby mode, resulting in a reduction in overall macro power usage. Significant dynamic power consumption in SRAM emanates from bitline charging and discharging as well as enabling wordlines. The use of multiple banks significantly contributes to the lowering of bitline power consumption.

The rCiM can be designed using two architectural configurations as depicted in Fig. 6. The SRAM topology showcased in Fig. 6(a) utilizes a single macro, restricting the system to perform only one type of logical operation in a computational cycle. This architecture is particularly advantageous for scenarios with fewer logic levels but more operations within each level. Increasing the column count enables a greater number of parallel operations within a single bank, reducing the latency of the logical operation. Fig. 6(b) demonstrates the use of multiple SRAM macros in the rCiM. In this topology, each SRAM macro can execute a distinct logical operation. For instance, using three macros allows for the concurrent execution of NAND2, NOR2, and NOT logic operations, with each macro dedicated to one operation. This article proposes an algorithm in Section III-D designed to choose an optimal topology from the available SRAM macro banks.

D. Proposed Combinational Logic Operation Mapping Methodology

Fig. 7 presents two AIGs for the same 2-bit adder Verilog circuit, each generated using the ABC tool [37] with different synthesis recipe options. These AIGs are used in YOSYS to generate netlists, which are crucial for simulating CiM designs. The variations in synthesis options result in AIGs with different levels and gate counts, significantly influencing the implementation's latency and performance in CiM.

Fig. 7(a) shows an AIG with eight levels, each level represented by a distinct color. Although it has fewer gates compared with Fig. 7(b), the higher number of levels implies greater latency when implemented in a CiM system, as each level requires one clock cycle for execution. In contrast, Fig. 7(b) displays a more complex AIG in terms of gate count but with only six levels. Despite its complexity, the lower number of levels enables faster execution in CiM due to reduced clock cycles required for processing.

These diagrams effectively illustrate how different synthesis recipes affect the structure of AIGs, impacting the number of levels and the performance characteristics of the CiM systems. Thus, the choice of synthesis recipe becomes a crucial factor in optimizing computational efficiency and speed in CiM applications. Fig. 7(a) illustrates the mapping strategy for a single macro implementation, while (b) shows the mapping strategy using a three-macro implementation. The AIG graphs are mapped in the single macro approach by assigning each logic level to a specific row or column in the SRAM array. The first level of the AIG is mapped to the first row, with its outputs stored in the second row. This pattern continues, with each level of the AIG occupying a new row and the corresponding outputs stored in subsequent rows until all AIG levels have been processed. The algorithm selects the SRAM size to ensure it can accommodate all required inputs and outputs based on the total number of gates in the design. In the three-macro implementation, the logic levels are distributed across the three macros. Each level of logic operations is divided, sorted, and assigned to a specific macro, with operands grouped accordingly. The mapping strategy then places each logic level across the SRAM rows. By aligning the data and operation execution across multiple macros, the architecture effectively manages resource constraints and maximizes throughput. If a row becomes full, the 10T bitcell allows for operands to be stored across columns as well. Since the architecture shares SAs between two columns, operands can be placed flexibly within the two columns, not strictly confined to a single row or column. This flexibility enhances the architecture's ability to store and manage operands across multiple columns, optimizing the use of available SRAM resources.

To enable energy-efficient in-memory computation, we propose an algorithm that maps combinational logic workloads to optimal resonant cache architecture, as shown in Algorithm 1. The algorithm takes as input the RTL netlist (i.e., Verilog/VHDL/SystemVerilog) of the design, AIG synthesis options ($\text{AIGsyn}_{\text{opt}}$), and the list of available SRAM topologies (SRAM_{list}). The algorithm's output is an optimal energy-efficient rCiM architecture.

The algorithm starts with generating unique (AIG_{list}) using the AIG synthesis transformations ($\text{AIGsyn}_{\text{opt}}$) and the given RTL netlist as indicated in Line 3. The open-source synthesizer ABC is used to create unique AIGs using subgraph optimizations: B_a, R_f, R_w, and R_s [37]. The number of

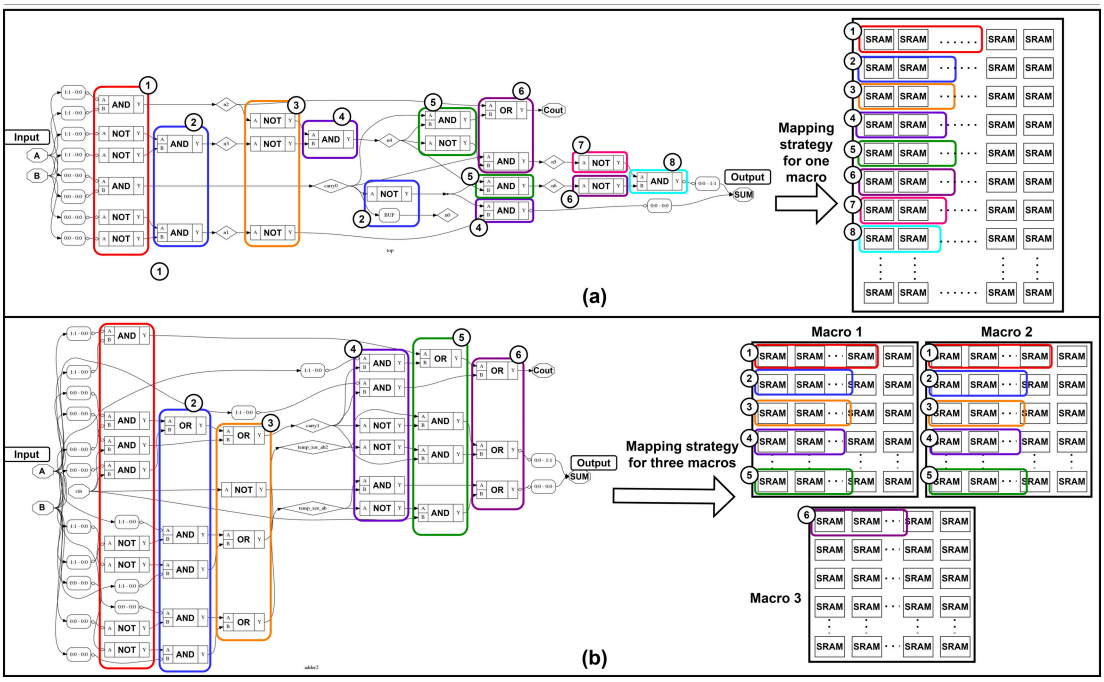

Fig. 7. AIG graph generated using different synthesis transformations results in AIGs with different levels and different numbers of gates at each level along with mapping strategies. (a) Example AIG with eight levels mapped onto a single macro SRAM. (b) Example AIG with six levels mapped onto a three-macro SRAM implementation.

unique AIG synthesis transformations generated from S different subgraph optimizations is expressed by $\sum_{i=1}^{S} {}^{S}P_i$. For instance, considering $S = 3$ where the provided subgraph optimizations are B_a, R_f, and R_w, would result in 15 unique subgraph optimizations, such as $\{(B_a); (R_f); (R_w)\}$, $\{(B_a, R_f); (B_a, R_w); (R_f, B_a); (R_f, R_w); (R_w, B_a); (R_w, R_f)\}$ and, $\{(B_a, R_f, R_w); (B_a, R_w, R_f); (R_f, B_a, R_w); (R_f, R_w, B_a); (R_w, B_a, R_f); (R_w, R_f, B_a)\}$. This work uses four subgraph optimizations, resulting in 64 unique AIG synthesis transformations.

When presented with an input RTL, the ABC tool initially constructs an AIG represented as a DAG. This DAG serves as the foundation for the subgraph optimizations performing tree-balancing transformations, logic rewriting, and node reduction, which results in minimizing the delay of the design and improving logic sharing.

