Cryogenic Logic-In-Memory Bit-Serial Addition With Majority Gates Based On The Quantum Anomalous Hall Effect

Smitha Sunil[1] , Dr. D. S. Shylu Sam[2]

[1]PhD Scholar Department of Electronics and Communication Engineering Karunya Institute of Science and Technology Karunya University, Coimbatore, (India)
smithasunil@karunya.edu.in
[2]Associate Professor Department of Electronics and Communication Engineering Karunya Institute of Science and Technology Karunya University, Coimbatore, (India)
shylusam@karunya.edu

Index Terms

– Boolean function, majority expression, quantum cellular automata (QCA), cryogenic, in-memory computing, quantum anomalous hall effect (QAHE), memristive logic, non-volatile memory (NVM), Cryogenic memory.

Abstract:

As CMOS technology approaches its physical limits, new alternatives are being investigated to improve computational performance and energy efficiency. Cryogenic computing offers a viable solution, enhancing computing speeds without the need for further scaling. However, the issue of the "memory wall" continues to challenge traditional von-Neumann architectures, even under cryogenic conditions. To address this, logic-in-memory Computing architectures, which enable computation within the memory unit, provide benefits by minimizing data transfer between the memory and processing units. This reduces energy consumption and cooling costs at low temperatures. In this study, we introduce CryoLiMC, a cryogenic logic-in-memory Computing framework using a non-volatile memory system based on the quantum anomalous Hall effect (QAHE). This memory system enhances energy efficiency, resilience to process variations, and scalability. With Moore's Law slowing down, alternative technologies such as memristive devices and resistance-switching memories show promise in overcoming the memory wall by enabling computation within memory arrays. Additionally, the potential of majority logic as a Boolean logic primitive for in-memory computing is explored, particularly in the context of memristive systems. The paper also examines the use of majority gates in quantum cellular automata (QCA), offering methods to simplify Boolean functions and minimize hardware requirements in QCA designs.

INTRODUCTION

The growing interest in quantum computing, space electronics, and superconducting circuits has driven significant advancements in cryogenic data storage technologies. Cryogenic in-memory computing offers remarkable energy efficiency, addressing the memory bottleneck in both classical and quantum computing while mitigating cooling challenges in cryogenic systems. Beyond traditional computing, cryogenic memory plays a crucial role in the development of quantum computers, which have the

potential to tackle complex scientific and commercial problems that are infeasible for classical systems to solve within a reasonable timeframe. Quantum computers can expedite tasks involving number theory, algebra, and optimization [1]. Nevertheless, the absence of an effective cryogenic memory solution remains a significant hurdle in building scalable quantum computing systems. This review presents a novel approach to enhancing computational efficiency at cryogenic temperatures by integrating logic operations directly within memory units. This method leverages the Quantum Anomalous Hall Effect (QAHE) to implement majority logic gates, facilitating bit-serial addition in a cryogenic environment [2].

For energy-efficient operations, compute-in-memory (CiM) architectures have been investigated for both cryogenic and ambient temperatures. These architectures perform computation directly within memory arrays, utilizing volatile memory like SRAMs [3,4,5] and DRAMs [4,6,7], as well as non-volatile options such as RRAMs [8.9.10], MRAMs [11,12,13,14], FeRAMs [15,16,17], and PCMs [18,19]. The CryoCiM platform, leveraging the Quantum Anomalous Hall Effect, eliminates the need for energy-intensive data transfers between processing and memory units, significantly enhancing energy efficiency.

Majority logic, a form of Boolean logic, is considered an effective logic primitive due to its strong expressive capabilities. This review evaluates the efficiency of majority logic in the context of in-memory computing. Traditional CMOS technology supports diverse logic gates like NAND, NOR, and XOR, whereas in-memory computing often relies on a single type of gate (e.g., only IMPLY, NAND, or MAJORITY). Given this, memristive logic systems capable of implementing MAJORITY and NOT gates (ensuring functional completeness) are preferable for in-memory designs. A comparison of one-bit full adders created with various logic primitives highlights the advantages of majority-based designs. Furthermore, to explore whether this efficiency scales, eight-bit adders using different logic primitives are analyzed within memory arrays [20].

Need for Cryogenic Memory

Cryogenic memories play a vital role in quantum technologies. A standard quantum computer typically consists of three key components: quantum qubits, a control processor, and a memory block shown in fig.1.

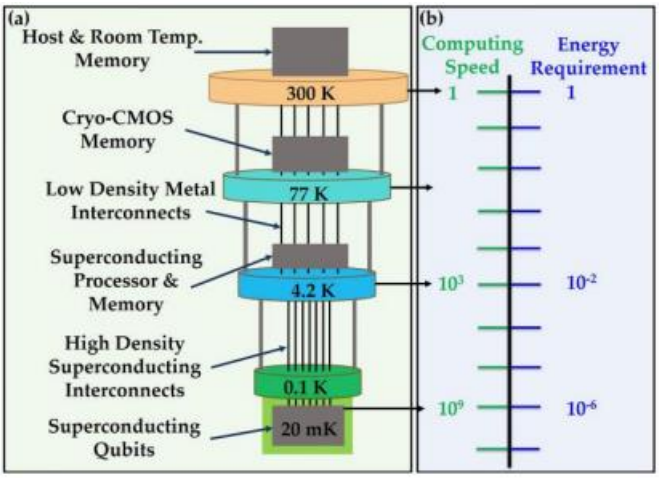


Fig. 1: (a) Quantum Computing Hardware Stack with Cryogenic Layers
(b) Computing Speed vs. Energy Requirement

Quantum Computing Hardware Stack with Cryogenic Layers:

This diagram illustrates a typical quantum computing system organized across multiple temperature stages, each with distinct components:

- **300 K (Room Temperature)**: This is where the classical control hardware and host memory reside, interfacing with the quantum processor through digital-to-analog converters and signal processing units [21].
- **77 K (Liquid Nitrogen Temperature)**: Cryogenic CMOS (Complementary Metal-Oxide-Semiconductor) memory and low-density interconnects, which help reduce thermal noise while still enabling communication with room-temperature electronics [22].
- **4.2 K (Liquid Helium Temperature)**: Superconducting processors and memory operate here. At this stage, superconducting materials can carry signals without resistance, minimizing energy loss [23].
- **0.1 K & 20 mK (Millikelvin Range)**: The superconducting qubits operate in dilution refrigerators at ultra-low temperatures to maintain quantum coherence. These conditions suppress thermal noise and extend qubit lifetimes [24].

Computing Speed vs. Energy Requirement:

This panel shows the trade-off between computing speed and energy requirements across temperature stages:

- **Computing Speed (Green)**: Increases exponentially as we move to lower temperatures, reaching up to a billion times faster operations near the qubits.
- **Energy Requirement (Blue)**: Decreases significantly at lower temperatures, highlighting the efficiency of quantum operations compared to classical systems.

The figure emphasizes how quantum systems leverage extreme cryogenic cooling to enable high-speed, low-power computation — essential for scaling up quantum processors [25].

QAHE-based memory system

A typical structure of a memory cell connected in series with a selector device, commonly used in resistive random-access memory (RRAM) or other emerging non-volatile memory technologies shown in fig.2.

