Energy-efficient Multi-Operand XOR Logic-based CIM Accelerator using RRAM technology

Abhairaj Singh[1,2], Konstantinos Stavrakakis[1], Rajendra Bishnoi[1], Rajiv V. Joshi[1] and Said Hamdioui[1]

[1]Computer Engineering Laboratory, TU Delft, The Netherlands: (k.stavrakakis, r.k.bishnoi, r.v.joshi, s.hamdioui)@tudelft.nl
[2]Hybrid Cloud, IBM Research Europe, Rüschlikon, Switzerland: abhairaj.singh@ibm.com

2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 979-8-3315-1560-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICCAD66269.2025.11240975

Abstract*—Recent advances in *Resistive RAM* (RRAM) based *Computation-in-Memory* (CIM) architectures highlight significant potential for accelerating data-intensive computing tasks. However, non-idealities in RRAM devices, such as variability, result in small sensing margins that can significantly affect the computational efficiency. This issue becomes even more pronounced when dealing with complex multi-operand logic operations. This paper introduces a circuit-level scheme for CIM-based multi-operand XOR logic operations, leveraging a *Voltage-to-Time converter* (VTC) to perform multi-phased XORs in a single clock cycle. In this approach, we exploit bitline capacitances for voltage-based sensing during computation, generating an output voltage that is linearly proportional to the operand values. This voltage is then converted into the desired logic output using the VTC design. Furthermore, low-power techniques are employed in the deployment of sense amplifiers, such as regulating power consumption during operation and disabling the amplifiers once the decision is made. Simulation results for a post-layout extracted 512x512 (256Kb) RRAM-based CIM array show that up to 16-operand XOR operation can be accurately and reliably performed as opposed to a maximum of three operands supported by state-of-the-art solutions, while offering up to 49× better figure-of-merit combining energy-efficiency and throughput.

I. Introduction

Non-volatile computation-in-memory (CIM) has been vastly explored to realize logic and arithmetic units, thus alleviating the need for excessive movement of data between storage and processing units [1]. In particular, CIM based on resistive random access memory (RRAM) devices that are found to be highly scalable, CMOS-compatible, non-volatile storage elements, can inherently allow the storing and processing of data within the memory array [2, 3]. These RRAM-based CIM architectures are extensively employed to accelerate the exclusive OR (XOR) and other logic operations that can be used for many applications such as database queries, encryption, etc [4–8]. However, their efficacy is severely affected by their analog nature of storing and computing data, resistance drifts in RRAMs, device variation, etc. These inherent properties can impose serious limitations on the computing accuracy and the energy efficiency of the CIM architecture [9, 10].

Several two and multi-operand CIM-based XOR implementations have been proposed where *all* operands reside in the memory. These can be classified into three broad classes; i) cascading and combining partial results obtained by either sequentially performing OR and NAND operations [7, 11, 12], ii) or by sequentially performing XOR operations [4, 8, 13, 14], and iii) dedicated sense amplifier (SA) and multi-reference generating blocks [6, 7, 13, 15–19]. However, these solutions suffer from many challenges. First, the sequential nature of executing partial results and accumulating in the periphery heavily impacts the overall performance and energy efficiency. Second, the SAs incur high area and consume high energy as they need to generate several reference schemes that are controlled typically using multi-input MUXs. Third, due to the static nature of their referencing schemes, these solutions result in small read margins, especially when considering multi-operand operations in the presence of process variation and wire parasitics; this reduces the computing accuracy and limits the maximum number of operands. Last, unlike multi-operand (N)OR and (N)AND operations proposed in [20], where critical states deciding the logic output are either all 0's or all 1's, XOR functionality demands the output to change in an alternate manner with an incremental number of 1's among the operands, which is extremely challenging with the current methods. On the other hand, few logic solutions have adopted differential sensing scheme [3, 21, 22]; however, the use of multiple references implies that either they face the aforementioned issues or are limited to multi-operand (N)OR and (N)AND operations [6, 20]. In short, there is a need for cost-effective circuit-level solutions that not only provide accurate and energy-efficient XOR operations but also push the limit of maximizing the number of operands.

This paper presents MOXOR-CIM, a multi-operand XOR-based CIM using compact voltage-to-time conversion (VTC) techniques for high-performance and energy-efficient computing. Our techniques accurately and reliably perform XOR operations with up to 16 operands in a single cycle, while realizing up to 12× speedup and 5.5× energy efficiency compared to state-of-the-art CIM-based XOR solutions [4, 8, 11–19]. The key contributions are:

- Proposes two schemes: a compact one-reference, uni-polar VTC (UVTC)-based scheme and a high-speed reference-less bipolar VTC (BVTC)-based scheme with trade-off analyses of the overall efficiency and the maximum number of operands supported.
- Integrates UVTC and BVTC schemes in CIM by utilizing bitline caps for voltage-based sensing and computing, whereby, CIM generates an output voltage signal linear to the operand values which is converted into desired logic output by the aforementioned VTC-based designs.
- Reports simulation results using a post-layout extracted 256x8 RRAM-based CIM and a comprehensive circuit and system-level comparison using an emulated 512x512 RRAM array.

The rest of the paper is organized as follows. Section II covers the fundamentals and challenges related to XOR operations using RRAM-based CIM. Section III presents the proposed MOXOR-CIM schemes, followed by results in Section IV. Finally, Section V concludes the paper.

II. Background

A. RRAM-based CIM architecture

CIM performs *in-situ* analog computing within a memory unit by leveraging Kirchhoff's and Ohm's law [1], as shown in Fig. 1a. By exploiting the cross-point structure of the memory unit, logic, and arithmetic operations can be performed in each column. CIM

This work is partially funded by the European Union, CONVOLVE (Grant No. 101070374), NEUROKIT2E (Grant No. 101112268) and DAIS (Grant No. 101007273.

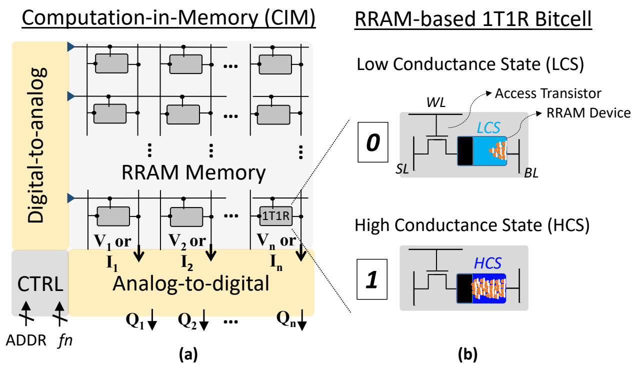


Fig. 1: (a) CIM architecture (b) Typical RRAM-based bitcell design.