The flowchart in Fig. 8 visually represents the proposed methodology described in Algorithm 1, starting with generating gate-level netlists using YOSYS and synthesis transformations using ABC. The number of gates and hierarchy levels then characterizes each AIG. These AIGs are sorted to identify those with optimal gate and logic levels. Subsequently, a set of SRAM topologies are determined based on gate counts and design cycles. The identified SRAM range is then evaluated for power, latency, and energy consumption metrics. Finally, the optimal SRAM topology is used to calculate the inductor size for the resonant inductor tuning, leading to the optimal rCiM architecture.

The For loop (Lines 4–6) iterates over every synthesized graph to characterize each AIG ($\text{ChaAIG}_{\text{list}}$). The characterization phase determines the number of stages in the design hierarchy and counts the number of logical operations at each

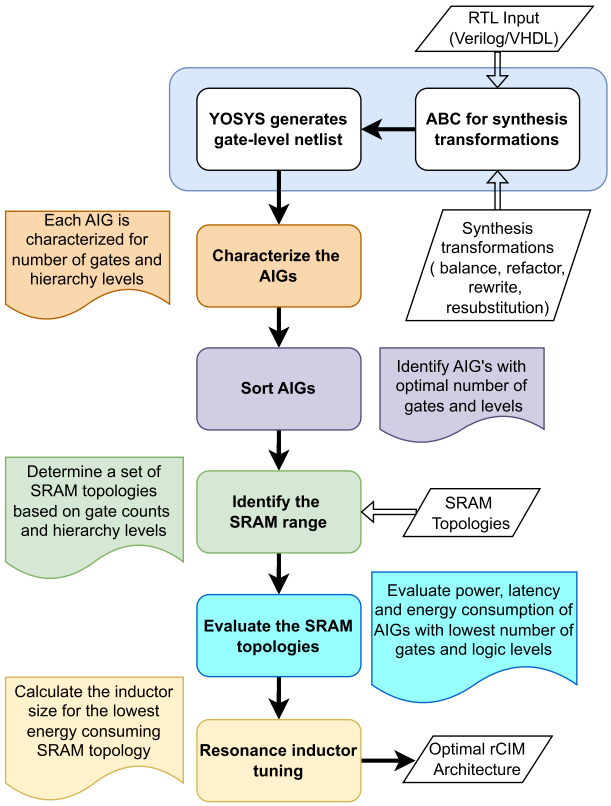

Fig. 8. Proposed methodology flowchart shows different operations in sequential order to determine the optimal SRAM topology for a given input RTL.

stage. Line 7 and Line 8 identify the AIGs with optimal gate count and minimum logic level count among all the synthesized AIGs, respectively. Line 9 is used to identify a

Algorithm 1 Mapping Combinational Logic Workloads to Optimal Resonant Cache Architecture

1: **Input:** RTL netlist (RTL), SRAM Toplogies ($SRAM_{list}$), AIG synthesis options ($AIGsyn_{opt}$);
2: **Output:** rCiM Architecture;
3: $AIG_{list} \leftarrow CreateAIG(RTL, AIGsyn_{opt})$; ▷ Create unique AIGs using different AIG synthesis options
4: **for all** AIG in AIG_{list} **do** ▷ Loop through each AIG
5: $ChaAIG_{list} \leftarrow ChaAIG(AIG)$; ▷ Count number of hierarchy/ logic levels and logical operations in each level of the AIG
6: **end for**
7: $OptOpeAIG \leftarrow IdentifyOptOpeAIG(ChaAIG_{list})$; ▷ Identify AIGs with optimal number of operations
8: $OptLogLevAIG \leftarrow IdentifyOptLogAIG(ChaAIG_{list})$; ▷ Identify AIGs with optimal number of logic levels
9: $SRAMRange_{list} \leftarrow IdentifySRAM(OptOpeAIG, OptLogAIG, SRAM_{list})$; ▷ Determine a set of SRAM topologies based on the total number of gate counts in the AIGs.
10: **for all** $SRAM$ in $SRAMRange_{list}$ **do** ▷ Loop through each SRAM topology
11: $AIGMetrics_{list}[SRAM] \leftarrow Evaluate(OptLogLevAIG, SRAM)$; ▷ Evaluate power, latency, and energy of lowest gate count AIG for each SRAM topology
12: $AIGMetrics_{list}[SRAM] \leftarrow Evaluate(OptOpeAIG, SRAM)$; ▷ Evaluate power, latency, and energy of lowest logic level AIG for each SRAM topology
13: **end for**
14: $BestAIG \leftarrow FilterEnergy(AIGMetrics_{list})$; ▷ Determine lowest energy consuming AIG
15: $L_{res} \leftarrow CalculateInductor(BestAIG.SRAM)$; ▷ Calculate the inductor size for the chosen SRAM topology
16: **Output:** rCiM Architecture $\leftarrow BestAIG.SRAM$; ▷ Resulting rCiM architecture along with its corresponding inductor size

range of SRAM topologies ($\text{SRAMRange}_{\text{list}}$), considering the total number of gate counts. The range of SRAM topologies is chosen to accommodate all inputs and outputs. The memory size is chosen to be at least four times the number of gates (two inputs + two outputs per gate), accounting for cases where complementary outputs are required. For example, an AIG with 128 gates requires 256 bits for inputs and 256 bits for outputs, requiring a minimum of 512 bits. Based on the AIGs chosen from Line 7 and Line 8, the algorithm determines a list of suitable SRAM topologies ($\text{SRAMRange}_{\text{list}}$) from the available range of SRAM topologies.

The For loop (Lines 10–13) iterates through the library of SRAM topologies ($\text{SRAM}_{\text{list}}$) to compute the power, latency, and energy consumption metrics for the optimal SRAM ($\text{AIGMetrics}_{\text{list}}$[SRAM]) associated with optimal AIGs considering lowest gate count(Line 11) and lowest logic level (Line 12). In Lines 11 and 12, power, latency, and energy metrics are derived through an analytical estimation approach combined with initial simulation data. We performed standard SRAM characterization for various topologies using postlayout analysis in Cadence Virtuoso, obtaining accurate power and latency values for different SRAM configurations. These results were used to evaluate typical read, write, precharge, and logic computation cycles for rCiM. Line 14 is used to identify optimal AIG with the lowest energy consumption among all the SRAM topologies. Line 15 uses the optimal SRAM topology to calculate the sizing of the resonant inductor. This methodology would result in the most optimal rCiM architecture implementation for the given RTL netlist.

The time complexity of the proposed methodology is determined by the number of AIGs (n) with k levels. In addition, the number of available SRAM topologies also plays a crucial role and is defined by m. The overall time complexity is expressed using BigO notation as $\boldsymbol{O(n) = O(m + nk)}$. In this work, the analysis was performed using 12 different SRAM topologies and four synthesis transformations. These four synthesis transformations resulted in 64 unique AIG synthesis options, thus setting the number of AIGs (n) to 64 and the size of m-to-12. As m and n are relatively small, the time complexity becomes linear and is primarily affected by the size of the levels in the AIG k.

IV. Experiments and Results

A. Experimental Setup

To demonstrate the efficacy of the proposed algorithm, we analyzed EPFL combinational benchmark suite circuits [39] synthesized using YOSYS [38]. The logic optimization of AIGs is performed using ABC [37]. We explored 64 unique AIG synthesis options for each benchmark circuit, analyzing them across 12 different SRAM topologies for cache sizes ranging from 4 to 192 kB. The rCiM architecture was designed using Taiwan Semiconductor Manufacturing Company (TSMC) 28-nm technology, and the transient simulations were performed using the Cadence Spectre simulator. Our study utilized a library of SRAM macros with sizes of 4, 8, 16, and 32 kB. Three different topologies were employed for a comprehensive analysis of each macro size resulting in 6912 unique AIG implementations.

B. AIG Transformation Analysis

Fig. 9 shows the comparison of power, latency, and energy consumption across all 6912 unique AIGs, considering 12 distinct rCiM topologies using nine EPFL combinational benchmark circuits. The single-macro topology is limited to performing only one type of logical operation per computational cycle. In contrast, the SRAM topology with three macros can execute NAND2, NOR2, and NOT operations concurrently in each macro. For example, the three logical operations can be conducted concurrently using two macros in any six-macro implementation.