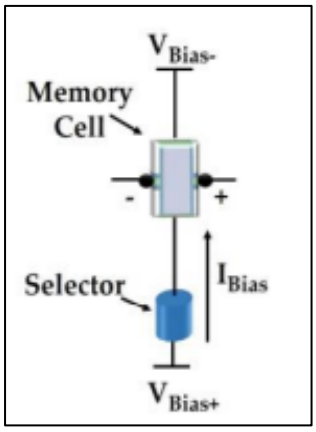


Figure (2): One cell structure of the QAHE-based memory system, contains a QAHE-base memory cell and a mixed-ionic-electronic-conduction (MIEC) material-based selector.

- **Memory Cell:** The top component represents the memory cell, which stores data as different resistance states. In RRAM, for instance, this could be a metal-oxide layer that switches between high-resistance and low-resistance states depending on the applied voltage. These resistance states encode binary data (0s and 1s) [26].

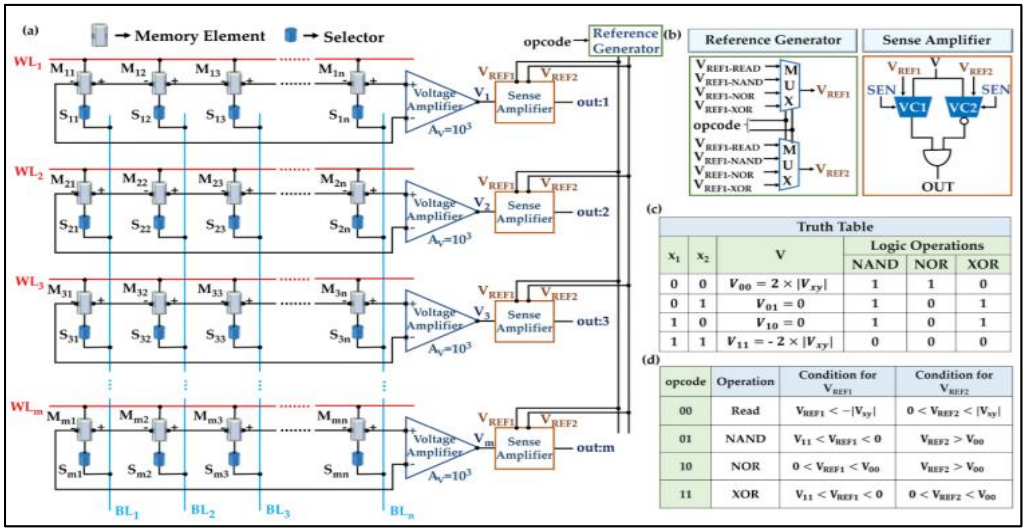
- **Selector:** Below the memory cell is a selector device, which is used to control access to the memory cell. This prevents sneak-path currents in crossbar arrays, improving read and write accuracy. Selectors can be diodes, transistors, or even two-terminal devices like threshold switching elements [27].

- **Bias Voltages (V_{Bias+} and V_{Bias-}):** Bias voltages are applied across the memory cell and selector to program or read the memory state. V_{Bias+} and V_{Bias-} control the voltage across the stack, and the resulting current (I_{Bias})indicates the cell's resistance state.

- **Current Flow (I_{Bias}):** The bias current flows through both the selector and memory cell, allowing the resistance state to be determined during a read operation or modified during a write operation.

Cryogenic CiM based on QAHE

In the memory array, to activate a cell for read and logic operations, a suitable voltage is applied between the bit-line (BL) and word-line (WL) which creates the read current through the corresponding memory cell is shown in Fig.3.



x_1	x_2	V	Logic Operations NAND	NOR	XOR
0	0	$V_{00} = 2 \times \lvert V_{xy}\rvert$	1	1	0
0	1	$V_{01} = 0$	1	0	1
1	0	$V_{10} = 0$	1	0	1
1	1	$V_{11} = -2 \times \lvert V_{xy}\rvert$	0	0	0

opcode	Operation	Condition for V_{REF1}	Condition for V_{REF2}
00	Read	$V_{REF1} < -\lvert V_{xy}\rvert$	$0 < V_{REF2} < \lvert V_{xy}\rvert$
01	NAND	$V_{11} < V_{REF1} < 0$	$V_{REF2} > V_{00}$
10	NOR	$0 < V_{REF1} < V_{00}$	$V_{REF2} > V_{00}$
11	XOR	$V_{11} < V_{REF1} < 0$	$0 < V_{REF2} < V_{00}$

Figure (3): (a) QAHE-based cryogenic CiM structure. (b) Schematics of the reference generator and the sense amplifier. (c) Truth table for the 2-input logic operations and the corresponding voltage levels for different logic combinations. (d) Conditions for choosing the reference voltages for read and in-memory logic operations such as NAND, NOR, and XOR.

(a) Crossbar Memory Array and Sense Amplifiers: The left part of the figure shows a crossbar memory array, where each memory cell consists of a memory element (M) and a selector device (S). Word lines (WL) and bit lines (BL) control the access to each cell. When a word line is activated, the voltage of the corresponding bit line is amplified and sensed by a voltage amplifier (gain $A_V = 10^3$) before being sent to a sense amplifier.

This structure is similar to resistive RAM (RRAM)-based architectures, where the resistance state of a memory cell determines its stored value, and selectors prevent sneak-path currents, a common issue in crossbar designs [28].

(b) Reference Generator and Sense Amplifier Design: The reference generator produces different reference voltages depending on the opcode:

- **Read**: V_{REF1} and V_{REF2} are set for reading the state of the cell.
- **Logic Operations** (NAND, NOR, XOR): Different voltage conditions are generated to execute the desired logic directly in-memory.

The sense amplifier (right side of (b)) compares the input voltage with the reference voltage using two voltage comparators (VC1 and VC2) and generates the output logic value.

Such designs are inspired by state-of-the-art processing-in-memory (PIM) techniques, which aim to reduce data movement and enhance energy efficiency [29].

(c) Truth Table and Logic Operations: The truth table shows how logic operations are implemented using voltage levels:

- **NAND**, **NOR**, and **XOR** are realized by mapping the voltage combinations of two input cells directly to logic results.
- For example, for NAND, V00=2×V[x1∨x2] determines the result.

This approach is akin to prior work in memristor-based logic-in-memory architectures [30].

(d) Opcode and Voltage Conditions: The opcode dictates the operation:

- **Read**: Detect the cell state by comparing the sensed voltage against reference voltages.
- **Logic operations**: Apply specific conditions on reference voltages (V_{REF1} and V_{REF2}) to distinguish between different logic states.

The idea of using voltage-based comparisons for logic operations is closely related to threshold logic design in non-volatile memory devices [31].

Compute-in-memory Majority logic in QAHE-based memory

Majority logic is a fundamental concept in Boolean logic, where the output is '1' if the majority of inputs are '1'. Figure 4(a) illustrates the symbols and truth tables for 3-input and 5-input Majority gates. The Boolean expression for a 3-input Majority gate is given by:

$$\textbf{Maj (a, b, c) = (a \& b) | (b \& c) | (c \& a)} \qquad \textbf{(1)}$$

Here, 'a', 'b', and 'c' represent the inputs. The concept of Majority logic emerged in the 1960s, and since then, various advanced technologies have been explored to implement it. These include spin waves, single-electron tunnelling, quantum dot cellular automata, nanomagnetic logic, and superconducting Josephson junctions.

