UltraLiM: In-Memory Boolean Logic Architecture Using UltraRAM

Shamiul Alam[1], Kazi Asifuzzaman[2], and Ahmedullah Aziz[1*]

[1]Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, 37996, USA

[2] Oak Ridge National Laboratory, Oak Ridge, TN, USA

[*]Corresponding Author. Email: aziz@utk.edu.

***Abstract*— Conventional computing architectures encounter 'von Neumann' and 'memory wall' bottlenecks which arise due to the back-and-forth data movement between the physically separate memory and processing units and the speed mismatch between them, respectively. These bottlenecks hurt both energy efficiency and the throughput of computing systems. To address these challenges, in-memory computing architectures have emerged as a promising alternative. They reduce the need for frequent data movement by executing different computing tasks inside the memory system. Here, we present UltraLiM, a logic-in-memory architecture using the UltraRAM-based memory system. UltraRAM holds the promise of developing a 'universal memory', overcoming the limitations of charge-based memories thanks to their non-volatile behavior with lower operating voltage. This work presents an in-memory computing architecture that integrates an UltraRAM-based memory array with a custom-designed peripheral circuitry. With this architecture, we can perform various in-memory Boolean logic operations (such as NOT, NAND, NOR, and XOR) in a single cycle. Leveraging the separate read-write paths in the UltraRAM-based memory array, we optimize read operations without encountering design conflicts. This optimization enhances the sense margin, enabling the use of simpler peripheral circuitry for in-memory logic operations.**

***Keywords*— Boolean logic, in-memory computing, logic-in-memory, UltraRAM.**

I. Introduction

The rise of artificial intelligence, the widespread use of portable smart devices, and the prevalence of social media have led to an unprecedented increase in data volumes. Coping with the storage and processing demands of this massive data presents a formidable challenge for engineers. Furthermore, traditional computing architectures rely on frequent back-and-forth data transfers between memory and processing units, a process known for its high energy consumption [1], [2]. Recent studies by Google indicates that approximately 20-42% of energy is spent on driving the data bus responsible for these transfers [3], [4]. Moreover, the inherent speed mismatch between processing units and memory often leads to processing units idling while awaiting data transactions with memory [5]. In response to these challenges, there is a growing interest in in-memory or near-memory computing approaches for handling data-intensive applications [6]–[8]. The fundamental promise of this approach is to empower the memory system to perform certain computing tasks. By doing so, in-memory computing not only reduces the overhead associated with data movements but also leverages the extensive memory bandwidth to facilitate parallel processing.

To realize an in-memory computing system, it is essential to enable the memory systems to execute the basic Boolean logic operations, which are the foundational elements of any processing unit. Moreover, most of the data-intensive applications rely on simple Boolean logic operations on a massive scale. Additionally, ensuring the storage and security of the ever-increasing data volumes pose significant challenges. Integrity verification, encryption, and decryption of stored data play crucial roles in this regard, often involving Boolean XOR operations between the stored data and relevant keys [9], [10].

In this work, we capitalize on the unique attributes of UltraRAM-based memory system [11] to perform logic-in-memory. To date, charge-based static random-access memory (SRAM), dynamic random-access memory (DRAM), and flash have been the mainstream storage technologies and have dominated the global market [12], [13]. However, none of these memory technologies fulfill the required features for a 'universal memory', capable of serving as both long-term storage and active memory. A '*universal memory*' should feature fast speed, non-volatility, low voltage operation, low energy requirement, high endurance, high retention, and high cost-efficiency to satisfy the memory requirements in different applications [14]. UltraRAM [15] holds promise in achieving all these requirements of becoming a 'universal memory'.

Our work introduces an in-memory computing framework so that we can utilize the unique features of UltraRAM technology. We focus on executing Boolean logic operations, such as NOT, NAND, NOR, and XOR, by integrating the UltraRAM-based memory system with specially designed peripheral circuitry. The key contributions of this work include-

- Developing an in-memory computing system with an UltraRAM-based memory array
- Designing appropriate peripheral circuitry for UltraRAM technology to execute various Boolean logic operations in a single cycle
- Leveraging the separate read-write path capability of UltraRAM-based memory systems to optimize the read operation and achieve improved sense margin
- Developing a HSPICE framework and verifying the functionality of in-memory logic operations.