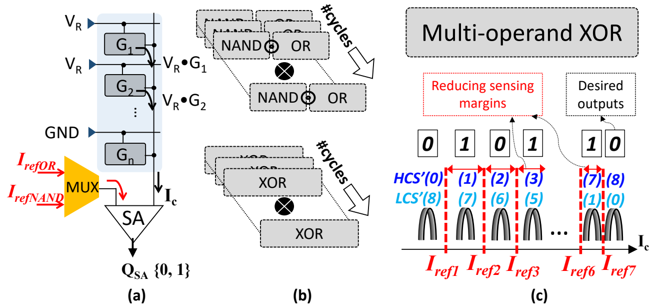


Fig. 2: (a) Working of XOR, operation; Typical implementations via (b) sequential cascading, and (c) multi-operand XOR operations.

performs logic operation f_n on the operands with addresses $ADDR$ using row drivers and customized SAs in the periphery; note that these are replaced by digital-to-analog and analog-to-digital converters, respectively, for analog multiply-and-accumulate operations [6]. RRAM device has become a natural choice of memory storing device due to its fast read/write latency, high ON/OFF conductance ratio, and inherent ability to store and compute data. Fig. 1b shows a typical one-transistor-one-resistor (1T1R) bitcell configuration based on RRAM devices, which are based on the reversible formation of a conductive filament delivering low (LCS) and high conductance states (HCS) [23]. Data bits '0' and '1' are represented as LCS and HCS of the RRAM, respectively.

B. XOR implementations and their challenges

The underlying principle of implementing an analog CIM-based logic operation is converting aggregated information of the operands into analog form (voltage or current) as a function of equivalent compute output. Typically, to perform a logic operation, a multi-row READ is performed by selecting the desired operands, and based on the aggregated conductance of the selected bitcells, CIM generates an output in the analog domain. Depending on the different types and number of operations, and the number of operands supported, one or several reference signals aid the sense amplifier (SA) in determining the desired output. Fig. 2a illustrates the interaction of selector signal V_R with conductances of the selected bitcells (operands) i.e., $V_R{\cdot}G_1$ and $V_R{\cdot}G_2$, resulting in total accumulated column current I_c; the references corresponding to OR or NAND can be selected using a MUX to perform the desired operations via SA.

CIM-based n-operand XOR operation represented as XOR^n, can be performed using either a cascading of operations [4, 7, 8, 11–14] or performing one-time multi-operand XOR [6, 17, 18]. The top and bottom parts of Fig. 2b illustrate 1) Cascading two-operand OR and NAND operations, and 2) cascading two-operand XOR operations, respectively. In these approaches, an operation is performed in each cycle and the partial result is stored in the periphery for further interaction with partial results in the subsequent cycles. However, this sequential approach of performing a series of OR·NAND or XOR operations significantly suffers from high latency and energy consumption, where each of these metrics increases *linearly* with operand size n. In addition, post-processing to accumulate the partial results in the periphery adds to large area overheads. On the other hand, XOR^n in a single cycle typically requires multi-reference signals I_{ref*} to differentiate each state. Fig. 2c illustrates this for XOR^8. Note that the signal margin, defined as the minimum sensing margin between two states, reduces as the operand size increases due to circuit non-idealities such as IR drop [6, 12, 20]. To address this, dedicated design units are introduced, such as complex SA and adaptive reference signals owing to an inefficient system. Unfortunately, this scheme is not scalable in terms of operand size as further reduction in sensing margin cannot be addressed.

III. Proposed VTC Techniques

This section presents the details of our proposed schemes.

A. Overview

The underlying principle is based on VTC to perform XOR^n of arbitrary operand size n, as described in Fig. 3. The entire operation is divided into two broad phases: 1) READ phase, where CIM generates an output voltage XORV being *linearly* proportional to the aggregated conductance value XORG (representing accumulated conductance of the selected data bits, XORG = $\sum_1^n G_i$); and COMPUTE phase, where the developed XORV is translated into *linearly* proportional time duration XORT. In the COMPUTE phase, first an initial comparison of BL voltage with a reference voltage (for scheme UVTC) or with complementary BL voltage (for scheme BVTC) is performed to generate a SIGN-bit. Then, the combination of this SIGN-bit and the least significant bit (LSB) of a global counter (*i.e.,* CNT<0>) are used to determine XOR^n output based on whether XORT (corresponding to the selected operands) have an ODD or an EVEN parity. Note that XOR cannot be performed within the READ phase using such a time-based approach, since the time to discharge the BL to reach a certain voltage threshold is *non-linear i.e.,* time to discharge is inversely proportional to XORG.

We leverage a two-transistor-two-resistor (2T2R) bitcell to have the following key advantages over the typical 1T1R bitcell; i) inherent ∼2X READ sensing margin [20], ii) inherent addressing of circuit non-idealities due to a common-mode (complementary) structure, thereby, providing a *linear* XORG-XORV relation, and iii) aids the use of a reference-less scheme for a memory READ operation. Subsequently, we present two approaches to perform XOR^n; i) UVTC: a compact scheme using a single reference, and ii) BVTC: a high-speed and energy-efficient reference-less scheme.

B. Proposed MOXOR-CIM Architecture

1) Bitcell structure and memory operations:: Fig. 4a shows a typical complementary 2T2R bitcell comprised of two select NMOS pass transistors sharing a common node CSL, and each is connected to bitlines BL and NBL through an RRAM device storing complementary data bits, respectively. RRAM devices can be programmed to such states using the scheme shown in the figure, where a voltage magnitude of V_W is applied in opposite polarity to switch the RRAM device from LCS to HCS, and vice-versa. The switching

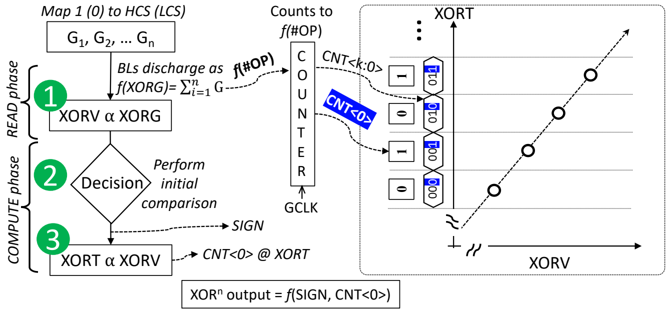


Fig. 3: (a) Overview of our proposed schemes

characteristics of an RRAM device concerning the polarity and the magnitude of the voltage applied are shown in the top half of Fig. 4c.