Fig. 9(a) shows the comparison of the overall power consumption of each benchmark circuit. The power consumption for both the single-macro and three-macro implementations remains the same, as the total number of operations is constant. The three-macro implementation can perform three

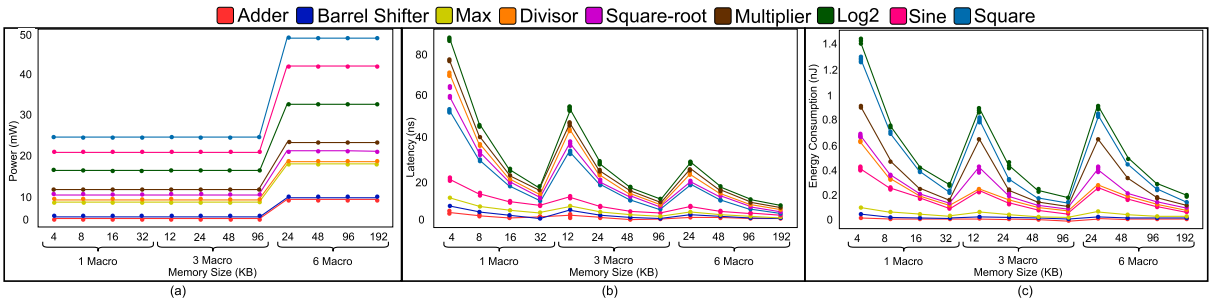

Fig. 9. After mapping each benchmark circuit to different SRAM architectures, we computed the power, latency, and energy. (a) Power consumption remained nearly constant for single macro and three macro SRAMs; however, it doubled for six macro SRAMs. (b) Six macro implementation achieves up to 66% average lower latency compared single-macro implementation. (c) Average energy consumption for single-macro implementations decreases up to 47% while using an 8-kB SRAM macro compared with a 4-kB macro.

times the number of operations performed by a single-macro implementation in a single cycle, but the total number of operations required for a whole combinational logic remains the same. As a result, while the power per cycle for the three-macro implementation increases by 3×, it consumes 3× fewer clock cycles, leading to the same overall power consumption. However, in the six-macro topology, power consumption increases by a factor of 2× compared with the three-macro implementation. This higher power consumption is primarily due to the doubling of power on the doubled-size macro implementation, even though the number of operations remains the same. The power per cycle for the six-macro implementation increases by 2×, while the number of clock cycles required to complete the operation remains the same as in the three-macro implementation, since the architecture can only perform one logic level per cycle. Thus, the total power consumption of the six-macro implementation is double that of the three-macro implementation.

Fig. 9(b) depicts all the benchmark circuits' latency. In a single macro, latency decreases with an increase in macro size. On average, there is a 47% reduction in latency when the macro area doubles from 4 to 8 kB and a 40% reduction when the macro area goes from 16 to 32 kB. Comparatively, three-macro implementations achieve an average latency reduction of 38%, taking advantage of the ability to perform parallel operations but incurring a 3× area penalty over single-macro implementations. Similarly, six-macro implementations achieve a latency reduction of 47% on average compared with three-macro implementations and a 66% lower latency compared with single-macro implementations. This latency improvement results from the capability to perform more parallel operations but comes at the price of a higher area and power consumption.

Fig. 9(c) illustrates the energy consumption results for all benchmark circuits. The energy consumption for single-macro implementations decreases by 47% while using an 8-kB SRAM macro compared with a 4-kB macro, aligning with the latency reduction as the total power consumption per benchmark computation stays nearly constant. On average, the three-macro implementations exhibit 39% lower energy compared with single-macro implementations. Despite achieving lower latency than three-macro implementations, six-macro implementations, on average, consume 15% higher energy due to higher power consumption.

In Table I, we present a comprehensive comparison of AIG implementations for the EPFL benchmark circuits, highlighting the best and worst-case AIG implementations. In addition, the table provides insights into the number of stages, gate counts, and synthesis transformations employed for each benchmark. The analysis uses four different synthesis options (i.e., B_a, R_f, R_w, and R_s). The analysis shows that employing multiple macros leads to the most energy-efficient design by leveraging concurrent operations. However, excessive macro use can compromise energy efficiency due to increased power consumption.

In the case of the Adder-128 benchmark, which has a small number of operations, dividing a 48-kB SRAM into three macros resulted in significantly lower energy consumption. The benchmark exhibits an 85% reduced energy consumption compared with a single 4-kB macro achieved by concurrent operations. For benchmark circuits with a substantial number of operations, such as Log_2, employing synthesis transformations to reduce 2% of the operations and opting for larger macros to execute a higher number of concurrent operations resulted in a 92% reduction of energy consumption, but with a 24× area penalty. In the case of the Sine circuit, with a moderate gate count, adopting a three-macro implementation of 96-kB SRAM size resulted in an 85.4% reduction in energy consumption. Similarly, using a three-macro implementation of 96-kB SRAM size for the square-root operation showcased a reduction of 93% energy consumption compared with a single 4-kB macro implementation.

In summary, this study highlights the trade-offs between area, latency, and the SRAM topology to achieve an energy-efficient rCiM implementation. To achieve lower latency, we have two main strategies: either increase the size of a single macro or employ multiple smaller macros to carry out parallel operations. For example, in the case of the divisor benchmark circuit, the rCiM circuit achieves a latency reduction of 92% with a 12× SRAM area penalty after utilizing the three-macro SRAM topology.

C. Process Variation Analysis

Fig. 10 evaluates the robustness of the proposed rCiM architecture against process variations for all input cases. We consider three different SRAM topologies: (4 kB) × 3, (8 kB) × 3, and (16 kB) × 3. For each topology, 5000 samples

TABLE I

WHILE COMPARING THE BEST-CASE AND WORST-CASE SCENARIOS OF RCIM TOPOLOGIES, THE THREE-MACRO IMPLEMENTATION, WITH CONCURRENT OPERATION CAPABILITIES, DEMONSTRATES AN AVERAGE ENERGY SAVING OF 89.12% COMPARED WITH SINGLE-MACRO IMPLEMENTATIONS WITH A 4-KB SRAM MACRO SIZE

Benchmark	Scenario	SRAM Macro Size (KB)	Macro Count	Synthesis Transformations	Level Count	NAND2 Gate Count	NOR2 Gate Count	Inverter Gate Count	Power (mW)	Latency (ns)	Energy (nJ)
Adder-128	Best-case	16	3	R_w, R_f, B_a	4	383	765	257	4.62	0.58	0.0027
	Worst-case	4	1	B_a, R_f, R_s	4	170	1102	271	4.63	3.81	0.0176
Barrel Shifter	Best-case	32	3	R_w, R_f, B_a	4	1474	1086	7	4.62	0.73	0.0034
	Worst-case	4	1	R_w, R_s, R_f, B_a	4	1866	1086	7	4.63	6.45	0.0299
Multiplier	Best-case	32	3	B_a	10	6505	20523	8638	11.57	7.395	0.0856
	Worst-case	4	1	B_a, R_w, R_s	10	6447	20545	8639	11.71	77.06	0.9022
Sine	Best-case	32	3	B_a, R_w, R_s, R_f	17	2341	4018	1169	20.80	2.90	0.0603
	Worst-case	4	1	R_f, R_s	18	2419	4107	1120	20.83	20.09	0.4185
Max	Best-case	32	3	R_f, B_a, R_w	8	655	2365	1164	9.25	1.31	0.0121
	Worst-case	4	1	R_s, R_f	8	740	2374	1176	9.26	10.36	0.0959
Divisor	Best-case	32	3	B_a, R_f, R_s, R_w	8	6696	18422	7776	9.26	6.09	0.0564
	Worst-case	4	1	R_w	8	6828	18397	7848	9.39	70.76	0.6641
Square-root	Best-case	32	3	B_a, R_w	9	10677	13561	6057	10.41	4.93	0.0513
	Worst-case	4	1	R_s, R_w, B_a	9	11504	14621	4217	10.53	64.51	0.6792
Square	Best-case	32	3	R_w, R_s, R_f	20	3276	13632	6308	24.28	5.66	0.1373
	Worst-case	4	1	R_f, R_w, R_s, B_a	21	3131	13977	6257	24.36	53.25	1.2973
Log_2	Best-case	32	3	R_f, R_s, B_a	13	10195	21848	7839	16.20	7.40	0.1198
	Worst-case	4	1	R_f, R_w, R_s	14	10482	22348	7836	16.35	87.77	1.4351