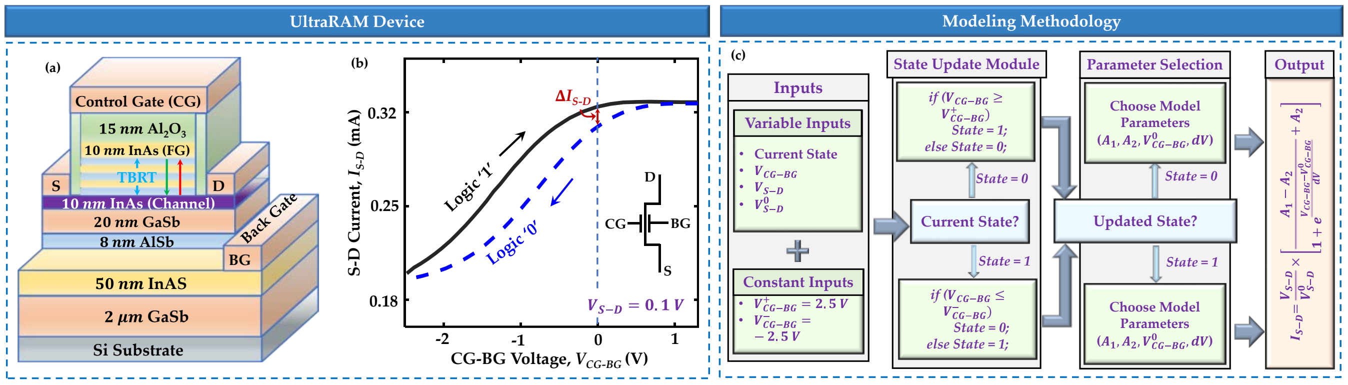


Fig. 1. **(a)** Device structure, **(b)** I-V characteristics (inset shows the symbol), and **(c)** modeling approach of the UltraRAM device.

We start with an overview of UltraRAM technology, followed by a discussion on UltraRAM-based logic-in-memory architecture. Subsequently, we talk about the simulation framework used in this work. Finally, we present the simulated results for both memory read and write, and various in-memory logic operations.

II. UltraRAM Technology

The UltraRAM device incorporates a triple-barrier resonant tunneling (TBRT) structure, employing multiple stacks of InAs quantum wells and AlSb barriers [16]. The schematic of this device is shown in Fig. 1(a). The unique TBRT structure enables UltraRAM to reconcile the conflicting requirements of non-volatility and low energy consumption, essential for a '*universal memory*' solution. In UltraRAM, the storage of logic '0' ('1') is performed by trapping (erasing) electrons into (from) the floating gate (FG). Due to its low voltage switching capability, non-volatile nature, and small capacitance, UltraRAM achieves orders of magnitude lower switching energy than DRAMs and flash memories [16]. Moreover, UltraRAM boasts impressive data retention exceeding 1000 years and endurance surpassing 10^7 cycles [16].

In UltraRAM, an Al_2O_3 layer is used to separate FG from the control gate (CG) and an n-type InAs is used to form the channel of the device. The TBRT structure is formed by stacking multiple thin layers of InAs/AlSb between FG and the channel. In the absence of any bias between CG and BG ($V_{CG-BG} = 0$), the TBRT structure prevents the flow of any electron to FG. However, when a sufficiently strong bias is applied ($V_{CG-BG} \geq 2V$), electrons can be trapped into the FG which is the write '0' operation. Conversely, a bias with opposite polarity ($V_{CG-BG} \leq -2V$) removes the stored electrons from the FG which is the write '1' operation. Therefore, writing the memory state into an UltraRAM device requires only an appropriate bias to be applied between the CG and the BG (Fig. 1(b)).

The *I-V* characteristics of an UltraRAM device is illustrated in Fig. 1(b), showing that even in the absence of any bias between the CG and the BG, a certain source-drain bias (V_{S-D}) can create distinguishable separation between the channel currents of the device in two logic states. Therefore, reading the memory state stored in a cell only needs the application of a suitable V_{S-D} while maintaining $V_{CG-BG} = 0$.

III. UltraLiM: UltraRAM-based Logic-in-Memory

In this work, we utilize the UltraRAM-based memory array reported in [11] to perform in-memory computing. Fig. 2(a) illustrates UltraRAM-based logic-in-memory architecture. In UltraRAM-based memory array, during the write operations in a specific cell, we need to apply biases to the bit lines (BLs) and the write word lines (WWLs) which will result in the application of the write voltage (V_{WRITE}) between the CG and BG contacts of that specific cell. For read operation in any specific cell of the array, we need to apply voltage biases to the read word lines (RWLs) and sense lines (SLs) so that the read voltage (V_{READ}) gets applied between the S and D contacts of the cell. This array benefits from the use of two separate paths for the read and write operations.

During the read operation, the value of V_{READ} dictates the SL current and the separation between the two state currents (ΔI_{S-D}). ΔI_{S-D}, representing the sense margin, directly influences the scalability, energy efficiency, and area efficiency of memory and in-memory computing applications. ΔI_{S-D} can be increased by applying a higher V_{READ}. The separate read-write path capability of UltraRAM-based memory array allows us to optimize the read operation and improve the sense margin without encountering any conflict between the read and write operations.