Fig. 4b shows a memory READ scheme using differential voltage ΔV= $|V_{BL}$-$V_{NBL}|$ sensing on the respective bitlines BL and NBL. The select transistors are enabled via a row-select wordline signal (WL) and ΔV is developed due to different conductance values of HCS and LCS. This ΔV is sensed and amplified using a SA, provided a minimum margin of ΔV_{SAmin} is developed between the two input signals of the SA. In the remainder of this paper, we assume ΔV_{SAmin}=$40mV$ [12, 20]. Owing to the availability of differential BL and NBL signals with 2T2R bitcells, a reference-less READ scheme requires only ΔV=$40mV$, as opposed to a minimum of ΔV=$80mV$ required in single-ended sensing using a reference signal for 1T1R bitcells. As shown in the bottom half of Fig. 4c, 1T1R and 2T2R are suffixed to their corresponding WL, BL0, and NBL0 signals, where BL0, NBL0 correspond to a READ 0 and BL1 corresponds to READ 1 in 1T1R bitcells. The figure shows that close to 40% of READ latency can be reduced compared to single-ended sensing involved in reading 1T1R bitcells.

2) XOR functionality:: Our proposed techniques explore the output pattern of XOR^n as a function of the number of 1's in the selected operands. ODD and EVEN parity cases are defined as the cases when ODD and EVEN number of 1's are present in the selected operands, respectively. Here, zero number of 1's is considered EVEN. Note that performing an XOR^n operation on ODD and EVEN parity cases results in logic output 1 and 0, respectively.

Fig. 5 describes how the input pattern of XOR^n translates to XORV that later serves as an input to our VTC-based schemes, using n=5 as an illustration. The figure shows the dependency of XORV in terms of the discharging currents flowing in the BL and NBL. In the left half of Fig. 5, different input cases corresponding to XOR^5 are described at the top of each column, where #BL:#NBL is defined as the number of 1's in the respective bitline sides. In addition, at the bottom of each column, corresponding BL and NBL currents are shown along with the desired XOR^5 output. Naturally, in the case of XOR^5, the total of #BL and #NBL always equals five. With five WLs always activated, naturally the total current corresponding to both BL and NBL is $5I_{ON}$ as well.

Two implementations are proposed to differentiate different input cases from each other; 1) Uni-polar VTC (UVTC): This uses XORV solely related to BL and a reference signal XORREF, and 2) bipolar VTC (BVTC): This uses XORV from both BL and NBL without the use of any reference signal. In UVTC, the polarity and magnitude of the difference in XORV and XORREF are utilized. XORREF is the voltage signal corresponding to $0.5I_{ON}$ discharge current of the reference line, to have an equal sensing margin between I_{OFF} ($<<I_{ON}$) and I_{ON} discharge currents in the BL corresponding to #0:#5 and #1:#4 cases, respectively. This implies a difference

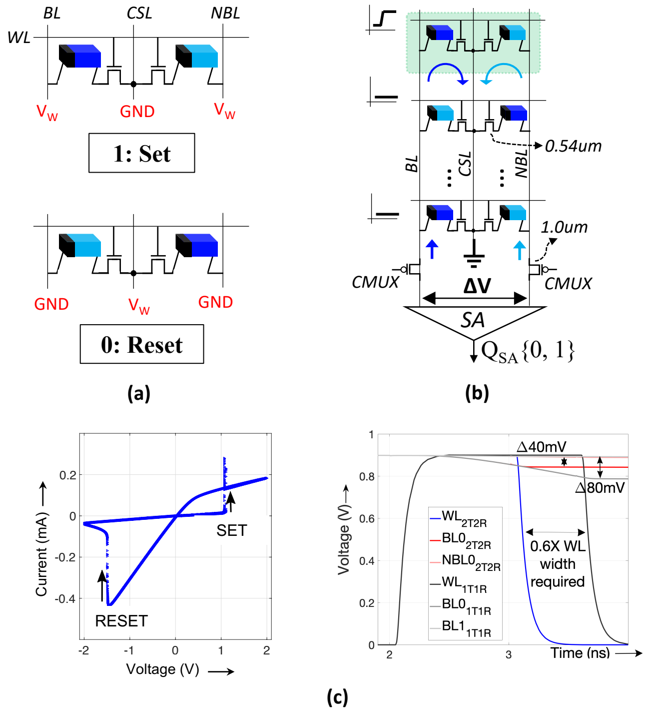
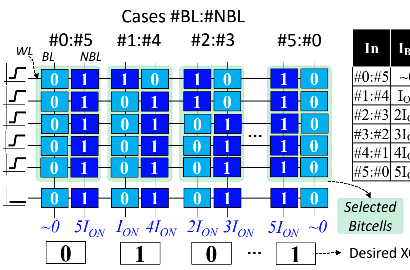


Fig. 4: (a) Memory WRITE and (b) READ operation using 2T2R bitcells, and (c) IV characteristics and timing diagram during READ.

of $0.5I_{ON}$ discharge current translates to the desired difference for the above two cases, albeit with different polarities. The table on the right half of Fig. 5 describes the following; (from left to right) input cases described as #BL:#NBL, BL current I_{BL}, NBL current I_{NBL}, two columns corresponding to UVTC for I_{REF} and ΔI= I_{BL}-I_{REF}, two columns corresponding to BVTC for SIGN and ΔI= $|I_{BL}$-$I_{NBL}|$, and the desired output. Next, we show these different implementations while utilizing different outcomes of SIGN and magnitude corresponding to different input cases.