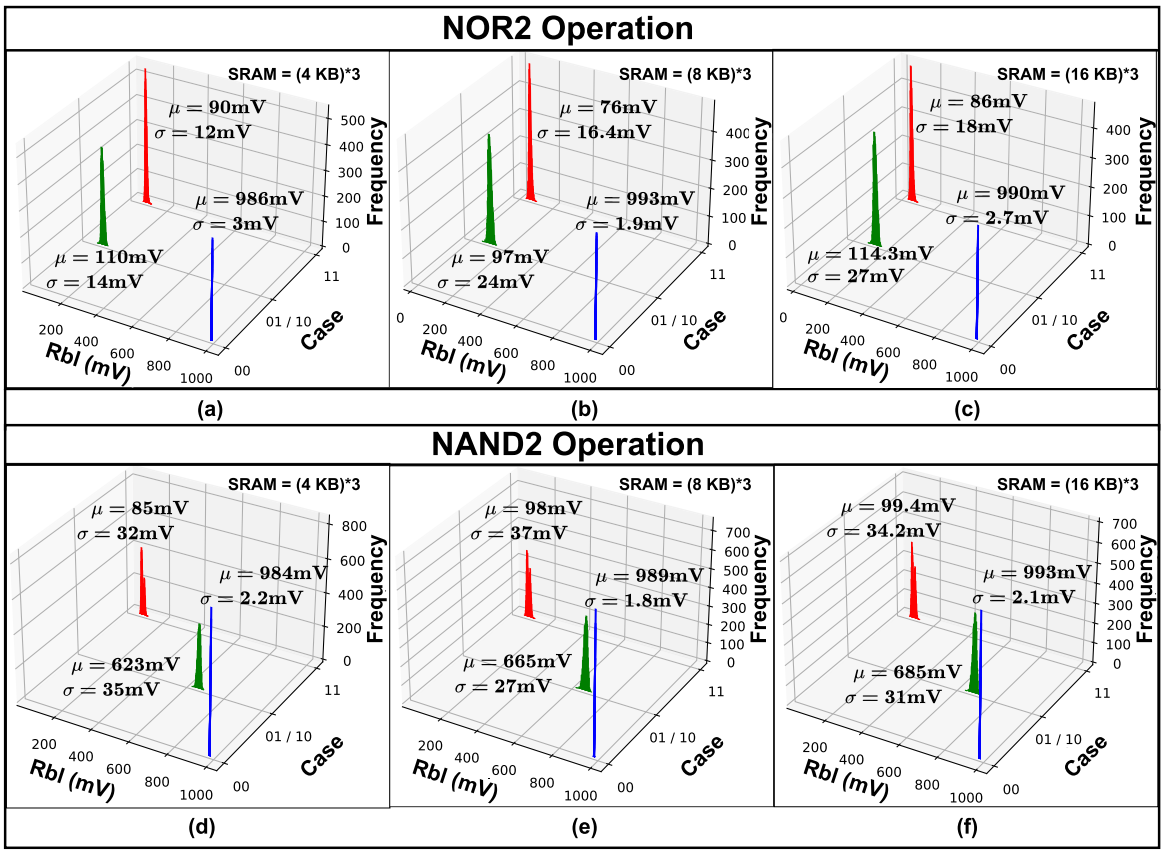

Fig. 10. Monte-Carlo simulations considering 5000 samples of the Rbl discharge across three SRAM topologies, namely (a) 4KB×3, (b) 8KB×3, and (c) 16KB×3 for NOR2 operations, and (d) 4KB×3, (e) 8KB×3, and (f) 16KB×3 for NAND2 operations, each topology evaluated under ±10% length variations with 3σ deviations for the cases "01/10," "00," and "11."

of the Rbl discharge were taken with ±10% length variation of all transistors under 3σ deviations.

The NOR2 operation analysis for the three SRAM topologies is shown in Fig. 10(a)–(c). For the (4 kB)×3 topology shown in Fig. 10(a), the mean Rbl voltages are 110, 986, and 90 mV with standard deviations of 14, 3, and 12 mV for cases "01/10," "00," and "11," respectively. In the (8 kB)×3 topology depicted in Fig. 10(b), the mean Rbl voltages are 97, 993, and 76 mV, with standard deviations of 24, 1.9, and 16.4 mV for the same cases. For the (16 kB)×3 topology shown in Fig. 10(c), the mean Rbl voltages are 114.3, 990, and 86 mV, with standard deviations of 27, 2.7, and 18 mV, respectively.

The NAND2 operation analysis is depicted in Fig. 10(d)–(f). For the (4 kB)×3 topology in Fig. 10(d), the mean Rbl voltages for cases "01/10," "00," and "11" are 623, 984, and