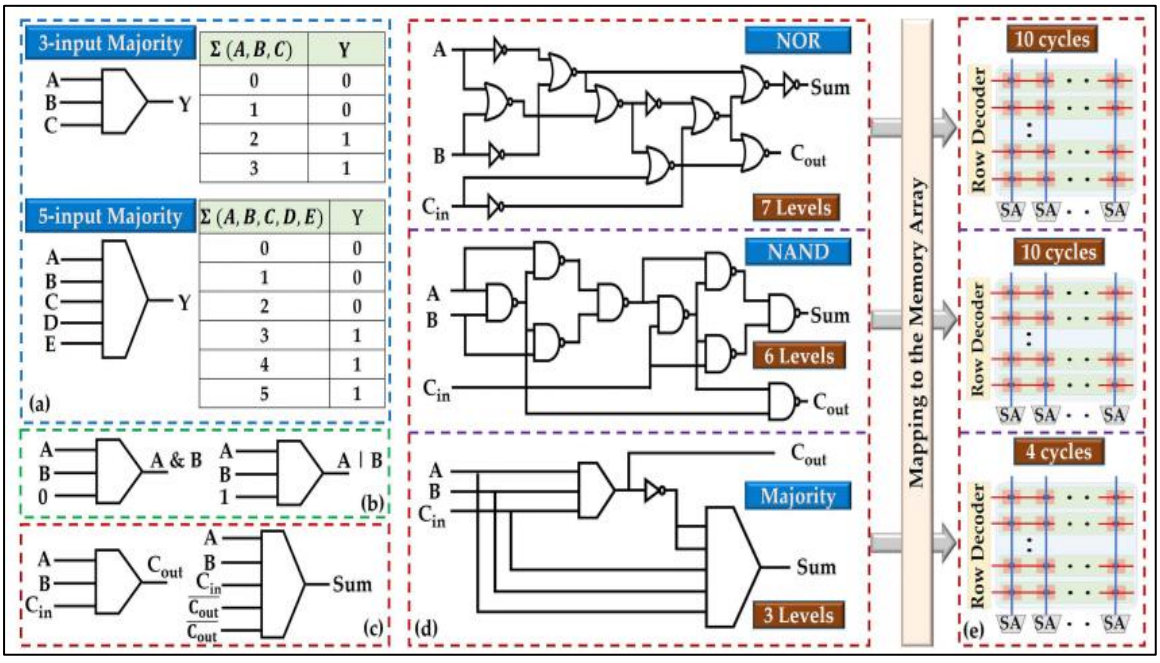


Figure (4): (a) Truth tables for 3-input and 5-input Majority logic. Implementation of (b) AND and OR gates, and (c) full adder using Majority logic. (d) Logic level and (e) in-memory implementation of full adder using NAND, NOR, and Majority. The use of Majority logic reduces the logical depth and number of cycles.

The benefits of Majority logic include the ability to construct other logic gates like AND and OR (Fig. 4(b)), and in some technologies, such as quantum dot cellular automata, it offers a more straightforward and efficient implementation compared to gates like NAND, NOR, and XOR. This logic is particularly advantageous for arithmetic-intensive systems, as it reduces the number of gates and lowers logical depth by up to 33% compared to AND-based designs [32,33,34].

In Compute-in-Memory (CiM) architectures, typically a single logic primitive (like NAND or NOR) is implemented, and other functions are built from that. However, this approach increases the number of logic levels. For instance, creating a XOR gate with NAND logic requires a sequence of four operations. In contrast, a 1-bit full adder can be designed with just one 3-input and one 5-input Majority gate (Fig. 4(c)), using the simplified logic expressions:

$$\text{Sum} = \text{Maj}(A, B, \text{Cin}, \text{Cout}, \text{Cout})$$

and

$$\text{Cout} = \text{Maj}(A, B, \text{Cin}) \quad (2)$$

Figure 4(d) demonstrates that this Majority gate-based approach reduces logic levels by 50% (or 57.14% for NOR) compared to NAND/NOR implementations [34]. When applied to in-memory systems, Majority logic reduces the required cycles by up to 60% compared to NAND/NOR-based full adder designs (Fig. 4(e)) [34].

This work introduces a cryogenic in-memory Majority logic, which is used to create a full adder — a critical component in processors for quantum computers, cryogenic systems, and spacecraft control units [35]. We leverage a quantum anomalous Hall effect (QAHE)-based memory array [36] to achieve this.

QAHE arrays have two useful properties for Majority logic:

(i) during read operations, cells produce a positive (or negative) Hall voltage for logic '0' ('1'), and
(ii) the voltages of all cells in a row are algebraically summed. These features result in a direct Majority output, simplifying peripheral circuit design and enhancing system efficiency.

TABLE 1. Implementation of 3-input and 5-input in-memory Majority logic in QAHE-based memory.

	∑ Inputs	V_R (mV)	V_{out} (logic)	
3-input	0	150	0	$V_R > 0$ / $V_R < 0$
	1	50	0	
	2	-50	1	
	3	-150	1	
5-input	0	250	0	$V_{out} = \begin{cases} 0, \\ 1, \end{cases}$
	1	150	0	
	2	50	0	
	3	-50	1	
	4	-150	1	
	5	-250	1	

In a QAHE-based memory system, the Majority function can be realized by detecting voltage levels when an odd number of cells are activated in a row. These voltage levels, which are either positive or negative, represent the two logical states ('1' and '0'). Table 1 outlines the truth table and corresponding voltage levels for 3-input and 5-input Majority operations. This approach simplifies the implementation, requiring only a basic voltage comparator during the read process. The QAHE memory system offers unique benefits: (i) distinct voltage levels with opposite polarities for the two memory states and (ii) the cumulative high voltage generated by all cells in a row. Notably, this voltage-level distinction remains consistent regardless of the number of inputs (e.g., 3, 5, or 7), as illustrated in Table 1.

Memristive Logic

Memristors belong to a new category of Non-Volatile Memory (NVM) devices that retain data by changing resistance. When subjected to voltage or current, their resistance transitions between a Low Resistance State (LRS) and a High Resistance State (HRS). The term 'memristor' originates from the combination of 'memory' and 'resistor,' highlighting its ability to store information through resistance changes.

Two widely used configurations for constructing a memory array with such devices are the 1Transistor-1 Resistor (1T–1R) and the 1 Selector-1 Resistor (1S–1R), as depicted in Figure 5a. The 1T–1R configuration incorporates a transistor as an access device for each memory cell, enabling selective access to individual cells without disrupting adjacent ones in the array [37,38]. On the other hand, the 1S–1R configuration employs a two-terminal component known as a 'selector,' which exhibits diode-like behavior. This selector is integrated in series with the memristive device. Various types of selectors have been experimentally validated in previous studies [39,40,41,42].