C. Design and Implementation

MOXOR-CIM implementations follow the following steps; 1) CIM develops an output voltage XORV proportional to the aggregated conductance values XORG in each column of the crossbar. This is achieved in the READ phase, where XORV represents the amount of bitline discharge during the multi-row-select XOR operation, 2) In the decision sub-phase of the COMPUTE phase, XORV is processed to determine the original condition of the bitlines post-READ phase, and 3) Based on this initial condition, a bitline or both bitlines are charged and/or discharged using pull-up and pull-down circuits, meanwhile, SA determines at what time instant XORT there is an alteration from the original condition, and translates XORT to either ODD or EVEN



Truth table: XOR^5

In	I_{BL}	I_{NBL}	UVTC		BVTC		Out
			I_{REF}	ΔI	Sign	ΔI	
#0:#5	~0	$5I_{ON}$	$0.5I_{ON}$	$-0.5I_{ON}$	1	$5I_{ON}$	0
#1:#4	I_{ON}	$4I_{ON}$	$0.5I_{ON}$	$0.5I_{ON}$	1	$3I_{ON}$	1
#2:#3	$2I_{ON}$	$3I_{ON}$	$0.5I_{ON}$	$1.5I_{ON}$	1	I_{ON}	0
#3:#2	$3I_{ON}$	$2I_{ON}$	$0.5I_{ON}$	$2.5I_{ON}$	0	I_{ON}	1
#4:#1	$4I_{ON}$	I_{ON}	$0.5I_{ON}$	$3.5I_{ON}$	0	$3I_{ON}$	0
#5:#0	$5I_{ON}$	~0	$0.5I_{ON}$	$4.5I_{ON}$	0	$5I_{ON}$	1

*$0.5I_{ON}$ is reference required in VTC scheme

Fig. 5: Generation of XORV during a XOR^5 operation.

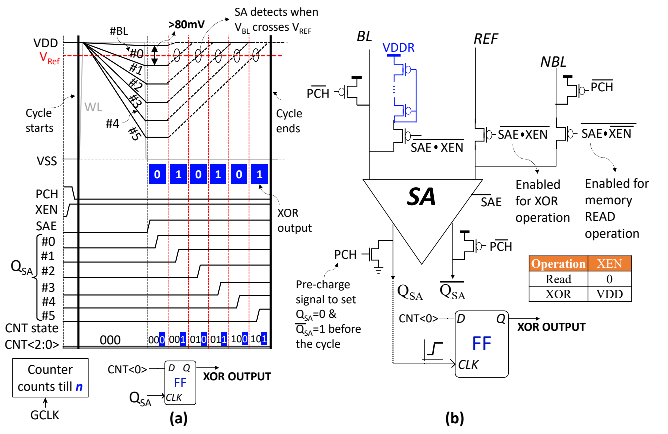


Fig. 6: (a) Timing diagram and (b) design implementation for UVTC.

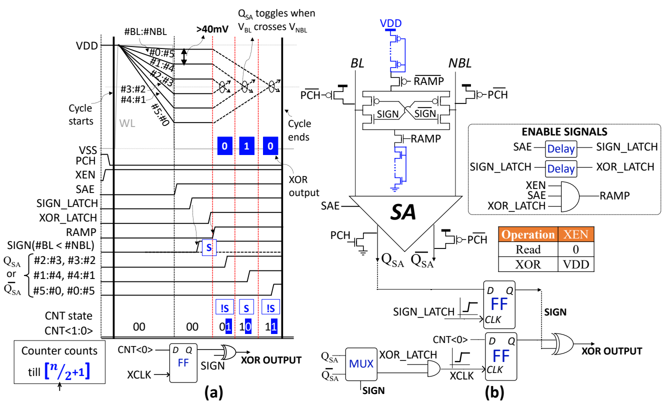


Fig. 7: (a) Timing diagram and (b) design implementation for BVTC.

time sub-phase in the COMPUTE phase. To keep an account of the ODD and EVEN sub-phases of the cycle, a global ripple-carry-counter (GRCC) is used. GRCC is initialized to the RESET state and counts up to a number proportional to the number of operands n during the COMPUTE phase, and its LSB determines the ODD or EVEN sub-phase of the cycle.

1) UVTC:: Implementation of UVTC is presented in Fig. 6 based on a single-ended sensing scheme to perform XOR^n, using XOR^5 as an illustration. Different cases in the figure are represented by #BL, where the number represents the number of 1's on the BL side. After the PCH signal pre-charges the BLs to VDD at the start of the cycle, XORV is generated in the READ phase. Given that XEN is enabled to perform a XOR operation, activating SAE enables the SA and the ramp-up operation. XORREF ensures that sufficient $\Delta V > \Delta V_{SAmin}$=40 mV [20] is developed for both #0 and #1 cases to have a deterministic SA functionality, implying at least 80 mV difference is required between the V$_{BL}$s of #0 and #1 cases. A SA with a complementary initialized output state is used i.e., Q$_{SA}$=0 and $\overline{Q_{SA}}$=1. The output of the SA Q$_{SA}$ toggles to 1, when V$_{BL}$ crosses XORREF. GRCC counts up from a RESET state and CNT<0> is latched at the time instant Q$_{SA}$ toggles to 1 to generate the XOR^5 output. Fig. 6b presents the circuit design to implement this scheme. Note that a tunable stacked (cascoded) pull-up circuit (shown in blue in the figure) is used to ensure a *linear* ramp-up of BL voltage up to VDD using a voltage supply VDDR>VDD.

2) BVTC:: By exploring a differential approach, a bipolar VTC can be used to design a reference-less implementation, as described in Fig. 7. This implies that a 40 mV difference is sufficient between the cases when the number of 1's in the operands increases by 1. This reduces the duration of the READ phase and since there is a fixed voltage dynamic range available, compared to UVTC, BVTC can support close to twice the number of maximum operands. As illustrated earlier, the READ phase is nearly reduced by 40% compared to single-ended UVTC. After the READ phase, the voltages developed on both BL and NBL are utilized in this approach. In the COMPUTE phase, first, SA determines the SIGN-bit depending on whether the initial V$_{BL}$ is greater or smaller than V$_{NBL}$. Following this, depending on this SIGN-bit, one of the bitlines is charged and the other is discharged to arrange a crossing of each other. For instance, SIGN=0 implies initially V$_{BL}$<V$_{NBL}$, therefore, BL is charged and NBL is discharged towards each other. Similar to the UVTC approach, the magnitude of the difference in V$_{BL}$ and V$_{NBL}$ is determined when SA toggles again. GRCC counts up from a RESET state till [n/2+1] and CNT<0> is latched at the time instant Q$_{SA}$ toggles again. XOR output is obtained by performing SIGN$\oplus$CNT<0>.

Fig. 7 presents the circuit design to implement this scheme. In the COMPUTE phase, after a sufficient delay margin to allow SA to resolve the SIGN, the SIGN-bit is latched using a delayed version of the SAE signal i.e., SIGN_LATCH. SIGN-bit *selectively* enables the ramp circuit to ramp up or down the bitlines. Two cascoded stacks of transistors (shown in blue) are introduced to ensure *linear* ramp signals. Note that to ensure a linear ramp, the voltage supply of the ramp circuit is now VDD (and not VDDR) since the BL or NBL ramp-up voltage stays well below VDD. After another sufficient delay of SIGN_LATCH, XOR_LATCH is enabled to activate the latching out of CNT<0> when Q$_{SA}$ is toggled. Performing SIGN$\oplus$CNT<0> generates the final XOR output. An area overhead of 30% is incurred in BVTC-based periphery logic compared to UVTC.