To execute in-memory computing tasks using the UltraRAM-based memory system, we employ a modified peripheral circuitry [10], shown in Fig. 2(b), in each column of the memory array. This circuitry comprises a current mirror, two current sense amplifiers (CSAs), an inverter, and an AND gate. It takes the SL current of each column as the input and generates a binary output based on the level of SL current and preset reference currents of the two CSAs. Fig. 2(c) presents the schematic of a CSA used in the peripheral circuitry which was adopted from [17], [18]. The advantage of this peripheral circuitry lies in its ability to execute both memory read and various in-memory Boolean logic operations (NOT, NAND, NOR, and XOR) with the same setup. We can achieve this versatility by selecting suitable

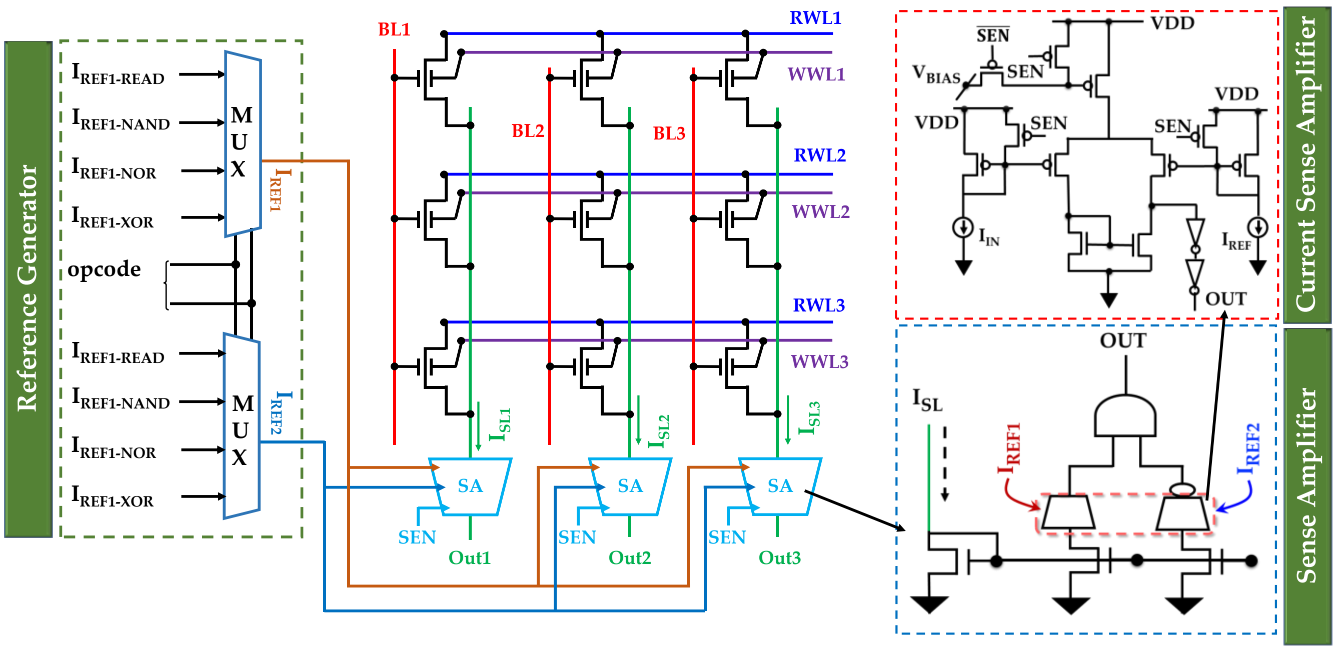


Fig. 2. (a) Schematic of the UltraRAM-based logic-in-memory architecture. Schematics of the **(b)** designed peripheral circuitry and **(c)** a current sense amplifier.

Table I: Truth Table for Logic-in-Memory Operations

x_1	x_2	SL Current (I_{SL})	Logic Operations		
			NAND	NOR	XOR
0	0	$I_{00} = 2 \times I_0$	1	1	0
0	1	$I_{01} = I_0 + I_1$	1	0	1
1	0	$I_{10} = I_1 + I_0$	1	0	1
1	1	$I_{11} = 2 \times I_1$	0	0	0

reference currents for the CSAs. Therefore, we utilize a reference generator to select the suitable reference currents for performing different operations.