The 1S–1R structure is efficient in terms of area utilization; however, it encounters a sneak-path issue since programming (reading or writing) a specific cell unavoidably affects its neighboring cells [43].

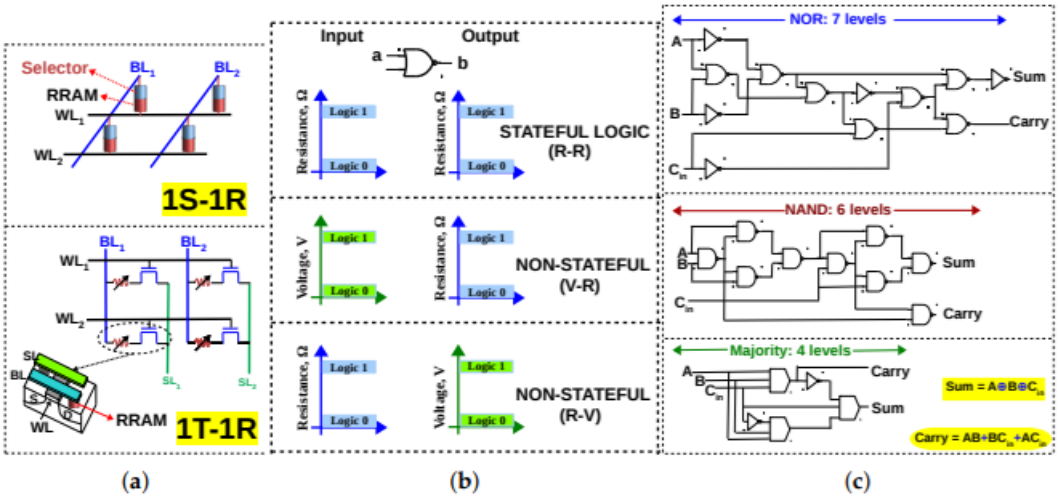


Figure (5): (a) 1S–1R and 1T–1R configuration of memristive memory array (b) If resistance is the only state variable, a memristive logic is said to be stateful. If voltage is also used in addition to resistance, it is said to be non-stateful (c) 1-bit full adder in terms of NOR gates [44], NAND gates [45,46] and majority gates [47]; Majority logic achieves less logical depth than NAND/NOR for 1-bit full adder.

A NOR gate is considered 'functionally complete' as it can be used to express any Boolean logic operation. Consequently, designing a NOR gate with memristive devices enables the implementation of any Boolean logic function using these devices. Based on the state variable utilized for computation, memristive logic families are categorized as either stateful or non-stateful. A memristive logic family is termed stateful when the Boolean variable is solely represented by the internal state of the memristor,

specifically its resistance, and computation is carried out by modifying this state [48]. Conversely, if voltage is employed alongside resistance, the logic family is classified as non-stateful (Fig. 5b).

In the NAND-based logic family discussed in [49], the XOR gate is realized through a sequence of four NAND operations. This suggests that when a memristive logic family relies on a weak fundamental logic primitive, all in-memory computations based on that family become inefficient, requiring lengthy sequences of operations. For instance, as illustrated in Fig. 5c, a 1-bit full adder can be represented using a specific logic primitive (NOR/NAND/Majority) to enable in-memory implementation. Utilizing stronger logic primitives, such as majority, can help reduce latency. This review aims to emphasize the advantages of memristive logic families that adopt majority as the core logic primitive (alongside NOT, since majority alone is functionally incomplete).

In-Memory Majority Logic

The concept of executing an in-memory majority gate within V–R and R–V logic is examined, along with an analysis of its respective benefits and limitations.

- V–R Majority Logic:
  In an RRAM array (1S–1R), the majority gate is realized by applying two of its input signals as voltages at the word line (WL) and bit line (BL), while the third input corresponds to the initial state of the RRAM. The resulting output is the updated non-volatile state of the device. This method of implementing a majority function is referred to as V–R logic. However, more precisely, it should be termed VandR–R logic, as the third input is determined by the resistance of the RRAM's initial state.

Table 2. Establishing the link between the majority function and Resistive RAM.

A	B	C	$M_3(A,B,C)$	$\overline{B}$	$M_3(A,\overline{B},C)$	$RM_3(A,B,C)$
0	0	0	0	1	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	1	1

The fourth column of Table 2 presents M3 (A, B, C), which represents the 3-input majority function derived from the first three columns. This function is defined as M3 (A, B, C) = AB + BC + AC. To illustrate how a Resistive RAM cell can realize the majority function, consider the Boolean variable C from Table 1, which corresponds to the initial state of a memory cell—where logic 0 is associated with the high-resistance state (HRS) and logic 1 corresponds to the low-resistance state (LRS).

Assuming that the RRAM cell storing C exhibits a symmetric switching characteristic, its internal resistance transitions from a high-resistance state (HRS) to a low-resistance state (LRS) when subjected to a voltage of VSET. Conversely, applying -VSET shifts the resistance from LRS back to HRS. Similar to CMOS technology, a high voltage, designated as VSET, represents logic 1, while logic 0 is defined as ground.

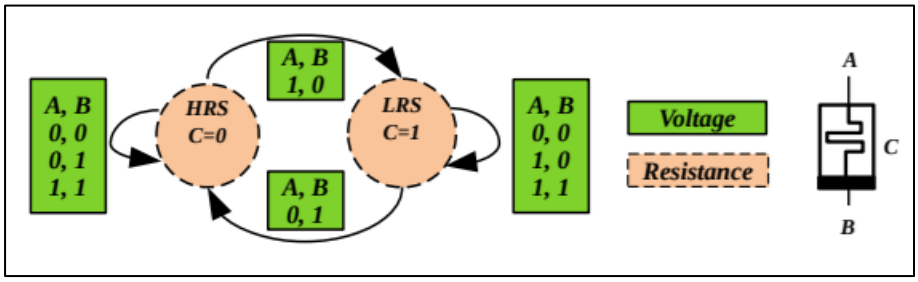


Figure (6): Illustration of V–R majority logic. Arrow indicates the state transition, which depends on the initial state of the RRAM cell C and the voltage applied across its terminals (A, B); dotted lines indicate the state variable of C, which is resistance, while A and B are voltages [50].

When inputs A and B are applied to the two terminals of an RRAM cell, the cell's state either remains unchanged or transitions based on its initial condition. Figure 6 demonstrates the various combinations of (A, B, C) acting on an RRAM cell. Specifically, when A is set to logic 1 and B to logic 0, the voltage applied across the cell corresponds to VSET, leading to a switch from HRS to LRS or vice versa. However, when A and B share the same value (either 0,0 or 1,1), the state of the memristor remains unaltered. This behavior introduces a novel Boolean function known as 'Resistive Majority,' RM3(A, B, C), which determines the resulting non-volatile state based on the initial internal condition C and the voltages applied at the terminals. Importantly, RM3(A, B, C) aligns with M3(A, B, C), as presented in Table 1. Complex logic functions can be effectively represented and manipulated using RM3 operators within Majority-Inverter Graphs (MIG), a logic framework composed of three-input majority nodes along with regular and complemented edges [51]. As detailed in [50], the authors illustrate how an eight-bit adder can be formulated using MIGs and subsequently implemented within a memristive memory array by leveraging the resistive majority function.