However, in the case of EVEN number of operands, where #BL and #NBL are equal, a sufficient sensing margin can not be developed as BL and NBL discharge equally in the READ phase, thus, resulting in $\Delta V << \Delta V_{SAmin}$. This may lead SA to resolve into an erroneous output. To address this, a dummy row is introduced with the same configuration as the array bitcells i.e., 2T2R. The idea is to bias one of the bitlines with an additional current (equal to the magnitude of one I$_{ON}$), which alleviates the case of having equal currents when the data and complementary bits have an equal number of 1's. This implies that there is always a minimum signal margin for the SA to work reliably. Note that this dummy row is only activated in the EVEN number of operands, which is a pre-defined information that can be easily configured before the operation.

3) Low-power SA design:: SA takes up the largest power share in any READ scheme, especially in our scheme where SA is activated for a longer duration compared to the conventional schemes. In order to reduce the SA power consumption, two mitigation techniques are introduced; i) adaptive biasing current of the SA, and ii) disabling SA as soon as it resolves the outcome. Fig. 8a and Fig. 8b present the circuit implementation of the proposed SAs for the UVTC and BVTC XOR schemes, respectively. A two-stage amplification is introduced, where the inner regenerative cross-coupled inverters are minimum-sized transistors. Regarding the regularization of the biasing current I$_{SA}$ of the SA, in the case of UVTC implementation, BL voltage biases the tail transistor N$_{TAIL}$ of the SA. During the COMPUTE phase, BL is charged from a low voltage value towards VDD, and only when V$_{BL}$ reaches close to V$_{REF}$ (which is close to VDD) is when SA is tuned to have maximum amplification gain. On average, almost 70-80% of the power is expected to be saved compared to a scenario where SA is strongly biased for the entirety of the

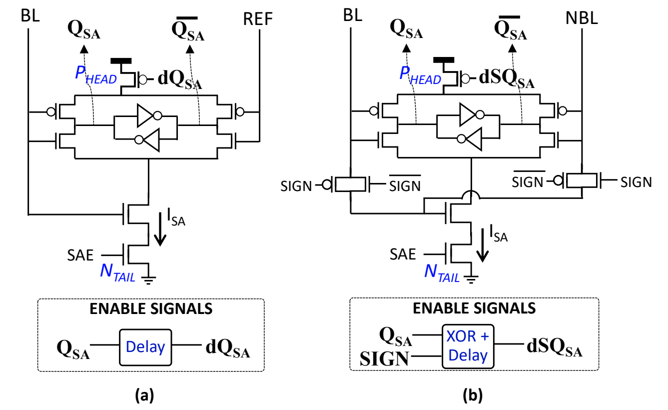


Fig. 8: Low-power SA designs for (a) UVTC, and (b) BVTC.

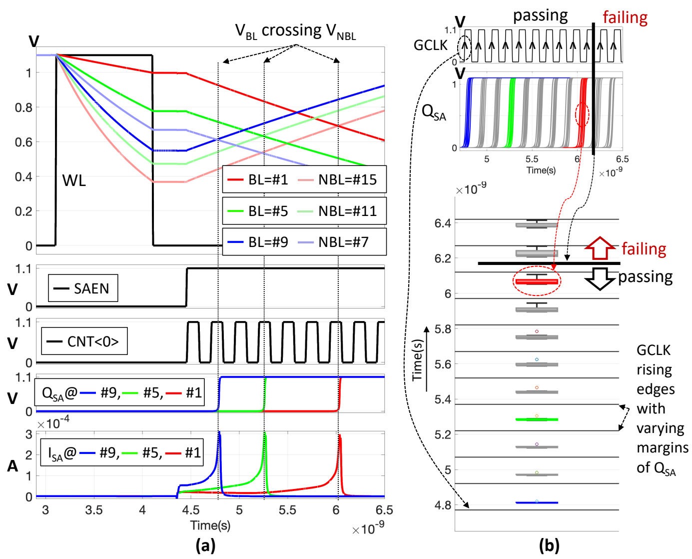


Fig. 9: (a) Timing diagram for BVTC-based XOR[16] showing bitline pairs and low-power SA (biasing I_{SA}) with adaptive biasing and disabling SA upon decision. (b) All possible Q_{SA} distribution with worst-case corners failing after 8 cycles (16 operands).

COMPUTE phase. In the case of BVTC implementation, SIGN-bit dependent biasing is introduced where a ramping-up signal (BL or NBL) is connected to N_{TAIL} with similar power savings. Fig. 9a shows instantaneous power consumed by SA in the BVTC case performing XOR[16]. This is illustrated using the operand cases #1, #5, and #9, where I_{SA} quadratically increases and only reaches the required biasing current near the crossing of the two bitlines. Regarding the disabling of SA, in the case of UVTC, a header P_{HEAD} is added to the cross-coupled inverting amplifier-based circuit. P_{HEAD} disconnects the power supply as SA makes a decision *i.e.,* Q_{SA} is set to 1 following the detection of V_{BL} crossing V_{REF}. Similarly, in the case of BVTC, a SIGN-dependent signal is generated to disable P_{HEAD}. On average, a further 50% of the power is saved compared to a scenario where SA is enabled for the entirety of the COMPUTE phase. This can be seen in Fig. 9a where the instantaneous I_{SA} drops to near zero right after Q_{SA} toggles to 1.

D. System-level Analysis

1) Application Selection:: To exercise the up-to-16-row XOR datapath we benchmark a parity-checking kernel representative of many I/O pipelines. Low-density parity-check (LDPC) decoding is selected because it realises the same large-fan-in XOR reduction found in WiFi 6/7 (IEEE 802.11 ax/ay), WiMAX 802.16, 5G NR, DVB-S2/S2X, SATA/NVMe and DDR5 on-die ECC controllers. A single codeword invokes O(10^3) parity equations, each XORing tens of bits, and the whole set is revisited over several iterations. Crucially, the parity matrix is immutable - written once and read repeatedly - so the workload is read-heavy and endurance-friendly, aligning with the non-volatile RRAM fabric. Because the same motif underlies BCH/CRC engines and RAID parity updates, LDPC decoding offers a conservative baseline for the broader class of XOR-centric applications our macro targets.