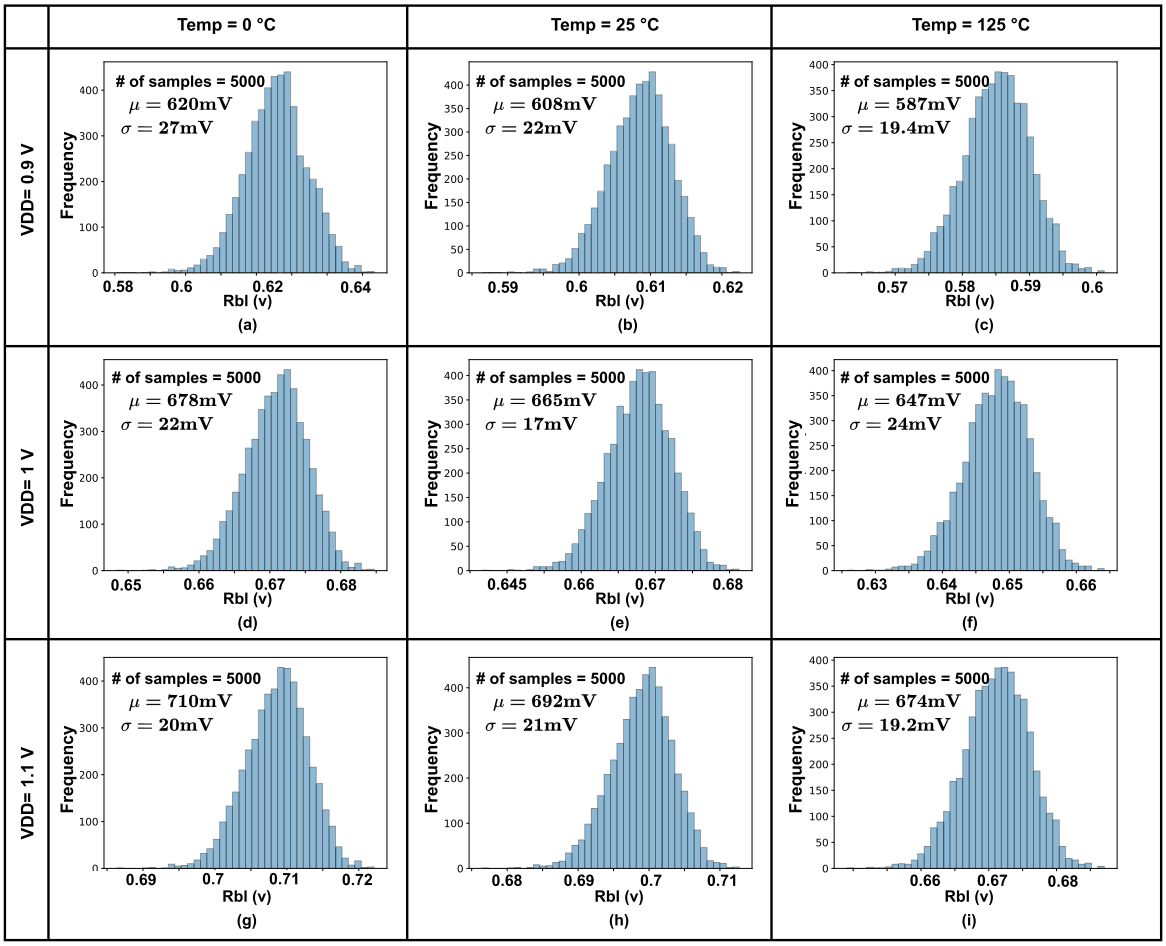

Fig. 11. Monte-Carlo simulations with 5000 samples under ±10% length variation for the borderline NAND2 "01/10" input vector case across VDD = 0.9V (a) Temp = 0°C, (b) Temp = 25°C, (c) Temp = 125°C; VDD = 1.0V (d) Temp = 0°C, (e) Temp = 25°C, (f) Temp = 125°C; and VDD = 1.1V (g) Temp = 0°C, (h) Temp = 25°C, (i) Temp = 125°C.

85 mV, with standard deviations of 35, 2.2, and 32 mV, respectively. In the (8 kB)×3 topology shown in Fig. 10(e), the mean Rbl voltages are 665, 989, and 98 mV, with standard deviations of 27, 1.8, and 37 mV. Finally, for the (16 kB)×3 topology in Fig. 10(f), the mean Rbl voltages are 685, 993, and 99.4 mV, with standard deviations of 31, 2.1, and 34.2 mV, respectively.

Monte-Carlo simulations were performed to evaluate the impact of temperature and voltage variations on the system's performance for the borderline case "01/10" for the (8 kB)×3 SRAM topology. A total of 5000 samples were analyzed for each combination of temperature and voltage. The simulations considered three different temperatures (0 °C, 25 °C, and 125 °C) and three voltage levels (0.9, 1, and 1.1 V). The results, depicted in Fig. 11, show the Rbl discharge distribution values.

At a temperature of 0 °C, the Rbl discharge for voltages of 0.9, 1, and 1.1 V, as illustrated in Fig. 11(a), (d), and (g), respectively, are of significant importance. For 0.9 V, the mean Rbl voltage is 620 mV with a standard deviation of 27 mV. At 1 V, the mean Rbl voltage is 608 mV with a standard deviation of 22 mV. For 1.1 V, the mean Rbl voltage is 587 mV with a standard deviation of 19.4 mV.

At 25 °C, the Rbl discharge for voltages of 0.9, 1, and 1.1 V, as shown in Fig. 11(b), (e), and (h), respectively, have been thoroughly analyzed. The mean Rbl voltage for 0.9 V is 647 mV with a standard deviation of 24 mV. For 1 V, the mean Rbl voltage is 665 mV with a standard deviation of 17 mV. For 1.1 V, the mean Rbl voltage is 678 mV with a standard deviation of 22 mV.

At a higher temperature of 125 °C, the Rbl discharge for voltages of 0.9, 1, and 1.1 V are presented in Fig. 11(c), (f), and (i), respectively. The mean Rbl voltage for 0.9 V is 710 mV with a standard deviation of 20 mV. For 1 V, the mean Rbl voltage is 692 mV with a standard deviation of 21 mV. For 1.1 V, the mean Rbl voltage is 674 mV with a standard deviation of 19.2 mV.

Fig. 12 demonstrates the robustness of the readout circuitry. We simulated 5000 samples with ±10% length variation and 3σ deviations in the SA, shown in Fig. 4, considering an 8-kB SRAM rCiM architecture. Fig. 12(a) and (d) shows the input case "00" for NAND2 and NOR2 operations, respectively. As Rbl does not discharge in the "00" case, the output of the SA (D_{comp}) remains at logic "1." For Fig. 12(c), (e), and (f), corresponding to NAND2 input case "11" and NOR2 input cases "01/10" and "11," the Rbl completely discharges, resulting

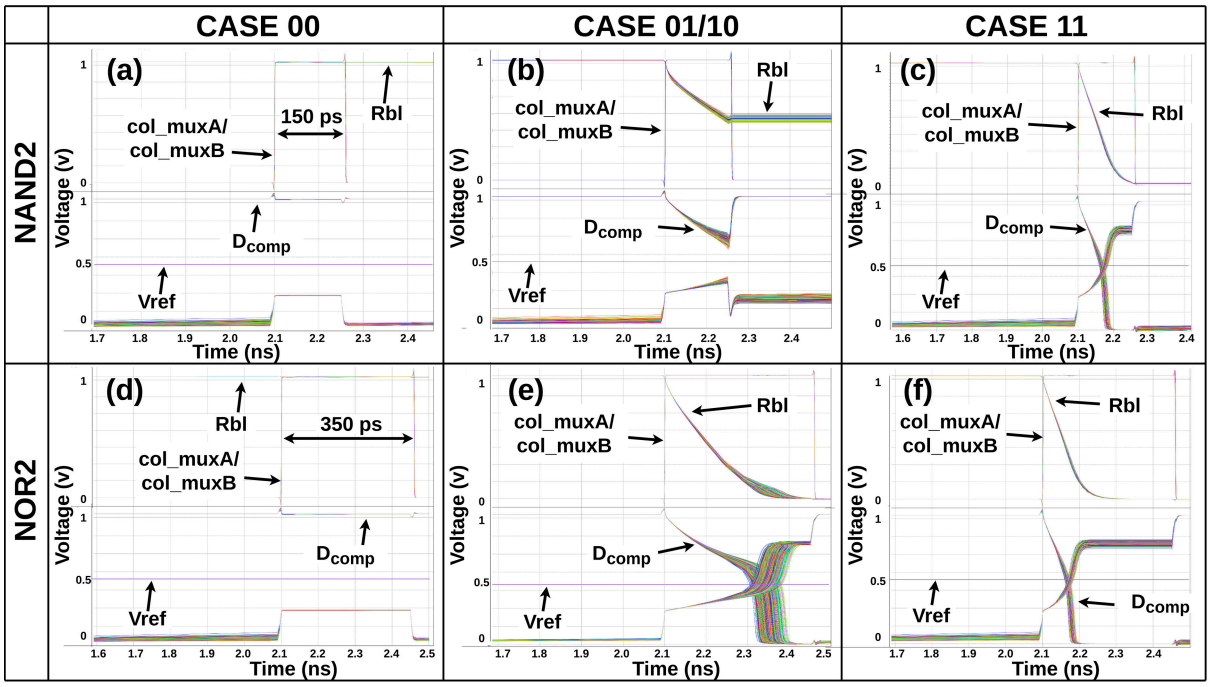

Fig. 12. Process variation analysis of the readout circuit considering the NAND2 operation cases (a) "00," (b) "01/10," and (c) "11," and the NOR2 operation cases (d) "00," (e) "01/10," and (f) "11," depicting successful computational results of the sense amplifier conducted with 5000 samples, incorporating 3σ deviations under $\pm 10\%$ length variation.