➢ R–V Majority Logic:

A majority gate is utilized during the read operation of a 1T–1R array, where the resistances of the memory cells serve as the inputs, and the output is detected as a voltage, forming an R–V logic system. In this configuration, an array of RRAM cells is structured in a 1T-1R arrangement, as illustrated in Figure 7. Each cell's state can be accessed or modified by selecting the corresponding wordline (WL) and applying the appropriate voltage across the bitline (BL) and sourceline (SL). During a read operation, if three rows are activated simultaneously (Rows 1 to 3 in Figure 7a), the resistances in column 1 effectively form a parallel network, assuming negligible parasitic resistance from BL and SL.

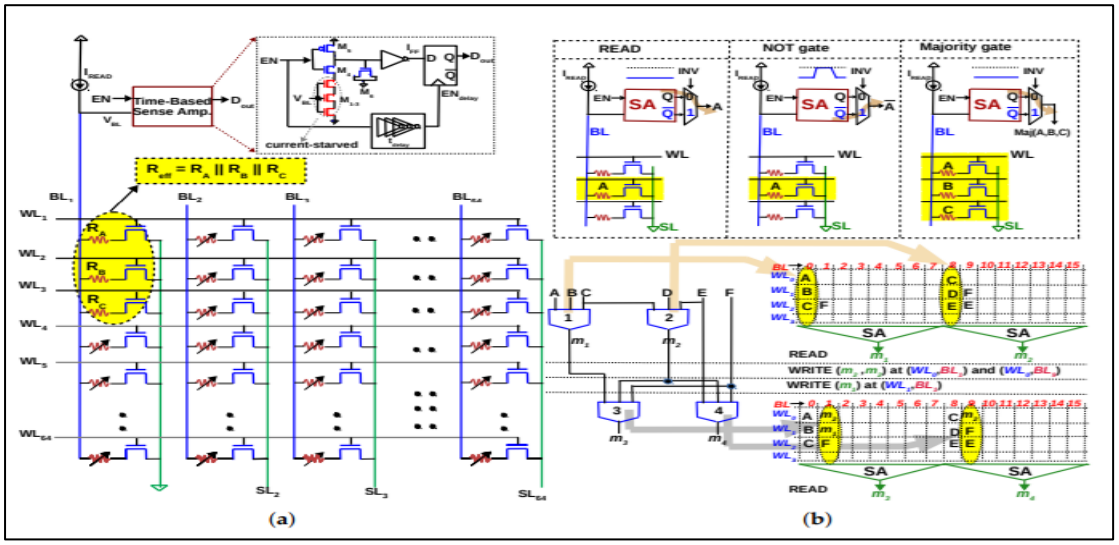


Figure (7): (a) In-memory majority gate proposed in [52,53]: When three rows are activated (WL1−3) simultaneously in a 1T-1R array, the three resistances RA, RB, RC will be in parallel (Inputs of the majority gate A, B, C are represented as resistances RA, RB, RC). An 'in-memory' majority gate can be implemented by accurately sensing the effective resistance Re f f during READ. (b) NOT operation implemented with a 2:1 multiplexer at the output of the SA. With majority and NOT gate implemented as READ, multiple levels of logic can be executed by writing the data back to the array, simplifying computing to READ and WRITE operations in memory. Multiple majority gates can be executed in parallel in the memory array, thereby reducing latency of in-memory computation.

A current-mode SA is utilized in [52], while a time-based SA is employed in [53] to ensure the majority gate operates correctly, even in the presence of typical RRAM variations. It is important to highlight that, unlike NAND and NOR gates, the majority gate alone does not constitute a functionally complete logic. However, when combined with a NOT gate, it becomes functionally complete, meaning any Boolean logic function can be realized using majority and NOT gates [51]. To achieve this, a NOT gate is implemented by capturing the inverted output of the SA, as demonstrated in Figure 7b.

A pictorial representation of the comparison between V–R logic and R–V logic is shown in Figure 8. In the V–R approach [54,55,56], the majority gate inputs are applied as voltages at WL/BL within the memory. This method alters the conventional role of row/column decoders, which are typically responsible for selecting rows or columns in a memory array. Consequently, the peripheral circuitry becomes more complex, requiring substantial modifications in the row/column decoders to handle both row selection during standard memory operations and input application during majority operations. On the other hand, the R–V implementation [52,53] preserves the traditional functionality of row/column decoders with only a slight modification—enhancing the row decoder to enable the selection of three rows during majority operations, which can be implemented using interleaved decoders [53]. Additionally, R–V logic [52,53] facilitates parallel processing, as multiple gates can be mapped onto the same set of rows, as demonstrated in Figure 8. This characteristic supports the development of in-memory parallel-prefix adders and ternary computing [53].

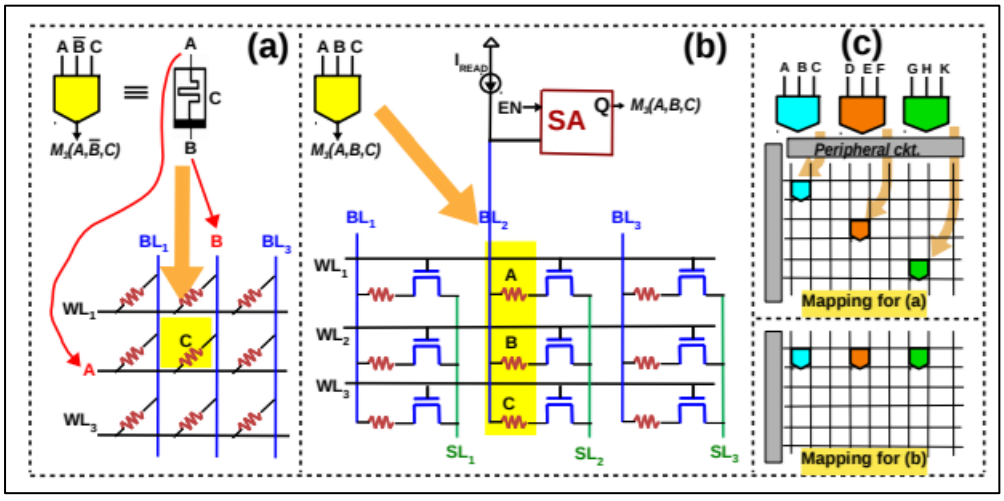


Figure (8): (a) V–R majority gate [54,55,56], (b) R–V majority gate [52,53], and (c) when executing multiple gates in parallel, the majority gates from [54,55,56] must be arranged diagonally, as executing two gates within the same row or column is not feasible.

Proposed In-Memory One-Bit Full Adders Using QCA

Fig. 9 a) shows the Approximate Adder (AA1). In this design, the sum output is achieved by the one majority gate with an input of A, B and ~C. Then the carry is generated from the one majority gate with an input of A, B and C. The majority gate count for this design is 2.