Algorithm 1: Bit-flip LDPC decoding on the 512×512 BVTC MOXOR-CIM tile

Input : $\boldsymbol{r} \in \{0,1\}^N$, $\boldsymbol{H}^\top$ in CIM
Output : decoded codeword $\boldsymbol{v}$
$\boldsymbol{v} \leftarrow \boldsymbol{r}$, $\mathbf{S} \leftarrow \mathbf{0}$
for *iter* $= 1$ **to** *MAX_ITER* **do**
  for $k = 0$ **to** *N-1 16* **do**
    activate($\boldsymbol{v}[k{:}k+15]$)
    $\mathbf{P} \leftarrow$read_columns()
    $\mathbf{S} \leftarrow \mathbf{S} \oplus \mathbf{P}$
  if $\mathbf{S} = \mathbf{0}$ **then return** $\boldsymbol{v}$
  for $i = 0$ **to** N-1 **do**
    $\Delta \leftarrow \sum_{c \in \mathcal{M}(i)} S_c$
    if $\Delta \geq T$ **then** $v_i \leftarrow v_i \oplus 1$
  $\mathbf{S} \leftarrow \mathbf{0}$
return $\boldsymbol{v}$

2) Low Density Parity Check:: The decoder operates in two stages, as illustrated in Algorithm 1. First, the CIM tile gathers the *syndrome* entirely in-situ: the transpose of the parity-check matrix $\mathbf{H}^\top$ is programmed once, and the controller streams up to sixteen tentative bits per cycle. Each sense amplifier XORs the selected word-lines and folds the result into a 1-bit latch placed beside it; after $\lceil N/k \rceil$ bursts the latch bank holds $\mathbf{s} = \mathbf{H}\mathbf{v}^\top \bmod 2$. If the syndrome is non-zero a hard bit-flip rule updates the variable nodes in digital logic, otherwise decoding halts. Because the array is never re-programmed and only word-lines toggle, the procedure remains fully compatible with the RRAM endurance budget.

IV. Results

A. Simulation setup

Table I summarizes the design specifications used for our circuit-level analysis. SA is designed to accurately sense a minimum ΔV of 40 mV while accommodating random and systematic variations of the SA and fluctuations caused by RRAM variations. T_{SAmin} is defined as the time required by the SA to make a decision when the input voltage polarities are reversed. With $3\,\sigma$ variation analysis, a minimum of $126\,ps$ is required to reliably determine the SA outcome. A pessimistic requirement of $150\,ps$ cycle time (determined by GCLK) of the GRCC is configured to allow reliable SA functionality. This allows reliably performing up to 16 (8) operands in the BVTC (UVTC) case *i.e.,* 8 GRCC counts with GCLK running at 6.67 GHz. The figure also shows the layout view of different HfO_2/TiO_x RRAM-based bitcells, where the proposed 2T2R bitcell consumes 61% more area than the conventional 1T1R bitcell. Access transistor (NMOS) is $540\,nm/40\,nm$ and it typically provides a resistance of $1.1\,K\Omega$. Vertical (Horizontal) wire resistance adds up to $0.4\,\Omega$ ($0.8\,\Omega$) and total capacitance to $0.3\,fF$ ($0.6\,fF$) per unit bitcell, thus, contributing to the IR drop and delay mismatch along the row and columns. Fig. 10 presents the layout footprint of CIM equipped with TSMC 40nm CMOS technology. The layout comprises a 256x8 crossbar array built using a 2T2R bitcell configuration.

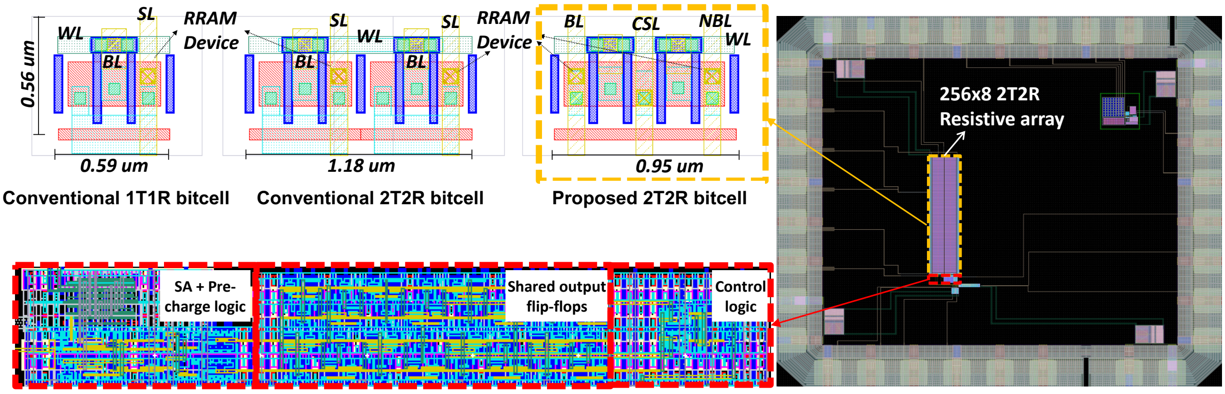


Fig. 10: Layout footprint for the resistive array, proposed MOXOR design, and conventional and proposed bitcell designs.

B. Circuit-level Results and Comparison

Fig. 9b shows the variation analysis of all possible input combinations of XOR^{16} while highlighting three cases where operands are denoted by the number of ON devices on the BL side *i.e.,* #1, #5 and #9 (also highlighted in Fig. 9a). GCLK running at 6.67 GHz (period of 150 ps) as determined by T_{SAmin} implies that CNT<0> toggles every 150 ps. This requires that the toggling of Q_{SA} must occur within this 150 ps period to ensure correct functionality. Note that since there are an EVEN number of operands, the dummy bitcell is enabled and it injects I_{ON} on the BL side and I_{OFF} ∼0 on the NBL side. The top of Fig. 9b shows the variation of the rising edge of Q_{SA} with respect to the rising edge of GCLK and these are also denoted by boxplot and horizontal lines, respectively, in the bottom half of the figure. It can be seen that $3\,\sigma$ variation fails beyond operating with 16 operands while having sufficient sensing margin up to 8 GCLK cycles. This ensures that up to 8 and 16 operands can be reliably performed using UVTC and BVTC schemes, respectively.