TABLE II

COMPARISON OF THE PROPOSED RCIM ARCHITECTURE USING THREE SRAM TOPOLOGIES WITH PREVIOUS WORKS SHOW 2.6× HIGHER THROUGHPUT AND 1.6× GREATER ENERGY EFFICIENCY COMPARED WITH [22], AND ACHIEVING 2.12× HIGHER ENERGY EFFICIENCY THAN [64]

	This work			TVLSI'21 [65]	ISSCC'19 [22]	DAC'20 [64]	DAC'19 [66]	TVLSI'23 [33]	JSSC'23 [67]
Technology	28nm			40nm	28nm	28nm	28nm	28nm	28nm
Cell Type	10T dual read port			7T	8T	6T	6T	6T	8T
Array Size	(256x256)x1	(256x256)x3	(512x256)x3	1Kb	(128x256)x4	(128x128)x4	256x64	128x128	128x128
Supply Voltage (V)	1V			0.9	0.6-1.1	0.6-1.1	1	0.8	0.75
Frequency (GHz)	1GHz			0.1	0.475	2.25	2.2	0.633	0.113
Throughput (GOPS)	88.2-106.6	264.83-320	529.66-640	5.594 44.752 (normalized to 8KB)	32.7	NA	560 (normalized to 8KB)	162 (normalized to 8KB)	1851
Energy Efficient (TOPS/W)	8.64-10.45	8.64-10.45	17.18-20.77	7.66 8.86 (normalized to 28nm)	0.55 (mult), 5.27 (add)	0.68 (mult), 8.09 (add)	NA	NA	270.5
Compute Density (GOPS/mm^2)	551.25-666.25			27	27.3	NA	NA	NA	NA
Type of Functions	SRAM/ LOGIC (NAND, NOR, NOT)			SRAM / NAND / NOR / XOR	Logic/ ADD/ SUB/ MULT/ DIV/ FP	SRAM/ LOGIC/ ADD/ MULT	SRAM/ Logic/ ADD/ Shift/ Copy	SRAM/ Logic/ ADD/ Compare	SRAM/ Logic/ Copy/ Matrix Transpose

in a logic "0" for D_{comp} value. In the NAND2 "01/10" case [Fig. 12(b)], where Rbl partially discharges, the pulsewidth characterization ensures that Rbl voltages do not drop below Vref voltage, resulting in the correct D_{comp} value of logic "0."

D. Architecture Comparison With Previous Works

A comparison of the proposed rCiM architecture with existing CiM architectures is presented in Table II. The proposed architecture consumes 65 fJ per NAND2 operation and 116 fJ per NOR2 operation, achieving a throughput ranging from 88.2 to 106.6 GOPS, depending on NAND2 and NOR2 operations, with an 8-kB single macro implementation. The energy efficiency remains constant when transitioning from a single-macro to a three-macro implementation. While throughput increases by 3× due to more operations being performed, the power consumption per cycle also increases by 3×, resulting in no net improvement in energy efficiency. However, when the array size is increased for the three-macro implementation, the power consumed by the computational circuits rises, but the control circuitry's overhead remains constant. This results in improved energy efficiency, as the increased throughput is greater than the increase in power consumption, leading to a higher overall energy efficiency. The proposed architecture achieves 551.25–666.25 GOPS/mm^2, depending on the number of NAND2 and NOR2 operations. All throughput values of the compared works have been normalized to an 8-kB memory size.

Wang et al. [65] propose a 7T bitcell and 2T switch are used for single-bit Boolean logic, addition, and multiplication operations. As this work is implemented in 40-nm technology, we have used Dennard's power scaling law [68] to scale the power and obtain the energy efficiency. The proposed rCiM architecture achieves a 10× higher frequency and 15% greater energy efficiency with an 8-kB single macro implementation and a 2.2× higher energy efficiency with a 16-kB three-macro implementation.

In [22], the transposable 8T cell performs multibit "add" and "multiplication" operations but has a lower frequency that results in higher energy/operation consumption. The proposed single-macro 8-kB rCiM architecture achieves 2.1× higher frequency, resulting in an increase of throughput by 2.6× and an increase in energy efficiency by 1.6× when compared with [22].

In [64], the architecture boosts the bitline for computing to avoid read disturb issues, resulting in higher energy consumption. The proposed architecture overcomes read-disturb issues with a dedicated dual read-port bitcell, achieving 2.12× higher

energy efficiency with a 16-kB three-macro implementation compared with [64].

Simon et al. [66] present a high-speed 6T SRAM cell capable of performing bitwise addition, shift, and copy operations while mitigating read disturbance issues by incorporating an additional inverter and transistor to each bitline. Similarly, Wang et al. [33] introduces a 6T compute-SRAM architecture with dual-split-VDD assist in addressing read disturbance concerns. In contrast, our work utilizes dedicated read ports to eliminate read disturbance problems, which are prevalent in 6T SRAM-based CiM architectures. The throughput reported for both [33] and [66] is normalized to an 8-kB SRAM array. While these works demonstrate higher throughput than the single-macro implementation, they do not account for the additional write-back cycle required for output storage, which adds additional latency to each computation cycle. In [67], the architecture stores the computational outputs directly in the same bitcell where the inputs are applied, resulting in significant latency and power savings. However, the reported throughput does not account for the additional latency required to read the operands and apply them as inputs to the bitcells. In addition, designing this unconventional 8T SRAM requires a higher level of design expertise. In contrast, the proposed rCiM architecture operates at an 8.8× higher frequency, leading to more efficient and conventional read and write operations.

V. Conclusion

This article proposes an architectural exploration tool designed to identify the optimal rCiM cache topology tailored to specific logical operations. The novel rCiM architecture facilitates concurrent NAND2/NOR2/NOT operations using three-macro and six-macro topologies, significantly reducing latency for logical operations. Furthermore, the rCiM architecture incorporates a series resonance-based write driver, effectively lowering the consumed dynamic power during write operations by recycling the energy dissipated. The proposed algorithm utilizes only the RTL and a list of available SRAM topologies as input, streamlining the process of exploring the most energy-efficient topology for the given RTL. Comprehensive analysis conducted on EPFL combinational benchmark circuits demonstrates a notable average energy savings of 40.52% across all the designs when employing the three-topology design implementations, as opposed to a single-macro implementation with the same macro size. The proposed three-topology implementation achieves 5.2× higher throughput compared with [35], and 8.2× higher throughput when compared with [33]. The robustness analysis was conducted using Monte Carlo simulations with 5000 samples, considering temperature variations, ±10% VDD, and ±10% variations in transistor lengths. The analysis shows that the mean bitline discharge of 665 mV with a standard deviation of 17 mV for case "10/01" of NAND2 operation, which falls within the sensing range of VDD/2 of the SA. Under the temperature and voltage variations the mean bitline discharge for case "10/01" of NAND2 operation ranged between 710 to 587 mV with a standard deviation range of 27–17 mV.

References

[1] M. Ali et al., "A 65 nm 1.4–6.7 TOPS/W adaptive-SNR sparsity-aware CIM core with load balancing support for DL workloads," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2023, pp. 1–2.

[2] R. Sreekumar, M. Park, M. N. Sakib, B. S. Reniwal, K. Lee, and M. R. Stan, "EASI-CiM: Event-driven asynchronous stream-based image classifier with compute-in-memory kernels," in *Proc. 25th Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2024, pp. 1–8.

[3] H. Wang, R. Liu, R. Dorrance, D. Dasalukunte, D. Lake, and B. Carlton, "A charge domain SRAM compute-in-memory macro with C-2C ladder-based 8-bit MAC unit in 22-nm FinFET process for edge inference," *IEEE J. Solid-State Circuits*, vol. 58, no. 4, pp. 1037–1050, Apr. 2023.

[4] X. Si et al., "A local computing cell and 6T SRAM-based computing-in-memory macro with 8-b MAC operation for edge AI chips," *IEEE J. Solid-State Circuits*, vol. 56, no. 9, pp. 2817–2831, Sep. 2021.

[5] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE J. Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, Nov. 2018.

[6] M. Ali, A. Jaiswal, S. Kodge, A. Agrawal, I. Chakraborty, and K. Roy, "IMAC: In-memory multi-bit multiplication and accumulation in 6T SRAM array," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2521–2531, Aug. 2020.

[7] S. Cheon, K. Lee, and J. Park, "A 2941-TOPS/W charge-domain 10T SRAM compute-in-memory for ternary neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 5, pp. 2085–2097, May 2023.