In computing mode, the read voltage is applied between the RWLs and SLs, ensuring the application of the necessary voltage between the S and D contacts of specific UltraRAM cells for performing computation on the stored data in those cells. Based on biasing and the memory states stored in the activated cells, different levels of SL current flow in each column. Table I presents the truth table for performing in-memory Boolean logic operations, along with corresponding SL current levels for different combinations of logic states stored in the activated cells. Here, I_0 and I_1 represent the amount of channel current that flows through the UltraRAM

Table II: Conditions for Choosing Reference Currents

Operations	Condition for I_{REF1}	Condition for I_{REF2}
Read	$I_0 < I_{REF1} < I_1$	$I_{REF2} > I_1$
NOT	$I_{REF1} < I_0$	$I_0 < I_{REF2} < I_1$
NAND	$I_{REF1} < I_{00}$	$I_{01} < I_{REF2} < I_{11}$
NOR	$I_{REF1} < I_{00}$	$I_{00} < I_{REF2} < I_{01}$
XOR	$I_{00} < I_{REF1} < I_{01}$	$I_{01} < I_{REF2} < I_{11}$

cell storing logic ‘0’ and ‘1’ states, respectively. Now, to reconfigurably perform both memory read and logic-in-memory operations with the same peripheral circuitry, we need to choose the suitable reference currents (I_{REF1} and I_{REF2}) for the CSAs. Table II outlines the conditions for selecting suitable values of I_{REF1} and I_{REF2} to execute memory read-write and various logic operations.

During memory read and NOT operations, only one cell is activated while NAND, NOR, and XOR operations require activation of two cells. The current contribution of activated cells is fed into the peripheral circuitry which sets the gate voltage of a specific transistor in each CSAs. Comparison of these voltages with reference voltages produces binary outputs. Among these operations, only one CSA is sufficient to execute memory read, NOT, NAND, and NOR operations. Therefore, we choose I_{REF1} and I_{REF2} in a way so that one of the CSAs always provides logic ‘1’ output and the other CSA provides the output of the intended operation. Finally, after performing the AND operation between the outputs of the two CSAs, the peripheral circuitry generates the final output for the performed operation. To perform the XOR operation in single cycle, we utilize the complete capability of the peripheral circuitry. Here, two complementary reference currents are chosen for the two CSAs to separate out the logic ‘01’ and ‘10’ combinations from the others.

IV. Simulation Framework

To simulate the UltraRAM-based in-memory logic operations, we develop a simulation framework in HSPICE. To accurately replicate the characteristics of UltraRAM devices, we utilize a Verilog-A-based compact model from [11]. Fig. 1(c) provides an overview of the modeling approach. The model utilizes a modified version of the

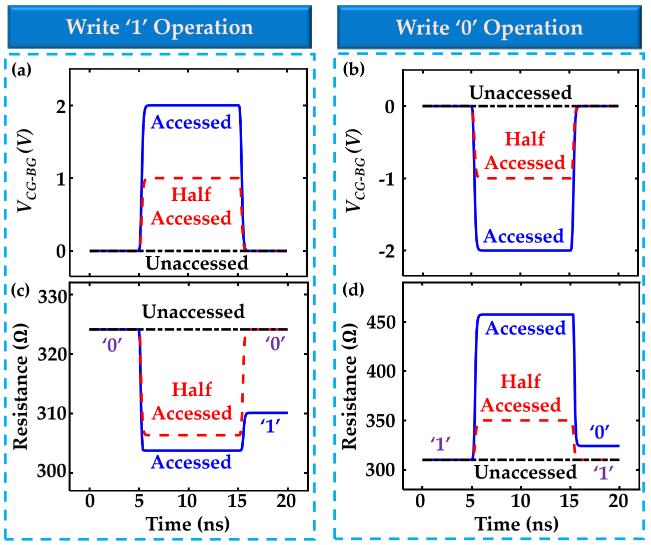


Fig. 3. Transient dynamics of voltage between the CG and the BG contacts of UltraRAM cells in the array during **(a)** write '1' and **(b)** write '0' operations. Switching of the channel resistance of UltraRAM devices during **(c)** write '1' and **(d)** write '0' operations.

Boltzmann growth function to mimic the I-V characteristics of UltraRAM devices. The function is as follows-

$$I_{S-D}(V_{CG-BG}, V_{S-D}) = \frac{V_{S-D}}{V_{S-D}^{0}} \times \left[\frac{A_1 - A_2}{1+e^{\frac{V_{CG-BG}-V_{CG-BG}^{0}}{dx}}} + A_2\right] \quad (1)$$

Here, A_1 and A_2 are fitting parameters. The values of different parameters of equation (1) for two states of the device are outlined in Table III. The model was calibrated with the experimental I-V characteristics of UltraRAM device reported in [16]. For simulating the CMOS transistors (FinFETs) used in the peripheral circuitry, we adopt 14 nm PTM (Predictive Technology Model) [19] transistors.

Table III: Values of Different Model Parameters.