The function of the sum and the carry are given by:

C0 = m(a, b, c) ;

Sum = m(a, b, ~c) ;

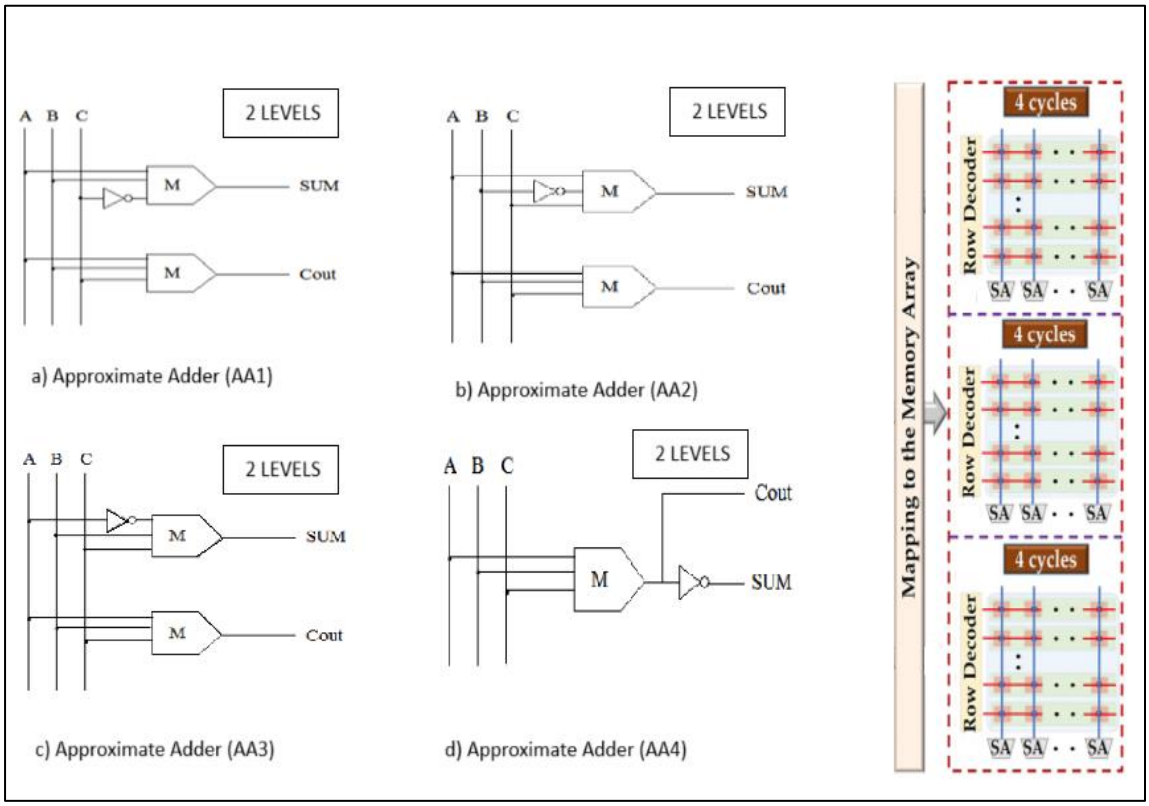


Figure (9): Full adder using Majority logic, logic level and in-memory implementation of full adder using Majority gate and inverter. The use of Majority logic reduces the logical depth and number of cycles.

Fig.9 b) shows the Approximate Adder (AA2). In this design, the sum output is achieved by the one majority gate with an input of A, ~B and C. Then the carry is generated from the one majority gate with an input of A, B and C. The majority gate count for this design is 2.

The function of the sum and the carry are given by:

C0 = m(a, b, c) ;

Sum = m(a, ~b , c)) ;

Fig.9 c) shows the Approximate Adder (AA3). In this design, the sum output is achieved by the one majority gate with an input of ~A, B and C. Then the carry is generated from the one majority gate with an input of A, B and C. The majority gate count for this design is 2.

The function of the sum and the carry are given by:

C0 = m(a, b, c) ;

Sum = m(~a, b, c)) ;

Fig.9 d) shows the Approximate Adder (AA4). In this design, the carry output is achieved by the one majority gate with an input of A, B and C. Then the sum is obtained from the inverted output of carry. The majority gate count for this design is 1. The function of the sum and the carry are given by: C0 = m(a, b, c) ; Sum = ~C0;

The full adder is the basic building block in the ripple carry adder, and most other adder circuits. A ripple carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascaded with the carry output from each full adder connected to the carry input of the next full adder in the chain. Fig. 10 shows the 4bit approximate adder (RCA) which is designed from the approximation adder 4(AA 4) with 45nm technology.

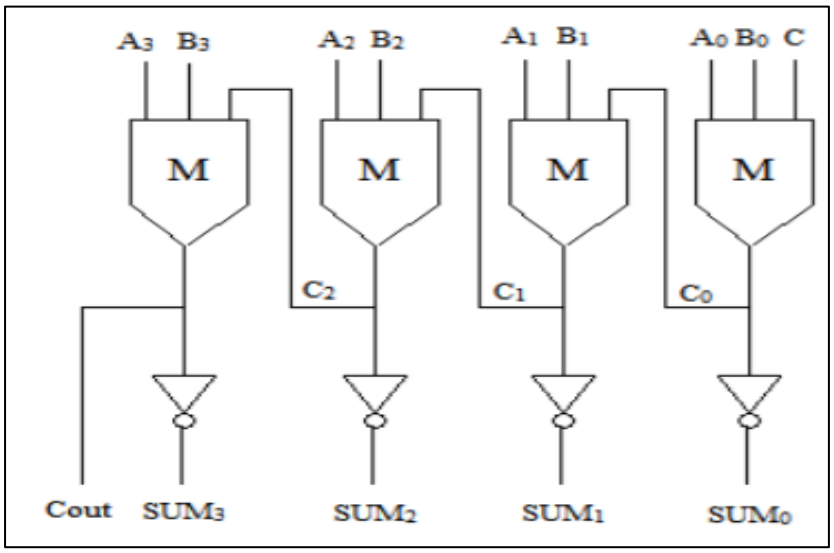


Figure (10): 4-bit Ripple Carry Adder

Table 3. Comparison table for various types of adders

Types of adders	No. of Majority gates	No. of Inverters	No. of Levels
Normal FA [71, 72]	5	3	5
Conventional FA [47, 73]	3	2	4
Majority FA [32, 33, 34]	2	1	3
AA1	2	1	2
AA2	2	1	2
AA3	2	1	2
AA4	1	1	2

From this table to know about that using the Approximate Adder4 (AA 4) instead of normal full adder we can reduce the circuit complexity.

CONCLUSION

In-memory computing is envisioned to provide a unique advantage for cryogenic systems by improving the energy efficiency and consequently, reducing the cooling issue. Here, proposed an in-memory adder based on Majority logic using the QAHE-based cryogenic memory. The proposed Approximate adder has been design using the QCA Designer tool in a cryogenic temperature with 45nm technology for in-memory bit-serial computing. In this paper, Approximate Adder Using QCA has been presented that reduces the number of majority gates compared to the conventional full adder. From all the above, approximation Adder AA4 will give effective results in terms of area and error rate. The proposed Approximate Adder produces the accurate output rather than the exact output with low error rate. When the errors introduced by these approximations were reflected at a high level like signal processing algorithms the impact on output quality was very little. A decrease in the number of majority cells helped in reducing overall area when number of bits increases.