[8] J. Song, X. Tang, X. Qiao, Y. Wang, R. Wang, and R. Huang, "A 28 nm 16 kb bit-scalable charge-domain transpose 6T SRAM in-memory computing macro," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 5, pp. 1835–1845, May 2023.

[9] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–490.

[10] D. Challagundla, I. Bezzam, and R. Islam, "A resonant time-domain compute-in-memory (rTD-CiM) ADC-less architecture for MAC operations," in *Proc. Great Lakes Symp. VLSI*. New York, NY, USA: Association for Computing Machinery, Jun. 2024, pp. 268–271, doi: 10.1145/3649476.3658773.

[11] S. Ananthanarayanan, B. S. Reniwal, and A. Upadhyay, "Design and analysis of multibit multiply and accumulate (MAC) unit: An analog in-memory computing approach," in *Proc. 36th Int. Conf. VLSI Design 22nd Int. Conf. Embedded Syst. (VLSID)*, Jan. 2023, pp. 109–114.

[12] B. J. Kailath and B. S. Reniwal, "CiMComp: An energy efficient compute-in-memory based comparator for convolutional neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2024, pp. 1–2.

[13] S. Yan et al., "A 28-nm floating-point computing-in-memory processor using intensive-CIM sparse-digital architecture," *IEEE J. Solid-State Circuits*, vol. 59, no. 8, pp. 2630–2643, Aug. 2024.

[14] L. Lu, A. Mani, and A. T. Do, "A 129.83 TOPS/W area efficient digital SOT/STT MRAM-based computing-in-memory for advanced edge AI chips," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5.

[15] P.-C. Wu et al., "An 8b-precision 6T SRAM computing-in-memory macro using time-domain incremental accumulation for AI edge chips," *IEEE J. Solid-State Circuits*, vol. 59, no. 7, pp. 2297–2309, Jul. 2024.

[16] Y.-W. Chen, R.-H. Wang, Y.-H. Cheng, C.-C. Lu, M.-F. Chang, and K.-T. Tang, "SUN: Dynamic hybrid-precision SRAM-based CIM accelerator with high macro utilization using structured pruning mixed-precision networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 7, pp. 2163–2176, Jul. 2024.

[17] R. Wang and X. Guo, "A hierarchically reconfigurable SRAM-based compute-in-memory macro for edge computing," in *Proc. IEEE 5th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2023, pp. 1–5.

[18] J. B. Shaik, X. Guo, and S. Singhal, "Impact of aging and process variability on SRAM-based in-memory computing architectures," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 6, pp. 2696–2708, Jun. 2024.

[19] S. Zhang, X. Cui, F. Wei, and X. Cui, "An area-efficient in-memory implementation method of arbitrary Boolean function based on SRAM array," *IEEE Trans. Comput.*, vol. 72, no. 12, pp. 3416–3430, Dec. 2023.

[20] K. Prasad, A. Biswas, A. Kabra, and J. Mekie, "PIC-RAM: Process-invariant capacitive multiplier based analog in memory computing in 6T SRAM," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.

[21] K. Soundrapandiyan, S. K. Vishvakarma, and B. S. Reniwal, "Enabling energy-efficient in-memory computing with robust assist-based reconfigurable sense amplifier in SRAM array," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 1, pp. 445–455, Mar. 2023.

[22] J. Wang et al., "A compute SRAM with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 224–226.

[23] J. Chen, W. Zhao, Y. Wang, and Y. Ha, "Analysis and optimization strategies toward reliable and high-speed 6T compute SRAM," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 4, pp. 1520–1531, Apr. 2021.

[24] J. Chen, W. Zhao, Y. Wang, Y. Shu, W. Jiang, and Y. Ha, "A reliable 8T SRAM for high-speed searching and logic-in-memory operations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 6, pp. 769–780, Jun. 2022.

[25] J. Wang et al., "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 76–86, Jan. 2020.

[26] V.-N. Dinh, N.-M. Bui, V.-T. Nguyen, D. John, L.-Y. Lin, and Q.-K. Trinh, "NUTS-BSNN: A non-uniform time-step binarized spiking neural network with energy-efficient in-memory computing macro," *Neurocomputing*, vol. 560, Dec. 2023, Art. no. 126838. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122300961X

[27] T. Li et al., "CafeHD: A charge-domain FeFET-based compute-in-memory hyperdimensional encoder with hypervector merging," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2024, pp. 1–6.

[28] M. Yang et al., "CILP: An arbitrary-bit precision all-digital compute-in-memory solver for integer linear programming problems," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2024, pp. 1–2.

[29] J. Mu, C. Yu, T. T.-H. Kim, and B. Kim, "A scalable and reconfigurable bit-serial compute-near-memory hardware accelerator for solving 2-D/3-D partial differential equations," *IEEE J. Solid-State Circuits*, vol. 59, no. 8, pp. 2706–2716, Aug. 2024.

[30] H. Ajmi et al., "Efficient and lightweight in-memory computing architecture for hardware security," *J. Parallel Distrib. Comput.*, vol. 190, Aug. 2024, Art. no. 104898. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731524000625

[31] J. Cai et al., "Energy efficient data search design and optimization based on a compact ferroelectric FET content addressable memory," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, New York, NY, USA, Feb. 2022, pp. 751–756 doi: 10.1145/3489517.3530527.

[32] Y. Chen, J. Mu, H. Kim, L. Lu, and T. T. Kim, "BP-SCIM: A reconfigurable 8T SRAM macro for bit-parallel searching and computing in-memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 5, pp. 2016–2027, May 2023.

[33] Y. Wang, S. Zhang, Y. Li, J. Chen, W. Zhao, and Y. Ha, "A reliable and high-speed 6T compute-SRAM design with dual-split-VDD assist and bitline leakage compensation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 5, pp. 684–695, May 2023.

[34] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (TCAM/BCAM/SRAM) using push-rule 6T bit cell enabling logic-in-memory," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, Apr. 2016.

[35] Z. Lin et al., "Two-direction in-memory computing based on 10T SRAM with horizontal and vertical decoupled read ports," *IEEE J. Solid-State Circuits*, vol. 56, no. 9, pp. 2832–2844, Sep. 2021.

[36] Y. Huang, Z. Chen, D. Li, and K. Yang, "CAMA: Energy and memory efficient automata processing in content-addressable memories," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Apr. 2022, pp. 25–37.

[37] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. 22nd Int. Conf. Comput. Aided Verification*, Edinburgh, U.K. Cham, Switzerland: Springer, 2010, pp. 24–40.

[38] C. Wolf. *Yosys Open SYnthesis Suite*. Accessed: 2024. [Online]. Available: https://yosyshq.net/yosys/

[39] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *Proc. 24th Int. Workshop Log. Synth. (IWLS)*, 2015, pp. 1–5.

[40] P.-C. Wu et al., "A floating-point 6T SRAM in-memory-compute macro using hybrid-domain structure for advanced AI edge chips," *IEEE J. Solid-State Circuits*, vol. 59, no. 1, pp. 196–207, Jan. 2024.

[41] C. Duan et al., "DDC-PIM: Efficient algorithm/architecture co-design for doubling data capacity of SRAM-based processing-in-memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 3, pp. 906–918, Mar. 2024.

[42] A. Malhotra, A. K. Saha, C. Wang, and S. K. Gupta, "ADRA: Extending digital computing-in-memory with asymmetric dual-row-activation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 8, pp. 3089–3093, Mar. 2023.

[43] Y. Hui, Q. Li, L. Wang, C. Liu, D. Zhang, and X. Miao, "In-memory Wallace tree multipliers based on majority gates within voltage-gated SOT-MRAM crossbar arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 3, pp. 497–504, Mar. 2024.

[44] S. Sridharan, J. R. Stevens, K. Roy, and A. Raghunathan, "X-former: In-memory acceleration of transformers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 8, pp. 1223–1233, Aug. 2023.