For comparison, we present data corresponding to four solutions to perform XOR^n; i) cascading $\text{OR}^2 \oplus \text{NAND}^2$ [7, 15], ii) cascading XOR^2 or multi-operand XOR [6, 13, 16, 18], iii) proposed UVTC, and iv) BVTC. The combined latency and the combined energy related to the READ phase *i.e.,* including WL decoding and driver delay, bitline discharge, T_{SA}, setup/hold time of address and configuration bits, bitline pair pre-charge before the start of the next cycle, are defined as T_{READ} and E_{READ}, respectively. In addition, the combined latency and combined energy dedicated to the COMPUTE phase are defined as T_{COMP} and E_{COMP}, respectively. In short, total latency and energy to perform XOR^n is $\text{T}_{READ}+\text{T}_{COMP}$ and $\text{E}_{READ}+\text{E}_{COMP}$, respectively.

In terms of latency, T_{READ} is similar to that of conventional memory READ or two-operand logic since there is no additional circuitry in the critical path for the UVTC case and is 40% less for the BVTC case, as illustrated earlier. T_{COMP} includes the following; i) the number of GRCC counts i.e., $n\times\text{T}_{SAmin}$ in the UVTC case and $\lfloor n/2+1\rfloor\times\text{T}_{SAmin}$ in the BVTC case, and an additional ii) time required by the SA to reliably determine the SIGN a.k.a an additional T_{SAmin} and a small delay before the ramping sub-phase of the COMPUTE phase starts. Therefore, total XOR^n latency is $\text{T}_{READ}+n\times\text{T}_{SAmin}$ and $0.6\times\text{T}_{READ}+\lfloor n/2+1\rfloor\times\text{T}_{SAmin}$, respectively, for UVTC and BVTC implementations. Whereas, in the case of conventional approaches, cascading OR and NAND lead to $n\times(\text{T}_{READ}+\text{T}_{SA})$ and cascading XOR leads to $\lfloor\frac{n+1}{2}\rfloor\times(\text{T}_{READ}+\text{T}_{SA})$. Similarly, in terms of energy, there is no penalty in E_{READ} comparable to conventional memory READ or two-operand logic. The impact of enabling SA i.e., E_{COMP} for a longer time on overall energy is significantly reduced thanks to the low-power techniques.

The resulting latency and energy to perform XOR^{16} in a column are reported in Table II. A performance improvement of 3.4-9.5× (2.1-5.6×) and an energy improvement of 4.4-13× (2.6-7.7×) is achieved by utilizing BVTC (UVTC) scheme compared to the state-of-the-art logic accelerators. In addition, up to an operand size of 16 (8) can be reliably supported using BVTC (UVTC) scheme, whereas prior-art is limited to an operand size of 3 [6, 18]. Between our two MOXOR-CIM implementations, a performance and energy efficiency improvement of 1.6× and 1.7×, respectively, can be achieved at an expense of ∼30% additional area-overhead in the periphery logic (∼6% in the CIM tile) in the BVTC case compared to UVTC.

C. System-level Results and Comparison

System-level evaluation targets twelve IEEE 802.11n LDPC modes $\{N,R\} = \{648,1296,1944\} \times \{1/2,2/3,3/4,5/6\}$. For each code we decode one full frame under the worst-case iteration cap MAX_ITER = 20. A python profiler instruments our reference decoder and records for every iteration the number of (k)-row activation cycles $C_{\text{act}} = \lceil N/k\rceil$, column-sensing events $512\,C_{\text{act}}$, and digital bit-flip operations $\sum_i\,[\Delta_i \geq T]$. These counters are post-multiplied by the per-operation latency t_{op} and energy E_{op} characterised at circuit level for each candidate architecture Pinatubo [15], FeMIC [18], STT-MRAM-CIM [12], UVTC and BVTC. Frame energy is $E_{\text{frame}} = C_{\text{act}}(E_{\text{act}} + 512E_{\text{sense}}) + \sum_i E_{\text{flip}}$. Frame latency

Parameters	Specifications
CIM Array	512x512
RRAM Device	HfO_2/TiO_x [23]
HCS/LCS	100 $K\Omega$ / 3 $K\Omega$ (±20 %)
Voltage supply	1.1 V (±10 %)
CMOS (variations)	SVT, 40 nm TSMC (3 σ)
Temperature	-40°C to 125°C
ΔV_{SAmin}, T_{SAmin}	40 mV, 126 ps

TABLE I: Design parameters.

Design Metrics	[13]	[15]	[18]	[16]	[7]	[6]	UVTC	BVTC
Technology (nm)	90	65	45	45	90	65	**40**	**40**
Max. XOR Ops/cycle	2	2	4	2	2	3	**8**	**16**
Latency (ns)	20.8*	41*	16*	17*	48	24.5	**6.2**	**3.6**
Energy (fJ)	272	362	131	156*	176	-	**64**	**38**
Energy/Op (fJ)	17	22.7	8.2	9.7	11	-	**4**	**2.4**
Reference signals	2	2	3	2	1	3	**1**	**0**
Variation Tol.	No	No	Yes	Yes	No	Yes	**Yes**	**Yes**

TABLE II: Comparison of our proposed design with prior techniques. * indicates normalized to RRAM-based READ of 2 ns and 15 fJ.