ACKNOWLEDGMENT

I express my deepest gratitude to my research Supervisor Dr. D. S. Shylu Sam, Internal Expert Dr. D. Nirmal and External Expert Dr. P.T. Vanathi for their invaluable guidance, support and encouragement throughout this research. Their expertise and insightful feedback have been instrumental in shaping this work. I extend my sincere thanks to Karunya University and the Department of Electronics and Communication for giving the opportunity to present my new ideas. I also acknowledge the contributions of the reviewers who provided constructive feedback and helped refine this paper for publication. My heartfelt appreciation goes to my family and friends for their unwavering support, patience, and motivation. Their encouragement has been a pillar of strength throughout this journey.

REFERENCES

[1] Patra, B. et al. “Cryo-CMOS Circuits and Systems for Quantum Computing Applications”. IEEE J. Solid-State Circuits (2018) doi:10.1109/JSSC.2017.2737549.

[2] Shamiul Alam, Md. Mazharul Islam, Md. Shafayat Hossain et al. “Cryogenic In-Memory Bit-Serial Addition Using Quantum Anomalous Hall Effect-Based Majority Logic”, IEEE, Volume 11, June 2023 DOI: 10.1109/ACCESS.2023.3285604.

[3] Jiang, Z., Yin, S., Seo, J. S. & Seok, M. “C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism”. IEEE J. Solid-State Circuits 55, 1888–1897 (2020).

[4] Zhang, X., Mohan, V. & Basu, A. “CRAM: Collocated SRAM and DRAM with In-Memory Computing-Based Denoising and Filling for Neuromorphic Vision Sensors in 65 nm CMOS”. IEEE Trans. Circuits Syst. II Express Briefs 67, 816–820 (2020).

[5] Wang, P. et al. “Cryogenic Performance for Compute-in-Memory Based Deep Neural Network Accelerator”. 1–4 (2021) doi:10.1109/ISCAS51556.2021.9401756.

[6] Li, S. et al. “Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories”. Proc. - Des. Autom. Conf. 05-09-June, (2016).

[7] Yoo, T., Kim, H., Chen, Q., Kim, T. T. H. & Kim, B. “A Logic Compatible 4T Dual Embedded DRAM Array for InMemory Computation of Deep Neural Networks”. Proc. Int. Symp. Low Power Electron. Des. 2019-July, (2019).

[8] NiLeibin, HuangHantao, LiuZichuan, V., J. & YuHao. “Distributed In-Memory Computing on Binary RRAM Crossbar”. ACM J. Emerg. Technol. Comput. Syst. 13, (2017).

[9] Yin, S. et al. “Monolithically Integrated RRAM- And CMOS-Based In-Memory Computing Optimizations for Efficient Deep Learning”. IEEE Micro 39, 54–63 (2019).

[10] Bashar, M. K. et al. “An Oscillator-based MaxSAT solver”. (2021).

[11] Fan, D., Angizi, S. & He, Z. “In-Memory Computing with Spintronic Devices”. Proc. IEEE Comput. Soc. Annu. Symp. VLSI, ISVLSI 2017-July, 683–688 (2017).

[12] Kang, W., Wang, H., Wang, Z., Zhang, Y. & Zhao, W. “In-Memory Processing Paradigm for Bitwise Logic Operations in STT-MRAM”. IEEE Trans. Magn. 53, (2017).

[13] He, Z., Angizi, S. & Fan, D. “Exploring STT-MRAM based in-memory computing paradigm with application of image edge extraction”. Proc. - 35th IEEE Int. Conf. Comput. Des. ICCD 2017 439–446 (2017) doi:10.1109/ICCD.2017.78.

[14] Resch, S., Cilasun, H. & Karpuzcu, U. R. “Cryogenic PIM: Challenges Opportunities”. IEEE Comput. Archit. Lett. 20, 74– 77 (2021).

[15] Reis, D., Niemier, M. & Sharon Hu, X.” Computing in memory with FeFETs”. Proc. Int. Symp. Low Power Electron. Des. 18, (2018).

[16] Ni, K. et al. "In-Memory Computing Primitive for Sensor Data Fusion in 28 nm HKMG FeFET Technology". Tech. Dig. - Int. Electron Devices Meet. IEDM 2018-Decem, 16.1.1-16.1.4 (2019).

[17] Qiao, W. et al. "Non-volatile in Memory Dual-Row X(N)OR Operation with Write Back Circuit Based on 1T1C FeRAM". 2020 IEEE 15th Int. Conf. Solid-State Integr. Circuit Technol. ICSICT 2020 - Proc. (2020).

[18] Hosseini, P., Sebastian, A., Papandreou, N., Wright, C. D. & Bhaskaran, H. "Accumulation-based computing using phase-change memories with FET access devices". IEEE Electron Device Lett. 36, 975–977 (2015).

[19] Chakraborty, I., Saha, G. & Roy, K. "Photonic In-Memory Computing Primitive for Spiking Neural Networks Using Phase-Change Materials". Phys. Rev. Appl. 11, 014063 (2019).

[20] Jaberipur, G.; Parhami, B.; Abedi, D. "Adapting Computer Arithmetic Structures to Sustainable Supercomputing in Low-Power, Majority-Logic Nanotechnologies". IEEE Trans. Sustain. Comput. 2018, 3, 262–273.

[21] Krinner, S., et al. "Engineering cryogenic interfaces for quantum processors". Nature Electronics (2022).

[22] Patra, B., et al. "Cryo-CMOS circuits and systems for scalable quantum computing". IEEE Journal of Solid-State Circuits (2020).

[23] Devoret, M. H., & Schoelkopf, R. J. "Superconducting circuits for quantum information". Science (2013).

[24] Arute, F., et al. "Quantum supremacy using a programmable superconducting processor. Nature (2019).

[25] Preskill, J. "Quantum computing in the NISQ era and beyond". Quantum (2018).

[26] Wong, H.-S. P., & Salahuddin, S. "Memory devices based on emerging resistive switching technologies." Proceedings of the IEEE, 103(6), 1034–1045, (2015).

[27] Kim, S., et al. "Selector devices for cross-point memory arrays: A review." IEEE Transactions on Electron Devices, 66(2), 465–479, (2019).

[28] Yu, S. "Neuro-inspired computing with emerging nonvolatile memorys." Proceedings of the IEEE (2016).

[29] Xie, Y. et al. "Emerging Memory Technologies for Neuromorphic Computing." Nature Electronics (2017).

[30] Li, H. et al. "Memristor-based logic-in-memory: A survey and perspective." IEEE Transactions on Circuits and Systems (2018).

[31] Chen, P. et al. "Rethinking logic with memristors: A scalable approach to embedded computing." IEEE Transactions on Nanotechnology (2015).

[32] R. Lindaman, "A theorem for deriving majority-logic networks within an augmented Boolean algebra," IEEE Trans. Electron. Comput., vol. EC-9, no. 3, pp. 338–342, Sep. 1960, doi: 10.1109/TEC.1960.5219856.