[45] A. Dongre, B. Boro, and G. Trivedi, "ADC-less reprogrammable RRAM array architecture for in-memory computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 12, pp. 2053–2060, Dec. 2023.

[46] S. Liu et al., "HARDSEA: Hybrid analog-ReRAM clustering and digital-SRAM in-memory computing accelerator for dynamic sparse self-attention in transformer," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 2, pp. 269–282, Feb. 2024.

[47] Z. Lu et al., "An RRAM-based computing-in-memory architecture and its application in accelerating transformer inference," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 3, pp. 485–496, Mar. 2024.

[48] S. Choi, D. Han, C. Choi, and Y. Seo, "Layout-aware area optimization of transposable STT-MRAM for a processing-in-memory system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 2, pp. 245–255, Feb. 2024.

[49] H. Zhang et al., "CP-SRAM: Charge-pulsation SRAM marco for ultra-high energy-efficiency computing-in-memory," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, New York, NY, USA, Jul. 2022, pp. 109–114, doi: 10.1145/3489517.3530398.

[50] A. Basak Chowdhury, B. Tan, R. Karri, and S. Garg, "OpenABC-D: A large-scale dataset for machine learning guided integrated circuit synthesis," 2021, *arXiv:2110.11292*.

[51] R. Islam, B. Saha, and I. Bezzam, "Resonant energy recycling SRAM architecture," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 4, pp. 1383–1387, Apr. 2021.

[52] R. V. Joshi, M. M. Ziegler, and H. Wetter, "A low voltage SRAM using resonant supply boosting," *IEEE J. Solid-State Circuits*, vol. 52, no. 3, pp. 634–644, Mar. 2017.

[53] D. Challagundla, I. Bezzam, and R. Islam, "Design automation of series resonance clocking in 14-nm FinFETs," *Circuits, Syst., Signal Process.*, vol. 42, no. 12, pp. 7549–7579, Aug. 2023, doi: 10.1007/s00034-023-02458-4.

[54] D. Challagundla, "Power and skew reduction using resonance energy recycling in FinFET based wideband clock networks," Master's thesis, Dept. Comput. Sci. Elect. Eng., Univ. Maryland, Baltimore, MD, USA, 2022.

[55] D. Challagundla, M. Galib, I. Bezzam, and R. Islam, "Power and skew reduction using resonant energy recycling in 14-nm FinFET clocks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 268–272.

[56] R. Islam, D. Challagundla, and I. Bezzam, "System and methods of reducing wideband series resonant clock skew," U.S. Patent App. 18/627 479, Oct. 10, 2024.

[57] R. Islam, "Low-power resonant clocking using soft error robust energy recovery flip-flops," *J. Electron. Test.*, vol. 34, no. 4, pp. 471–485, Jun. 2018, doi: 10.1007/s10836-018-5737-6.

[58] D. Challagundla, I. Bezzam, B. Saha, and R. Islam, "Resonant compute-in-memory (rCIM) 10T SRAM macro for Boolean logic," in *Proc. IEEE 41st Int. Conf. Comput. Design (ICCD)*, Nov. 2023, pp. 110–117.

[59] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, "OpenRAM: An open-source memory compiler," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2016, pp. 1–6.

[60] S. Nalam, M. Bhargava, K. Mai, and B. H. Calhoun, "Virtual prototyper (ViPro): An early design space exploration and optimization tool for SRAM designers," in *Proc. 47th Design Autom. Conf.* New York, NY, USA: Association for Computing Machinery, Jun. 2010, pp. 138–143, doi: 10.1145/1837274.1837310.

[61] M. Liu, X. Tang, K. Zhu, H. Chen, N. Sun, and D. Z. Pan, "OpenSAR: An open source automated end-to-end SAR ADC compiler," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, Oct. 2021, pp. 1–9.

[62] J. Chen et al., "AutoDCIM: An automated digital CIM compiler," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2023, pp. 1–6.

[63] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective* (Prentice Hall electronic and VLSI series), 2nd ed., Upper Saddle River, NJ, USA: Pearson, 2004.
[64] K. Lee, J. Jeong, S. Cheon, W. Choi, and J. Park, "Bit parallel 6T SRAM in-memory computing with reconfigurable bit-precision," in *Proc. 57th ACM/IEEE Des. Autom. Conf. (DAC)*, Jun. 2020, pp. 1–6.
[65] C.-C. Wang, L. K. S. Tolentino, C.-Y. Huang, and C.-H. Yeh, "A 40-nm CMOS multifunctional computing-in-memory (CIM) using single-ended disturb-free 7T 1-Kb SRAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 12, pp. 2172–2185, Dec. 2021.
[66] W. Simon, J. Galicia, A. Levisse, M. Zapater, and D. Atienza, "A fast, reliable and wide-voltage-range in-memory computing architecture," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.
[67] Z. Lin et al., "In situ storing 8T SRAM-CIM macro for full-array Boolean logic and copy operations," *IEEE J. Solid-State Circuits*, vol. 58, no. 5, pp. 1472–1486, May 2023.
[68] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. JSSC-9, no. 5, pp. 256–268, Oct. 1974.

Dhandeep Challagundla (Graduate Student Member, IEEE) received the M.S. degree from the University of Maryland at Baltimore County (UMBC), Baltimore, MD, USA, in August 2022, where he is currently working toward the Ph.D. degree at the Computer Science and Electrical Engineering Department.

His research interests revolve around energy-efficient computing, compute-in-memories, static random-access memory (SRAM) design, low-power circuit design, mixed-signal IC design, and electronic design automation (EDA) tools.

Ignatius Bezzam (Member, IEEE) received the B.Tech. degree from IIT Madras, Chennai, India in 1983, and the Ph.D. degree in electrical engineering from Santa Clara University, Santa Clara, CA, USA, in 2015.

He holds several key patents in analog mixed-signal integrated circuit (IC) design with publications in top international conferences, including the International Solid-State Circuits Conference (ISSCC), European Solid-State Circuits Conference (ESSCIRC), IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS (TCAS).

Dr. Bezzam has owned 30 first silicon successes with global teams, with 33 years of next generation chip design experience in Silicon Valley, Europe, and Asia.

Riadul Islam (Senior Member, IEEE) was an Assistant Professor with the University of Michigan, Dearborn, MI, USA, from 2017 to 2019. He is currently an Assistant Professor with the Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD, USA. In his Ph.D. dissertation work at the University of California at Santa Cruz (UCSC), Santa Cruz, CA, USA, he designed the first current-pulsed flip-flop/register that resulted in the first-ever one-to-many current-mode clock distribution networks for high-performance microprocessors. He holds two U.S. patents and several IEEE/ACM/MDPI/Springer Nature journal and conference publications. His current research interests include digital, analog, and mixed-signal CMOS ICs/systems-on-chip (SoCs) for a variety of applications; verification and testing techniques for analog, digital and mixed-signal ICs; hardware security; CAN network; CAD tools for design and analysis of microprocessors and field-programmable gate arrays (FPGAs); automobile electronics; and biochips.

Dr. Islam is a member of the ACM, IEEE Circuits and Systems (CAS) Society, the VLSI Systems and Applications Technical Committee (VSA-TC) of the IEEE-CAS, and IEEE Solid-State Circuits (SSC) Society. He was a recipient of the 2021 NSF ERI Award, the 2021 Maryland Industrial Partnerships (MIPS) Award, and the 2021 Maryland Innovation Initiative (MII) Award. He is an Associate Editor of *Circuits, Systems, and Signal Processing* (CSSP) journal (Springer). He was a Technical Program Committee (TPC) Member of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2022), ACM Great Lakes Symposium on VLSI (GLSVLSI 2020, GLSVLSI 2021, and GLSVLSI 2022), 57th IEEE/ACM Design Automation Conference (DAC) 2020 LBR Session, IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2021, and IEEE International Conference on Consumer Electronics (ICCE) 2021.