Parameters	State = 1	State = 0
A_1	0.19196	0.19259
A_2	0.32938	0.32979
V_{CG-BG}^{0}	-1.35027	-0.81267
dV	0.45746	0.47148

V. Simulation Results

Using our developed simulation framework, we first showcase simulated results for memory write operations of the UltraRAM-based memory system, focusing on a 3×3 array. To perform various in-memory computing operations, we need to program the memory array by writing specific memory state into specific cells of the array. Therefore, we demonstrate how to perform the write operation into a specific cell (1, 1) in Fig. 3. For writing into a specific cell, we adopt the $V/2$ biasing scheme [20] for the BLs and WWLs. Specifically, to write into the (1, 1) cell, we apply V_{WRITE} ($\pm 2.3\ V$) to BL_1 and $\frac{V_{WRITE}}{2}$ ($\pm 1.15\ V$) to other BLs. Additionally, we apply 0 V to WWL_1 and $\frac{V_{WRITE}}{2}$ to other WWLs. As shown in Fig. 3(a) and (b), this biasing scheme ensures that the (1, 1) cell receives V_{WRITE}, while the half-accessed cells (those in the same row or same column as the accessed cell) receive $\frac{V_{WRITE}}{2}$ (insufficient to alter their states), and unaccessed cells receive 0 V between the CG and BG contacts. It is noteworthy that before applying biases for write '0' ('1') operations, all cells are initialized to logic '1' (logic '0') state. To verify successful write operations, we apply a V_{S-D} of 0.2 V via RWLs and SLs (not necessary for the write operation). Fig. 4(c) and (d) show switching of the channel resistance during the two write operations. For the