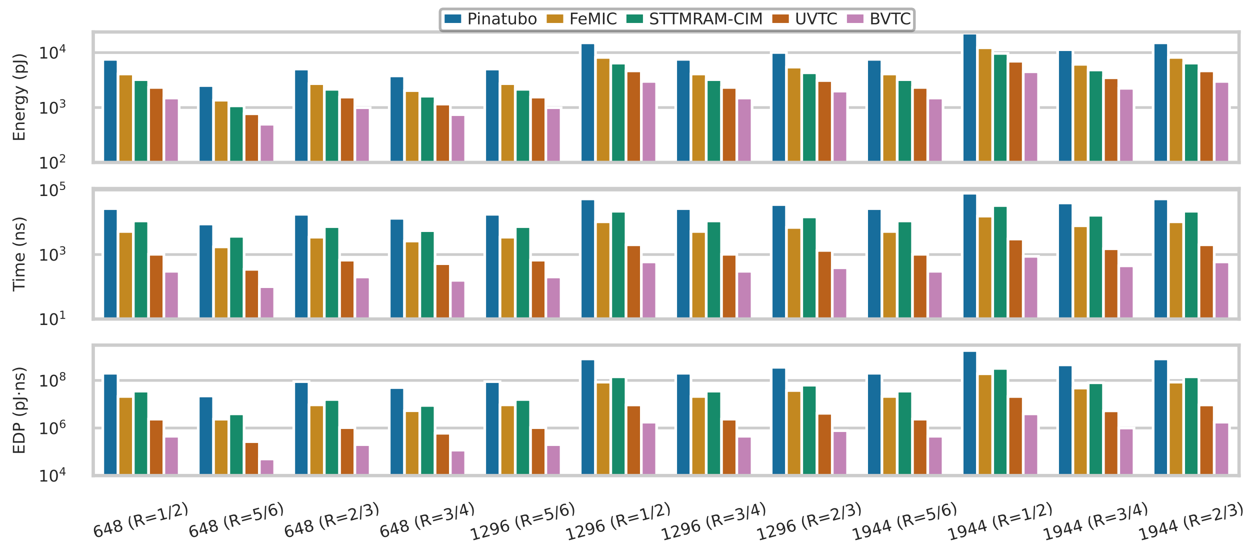


Fig. 11: Energy, Latency, and EDP across 12 IEEE 802.11n LDPC(N,R) configurations and Compute-in-Memory designs.

is obtained analogously from t_{op}. Energy–delay product EDP $= E_{\mathrm{frame}} \cdot t_{\mathrm{frame}}$ summarises overall efficiency. Fig. 11 reports energy, latency and EDP on logarithmic scale. Across all code lengths and rates BVTC delivers 16-18$\times$ lower latency and 2.1-2.2$\times$ lower energy than the best prior solutions, translating up to 49$\times$ EDP reductions UVTC incurs only a 1.6$\times$ energy and 3.4$\times$ latency overhead versus BVTC, yet still outperforms every earlier design by about 9$\times$ in EDP. The ordering of the bars is preserved for every N, R, confirming that the gain is attributable to the in-memory multi-operand XOR primitive rathen than code-specific artefacts.

V. Conclusion and Discussion

This paper presents MOXOR-CIM, which realizes novel VTC-based approaches to perform multi-operand XOR reliably in a single cycle for RRAM-based CIM architectures. In our schemes, CIM generates a voltage signal proportional to the operand data, and two VTC-based approaches are presented to determine if the operand has an ODD or an EVEN parity. Our two proposed schemes offer a desirable trade-off between computing efficiency and area overhead. Comparison results using 512x512 RRAM-based CIM show an improvement of 9-49$\times$ better figure-of-merit combining energy-efficiency and throughput, while also extending the maximum number of operands supported in a single cycle to 16.

Regarding the impact of global resistance drift of RRAM devices, the frequency of the global counter GRCC can be reduced to ensure sufficient sensing margins for the different effective resistance states. In other words, increasing the pulse duration of the GRCC allows additional time for the sensing margin to develop across the bitlines, which compensates for the increased resistance values due to drift. This leads to increased T_{READ} and E_{READ} components while T_{COMP} and E_{COMP} should remain unimpacted. Unfortunately, an increase in device-to-device variation of RRAM devices cannot simply be compensated by reducing the GRCC frequency since these variations increase the spread of T_{READ} corresponding to different effective resistance states. Therefore, it reduces the maximum number of operands supported in a single cycle.

References

[1] A. Sebastian *et al.*, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, 2020.
[2] D. Ielmini *et al.*, "In-memory computing with resistive switching devices," *Nature electronics*, 2018.
[3] L. Wang *et al.*, "Efficient and Robust Nonvolatile Computing-In-Memory Based on Voltage Division in 2T2R RRAM With Input-Dependent Sensing Control," *TCAS II: Express Briefs*, 2021.
[4] Y. Wang *et al.*, "Energy efficient in-memory AES encryption based on nonvolatile domain-wall nanowire," in *DATE*, 2014.
[5] M. Xie *et al.*, "Securing emerging nonvolatile main memory with fast and energy-efficient aes in-memory implementation," *TVLSI*, 2018.
[6] W.-H. Chen *et al.*, "CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors," *Nat. Electronics*, 2019.
[7] I. Giannopoulos *et al.*, "In-Memory Database Query," *Advanced Intelligent Systems*, 2020.
[8] M. Ma *et al.*, "Efficient In-Memory AES Encryption Implementation Using a General Memristive Logic: Surmounting the data movement bottleneck," *Nanotechnology Magazine*, 2022.
[9] J.-M. Hung *et al.*, "Challenges and trends of nonvolatile in-memory-computation circuits for AI edge devices," *Open JSSC Society*, 2021.
[10] A. Gebregiorgis *et al.*, "Dealing with non-idealities in memristor based computation-in-memory designs," in *VLSI-SoC*, 2022.
[11] M. Imani *et al.*, "MPIM: Multi-purpose in-memory processing using configurable resistive memory," in *ASP-DAC*, 2017.
[12] A. Singh *et al.*, "CIM-based robust logic accelerator using 28nm STT-MRAM characterization chip tape-out," in *AICAS*, 2022.
[13] L. Xie *et al.*, "Scouting logic: A novel memristor-based logic design for resistive computing," in *ISVLSI*, 2017.
[14] W.-H. Chen *et al.*, "A 16Mb dual-mode ReRAM macro with sub-14ns computing-in-memory and memory functions enabled by self-write termination scheme," in *IEDM*, 2017.
[15] S. Li *et al.*, "Pinatubo: Processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *DAC*, 2016.
[16] S. Jain *et al.*, "Computing in memory with spin-transfer torque magnetic RAM," *TVLSI*, 2017.
[17] M. Lee *et al.*, "FeFET-based low-power bitwise logic-in-memory with direct write-back and data-adaptive dynamic sensing interface," in *ISLPED*, 2020.
[18] R. Liu *et al.*, "FeMIC: Multi-operands in-memory computing based on FeFETs," in *ASP-DAC*, 2022.
[19] D. Reis *et al.*, "IMCRYPTO: An in-memory computing fabric for AES encryption and decryption," *TVLSI*, 2022.
[20] A. Singh *et al.*, "Referencing-in-array scheme for RRAM-based CIM architecture," in *DATE*, 2022.
[21] M. Bocquet *et al.*, "In-memory and error-immune differential RRAM implementation of binarized deep neural networks," in *IEDM*, 2018.
[22] M. Ezzadeen *et al.*, "Low-overhead implementation of binarized neural networks employing robust 2T2R resistive RAM bridges," in *ESSCIRC*, 2021.
[23] W. Kim *et al.*, "Multistate memristive tantalum oxide devices for ternary arithmetic," *Scientific reports*, 2016.