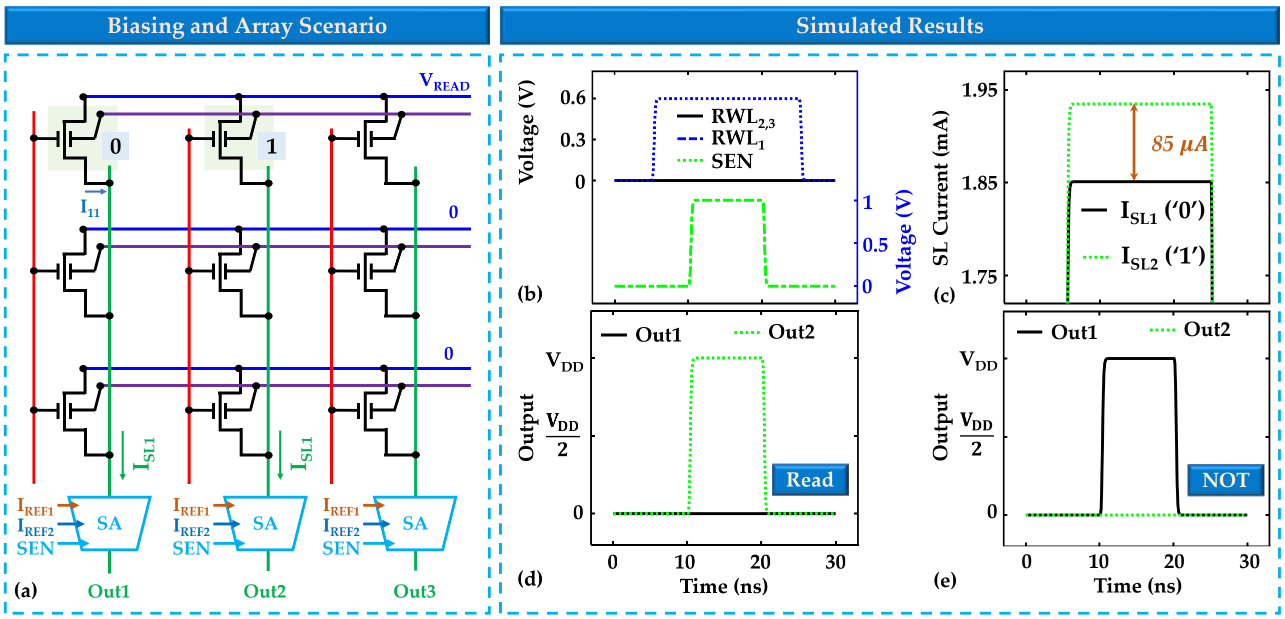
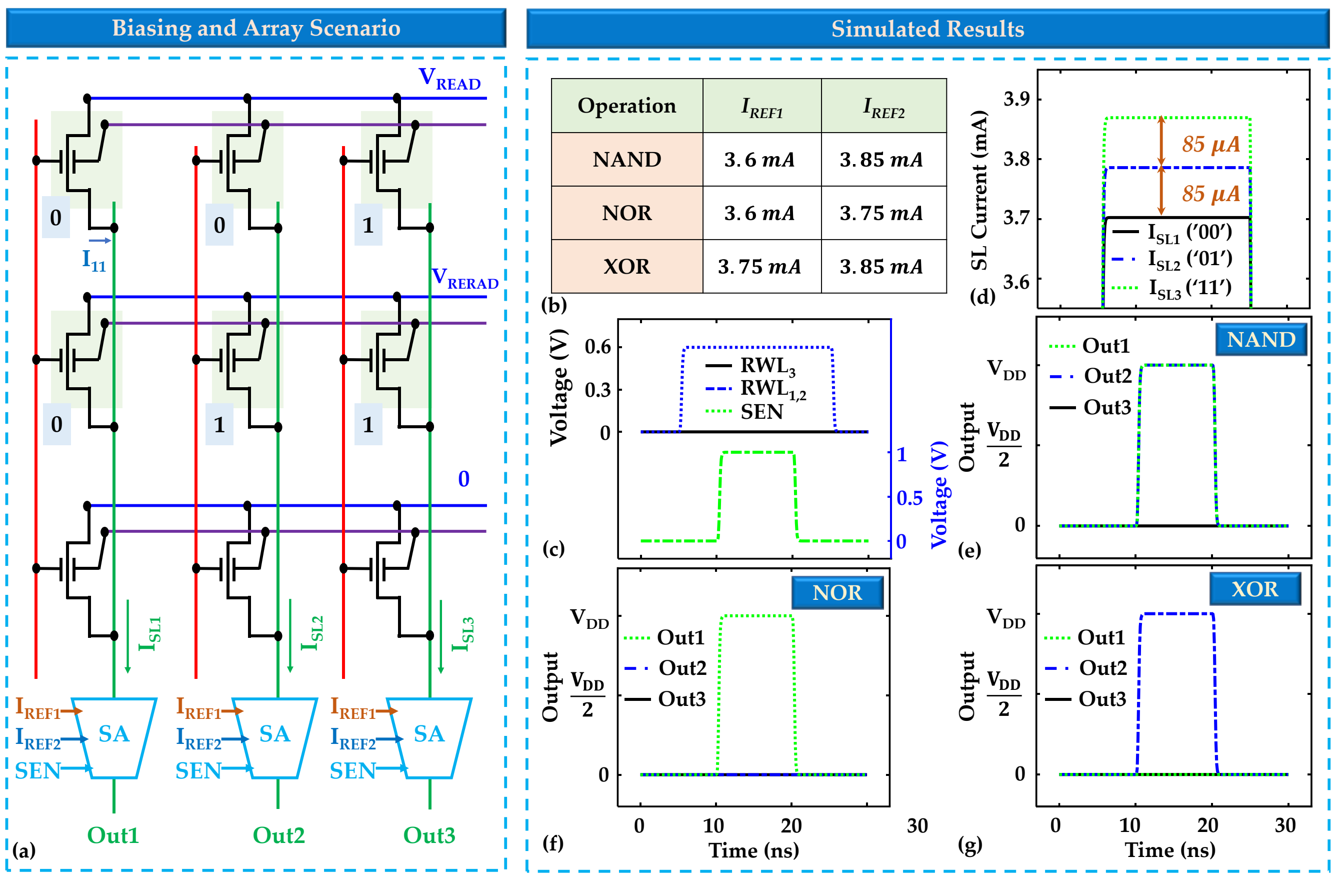


Fig. 4. **(a)** Programming of the memory array to execute memory read and in-memory NOT operations. **(b)** Voltage biases applied to RWLs and SEN to activate the required cells. **(c)** SL current levels corresponding to the stored states in the activated cells. Outputs of the peripheral circuitry during **(d)** memory read and **(e)** in-memory NOT operations from the first and second columns, respectively.



Operation	I_{REF1}	I_{REF2}
NAND	$3.6\ mA$	$3.85\ mA$
NOR	$3.6\ mA$	$3.75\ mA$
XOR	$3.75\ mA$	$3.85\ mA$

Fig. 5. **(a)** Programming of the memory array and **(b)** the selected values of the reference currents for the two CSAs in the peripheral circuitry to execute in-memory NAND, NOR, and XOR operations. **(c)** Voltage biases applied to RWLs and SEN to activate the required cells. **(d)** SL current levels corresponding to the stored states in the activated cells. Outputs of the peripheral circuitry during in-memory **(e)** NAND, **(f)** NOR, and **(g)** XOR operations.

same V_{S-D}, the cell storing logic '1' shows lower channel resistance than the cell storing logic '0'.

We proceed to demonstrate the simulation of in-memory Boolean logic operations. First, we simulate memory read and in-memory NOT operations utilizing the designed peripheral circuitry shown in Fig. 2. To demonstrate the read and NOT operations, we activate (1, 1) and (2, 1) cells in the array and tore logic '0' and '1' into these cells, respectively (Fig. 4(a)). To activate these cells, we apply a V_{READ} of 0.6 V with the help of RWLs and SLs (Fig. 4(a) and (b)). This results in two distinct levels of current in the corresponding SLs (SL_1 and SL_2) for the two memory states (Fig. 4(c)). To perform both memory and computing operations with the same peripheral circuitry, careful selection of I_{REF1} and I_{REF2} is necessary. For memory read and in-memory NOT operations, we choose $I_{REF1} = 1.9\,mA$ and $I_{REF2} = 2\,mA$, and $I_{REF1} = 1.8\,mA$ and $I_{REF2} = 1.9\,mA$, respectively. Upon activating the sense enable (SEN) signal of the CSAs in the peripheral circuitry, each CSA generates logic '0' or '1' output based on the comparison between the SL current and the set reference current. Using the outputs of the two CSAs, the peripheral circuitry generates the final output for the performed operation as shown in Fig. 4(d) and (e). Here, the peripheral circuitry of first and second columns (Out1 and Out2, respectively) generates the final output of these operations for logic '0' and '1' state, respectively.

Next, we simulate the 2-input logic operations such as NAND, NOR, and XOR by activating the first two rows in the array through the application of appropriate biases to the RWLs (Fig. 5(a)). Here, we program the cells in the activated rows in a way so that the first, second, and third columns correspond to logic '00', '01' or '10', and '11' conditions, respectively. The required biasing condition to activate the rows is shown in Fig. 5(c). Now, due to the biasing and programing of the memory array, we get different levels of current in the corresponding SLs (SL_1, SL_2, SL_3) for different combinations of logic states ('00', '01' or '10', and '11', respectively) of the activated cells (Fig. 5(d)). A sense margin of 85 μA is obtained between the adjacent current levels. Then, again to configure the peripheral circuitry for different logic operations, we need to choose suitable values for I_{REF1} and I_{REF2}. Fig. 5(b) outlines the values of I_{REF1} and I_{REF2} used in this work for executing different logic operations. When SEN is enabled, the peripheral circuitry generates the final binary output for the executed operations with the help of two CSAs (Fig. 5(e)-(g)). Here, for the logic operations, the peripheral circuitry of the first three columns show the outputs corresponding to different logic combinations of the activated cells.

VI. Conclusion

In this work, we introduce an innovative in-memory computing architecture tailored for UltraRAM-based memory technology. Central to our framework is the development of a novel peripheral circuitry capable of dynamically executing memory read operation and a spectrum of in-memory Boolean logic operations, including NOT, NAND, NOR, and XOR. This circuitry's adaptability enables seamless transition between memory-centric tasks and computational operations, all within a single cycle. A key advantage of our approach lies in its ability to unify memory and logic functionalities, streamlining computational processes and reducing overall complexity. Furthermore, we demonstrate how the distinct read-write path feature inherent in UltraRAM devices can be harnessed to optimize read operations, thereby enhancing the efficacy of in-memory computing tasks. Through this integrated framework, we unlock the full potential of UltraRAM technology, paving the way for efficient and versatile computing paradigms that transcend traditional memory architectures.

Acknowledgement

This research was supported in part by seed funding from the AI Tennessee Initiative at the University of Tennessee, Knoxville.

S. A. was supported with funds provided by the Science Alliance, a Tennessee Higher Education Commission center of excellence administered by The University of Tennessee-Oak Ridge Innovation Institute on behalf of The University of Tennessee, Knoxville.

This manuscript has also been authored in part by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of the manuscript or allow others to do so, for U.S. Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

References

[1] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nat. Nanotechnol. 2020 157*, vol. 15, no. 7, pp. 529–544, Mar. 2020, doi: 10.1038/s41565-020-0655-z.

[2] S. Alam, M. M. Islam, M. S. Hossain, A. Jaiswal, and A. Aziz, "CryoCiM: Cryogenic compute-in-memory based on the quantum anomalous Hall effect," *Appl. Phys. Lett.*, vol. 120, no. 14, p. 144102, Apr. 2022, doi: 10.1063/5.0092169.

[3] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G.-Y. Wei, and D. Brooks, "Profiling a warehouse-scale computer," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, 2015, pp. 158–169, doi: 10.1145/2749469.2750392.

[4] A. Boroumand, S. Ghose, Y. Kim, R. Ausavarungnirun, E. Shiu, R. Thakur, D. Kim, A. Kuusela, A. Knies, P. Ranganathan, and O. Mutlu, "Google Workloads for Consumer Devices: Mi-tigating Data Movement Bottlenecks," *Proc. Twenty-Third Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, vol. 18, 2018, doi: 10.1145/3173162.

[5] M. Lee, W. Tang, B. Xue, J. Wu, M. Ma, Y. Wang, Y. Liu, D. Fan, V. Narayanan, H. Yang, and X. Li, "FeFET-Based Low-Power Bitwise Logic-in-Memory with Direct Write-Back and Data-Adaptive Dynamic Sensing Interface," *Proc. ACM/IEEE Int. Symp. Low Power Electron. Des.*, 2020, doi: 10.1145/3370748.

[6] W. Kang, H. Wang, Z. Wang, Y. Zhang, and W. Zhao, "In-Memory Processing Paradigm for Bitwise Logic Operations in STT-MRAM," *IEEE Trans. Magn.*, vol. 53, no. 11, Nov. 2017, doi: 10.1109/TMAG.2017.2703863.

[7] S. Alam, J. Hutchins, M. S. Hossain, K. Ni, V. Narayanan, and A. Aziz, "Cryogenic In-Memory Matrix-Vector Multiplication using Ferroelectric Superconducting Quantum Interference Device (FE-SQUID)," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6, doi: 10.1109/DAC56929.2023.10247669.

[8] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nat. Electron. 2018 16*, vol. 1, no. 6, pp. 333–343, Jun. 2018, doi: 10.1038/s41928-018-0092-2.

[9] X. Xin, Y. Zhang, and J. Yang, "Reducing DRAM access latency via helper rows," *Proc. - Des. Autom. Conf.*, vol. 2020-July, Jul. 2020, doi: 10.1109/DAC18072.2020.9218719.

[10] S. Alam, J. Hutchins, N. Shukla, K. Asifuzzaman, and A. Aziz, "CMOS-based Single-Cycle In-Memory XOR/XNOR," *ArXiv Prepr.*, Oct. 2023.

[11] S. Alam, K. Asifuzzaman, and A. Aziz, "A Novel Scalable Array Design for III-V Compound Semiconductor-based Nonvolatile Memory (UltraRAM) with Separate Read-Write Paths," *Proc. - Int. Symp. Qual. Electron. Des. ISQED*, vol. 2023-April, 2023, doi: 10.1109/ISQED57927.2023.10129314.

[12] H. S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nat. Nanotechnol. 2015 103*, vol. 10, no. 3, pp. 191–194, Mar. 2015, doi: 10.1038/nnano.2015.29.

[13] Z. Shen, S. Srinivasa, A. Aziz, S. Datta, V. Narayanan, and S. K. Gupta, "SRAMs and DRAMs with separate read-write ports augmented by phase transition materials," *IEEE Trans. Electron Devices*, vol. 66, no. 2, pp. 929–937, Feb. 2019, doi: 10.1109/TED.2018.2888913.

[14] J. Åkerman, "Toward a universal memory," *Science (80-.).*, vol. 308, no. 5721, pp. 508–510, Apr. 2005, doi: 10.1126/SCIENCE.1110549/ASSET/E79FD10E-DE66-4D47-90D3-7D61F65F68F1/ASSETS/GRAPHIC/508-1.GIF.

[15] M. Hayne, "Electronic memory devices," US10243086B2, 2019.

[16] P. D. Hodgson, D. Lane, P. J. Carrington, E. Delli, R. Beanland, M. Hayne, P. D. Hodgson, D. Lane, M. Hayne, P. J. Carrington, and E. Delli, "ULTRARAM: A Low-Energy, High-Endurance, Compound-Semiconductor Memory on Silicon," *Adv. Electron. Mater.*, vol. 8, no. 4, p. 2101103, Apr. 2022, doi: 10.1002/AELM.202101103.

[17] M. F. Chang, S. J. Shen, C. C. Liu, C. W. Wu, Y. F. Lin, Y. C. King, C. J. Lin, H. J. Liao, Y. Der Chih, and H. Yamauchi, "An offset-tolerant fast-random-read current-sampling-based sense amplifier for small-cell-current nonvolatile memory," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 864–877, 2013, doi: 10.1109/JSSC.2012.2235013.

[18] M. M. Islam, S. Alam, M. A. Jahangir, G. S. Rose, S. Datta, V. Narayanan, S. K. Gupta, and A. Aziz, "Reimagining Sense Amplifiers: Harnessing Phase Transition Materials for Current and Voltage Sensing," *ArXiV Prepr.*, Aug. 2023, doi: 10.48550/arXiv.2308.15756.

[19] "Arizona State University Predictive Technology Models." [Online]. Available: http://ptm.asu.edu/.

[20] S. Alam, M. S. Hossain, and A. Aziz, "A non-volatile cryogenic random-access memory based on the quantum anomalous Hall effect," *Sci. Rep.*, 2021, doi: 10.1038/s41598-021-87056-7.