[33] T. Oya, T. Asai, T. Fukui, and Y. Amemiya, "A majority-logic nanodevice using a balanced pair of single-electron boxes," J. Nanoscience Nanotechnol., vol. 2, no. 3, pp. 333–342, Jul. 2002, doi: 10.1166/JNN.2002.108.

[34] Reuben, "Rediscovering majority logic in the post-CMOS era: A perspective from in-memory computing," J. Low Power Electron. Appl., vol. 10, no. 3, p. 28, Sep. 2020, doi: 10.3390/JLPEA10030028.

[35] M. F. Ali, A. Jaiswal, and K. Roy, "In-memory low-cost bit-serial addition using commodity DRAM technology," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 1, pp. 155–165, Jan. 2020, doi: 10.1109/TCSI.2019.2945617.

[36] S. Alam, M. S. Hossain, and A. Aziz, "A non-volatile cryogenic randomaccess memory based on the quantum anomalous Hall effect," Sci. Rep., vol. 11, no. 1, pp. 1–9, Apr. 2021, doi: 10.1038/s41598-021-87056-7.

[37] Reuben, J.; Fey, D.; Wenger, C. A Modeling Methodology for Resistive RAM Based on Stanford-PKU Model With Extended Multilevel Capability. IEEE Trans. Nanotechnol. 2019, 18, 647–656. [CrossRef].

[38] Golonzka, O.; Arslan, U.; Bai, P.; Bohr, M.; Baykan, O.; Chang, Y.; Chaudhari, A.; Chen, A.; Clarke, J.; Connor, C.; et al. Non-Volatile RRAM Embedded into 22FFL FinFET Technology. In Proceedings of the 2019 Symposium on VLSI Technology, Kyoto, Japan, 9–14 June 2019; pp. T230–T231.

[39] Hsieh, C.C.; Chang, Y.F.; Chen, Y.C.; Shahrjerdi, D.; Banerjee, S.K. Highly Non-linear and Reliable Amorphous Silicon Based Back-to-Back Schottky Diode as Selector Device for Large Scale RRAM Arrays. ECS J. Solid State Sci. Technol. 2017, 6, N143–N147.

[40] Lin, C.Y.; Chen, P.H.; Chang, T.C.; Chang, K.C.; Zhang, S.D.; Tsai, T.M.; Pan, C.H.; Chen, M.C.; Su, Y.T.; Tseng, Y.T.; et al. Attaining resistive switching characteristics and selector properties by varying forming polarities in a single HfO2-based RRAM device with a vanadium electrode. Nanoscale 2017, 9, 8586–8590.

[41] Kim, S.; Lin, C.Y.; Kim, M.H.; Kim, T.H.; Kim, H.; Chen, Y.C.; Chang, Y.F.; Park, B.G. Dual Functions of V/SiOx/AlOy/p++Si Device as Selector and Memory. Nanoscale Res. Lett. 2018, 13.

[42] Chen, C.; Lin, C.; Chen, P.; Chang, T.; Shih, C.; Tseng, Y.; Zheng, H.; Chen, Y.; Chang, Y.; Lin, C.; et al. The Demonstration of Increased Selectivity During Experimental Measurement in Filament-Type Vanadium Oxide-Based Selector. IEEE Trans. Electr. Devices 2018, 65, 4622–4627.

[43] Talati, N.; Ben-Hur, R.; Wald, N.; Haj-Ali, A.; Reuben, J.; Kvatinsky, S. mMPU—A Real Processing-in-Memory Architecture to Combat the von Neumann Bottleneck. In Applications of Emerging Memory Technology: Beyond Storage; Suri, M., Ed.; Springer: Singapore, 2020; pp. 191–213.

[44] Ben-Hur, R.; Ronen, R.; Haj-Ali, A.; Bhattacharjee, D.; Eliahu, A.; Peled, N.; Kvatinsky, S. SIMPLER MAGIC: Synthesis and Mapping of In-Memory Logic Executed in a Single Row to Improve Throughput. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2019.

[45] Adam, G.C.; Hoskins, B.D.; Prezioso, M.; Strukov, D.B. Optimized stateful material implication logic for three- dimensional data manipulation. Nano Res. 2016, 9, 3914–3923.

[46] Kumar, A.P.; Aditya, B.; Sony, G.; Prasanna, C.; Satish, A. Estimation of power and delay in CMOS circuits using LCT. Indones. J. Electr. Eng. Comput. Sci. 2019, 14, 990–998.

[47] Rumi, Z.; Walus, K.; Wei, W.; Jullien, G.A. A method of majority logic reduction for quantum cellular automata. IEEE Trans. Nanotechnol. 2004, 3, 443–450.

[48] Lehtonen, E.; Poikonen, J.H.; Laiho, M. Memristive Stateful Logic. In Handbook of Memristor Networks; Chua, L., Sirakoulis, G.C., Adamatzky, A., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 1101–1121, doi:10.1007/978-3-319-76375-0_38.

[49] Shen, W.; Huang, P.; Fan, M.; Han, R.; Zhou, Z.; Gao, B.; Wu, H.; Qian, H.; Liu, L.; Liu, X.; et al. Stateful Logic Operations in One-Transistor-One- Resistor Resistive Random Access Memory Array. IEEE Electr. Device Lett. 2019, 40, 1538–1541.

[50] Reuben, J.; Talati, N.; Wald, N.; Ben-Hur, R.; Ali, A.H.; Gaillardon, P.E.; Kvatinsky, S. A Taxonomy and Evaluation Framework for Memristive Logic. In Handbook of Memristor Networks; Chua, L., Sirakoulis, G.C., Adamatzky, A., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 1065–1099.

[51] Amarú, L.; Gaillardon, P.E.; Micheli, G.D. Majority-Inverter Graph: A New Paradigm for Logic Optimization. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2016, 35, 806–819.

[52] Reuben, J. Binary Addition in Resistance Switching Memory Array by Sensing Majority. Micromachines 2020, 11, 496.

[53] Reuben, J.; Pechmann, S. A Parallel-friendly Majority Gate to Accelerate In-memory Computation. In Proceedings of the 2020 IEEE 31st International Conference on Application-Specific Systems, Architectures and Processors (ASAP), Manchester, UK, 6–8 July 2020; pp. 93–100.

[54] Gaillardon, P.; Amaru, L.; Siemon, A.; Linn, E.; Waser, R.; Chattopadhyay, A.; De Micheli, G. The Programmable Logic-in-Memory (PLiM) computer. In Proceedings of the 2016 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 14–18 March 2016; pp. 427–432.

[55] Shirinzadeh, S.; Soeken, M.; Gaillardon, P.; Drechsler, R. Logic Synthesis for RRAM-Based In-Memory Computing. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2018, 37, 1422–1435.

[56] Bhattacharjee, D.; Easwaran, A.; Chattopadhyay, A. Area-constrained technology mapping for in-memory computing using ReRAM devices. In Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, Japan, 16–19 January 2017; pp. 69–74.

[57] Huang, P.; Kang, J.; Zhao, Y.; Chen, S.; Han, R.; Zhou, Z.; Chen, Z.; Ma, W.; Li, M.; Liu, L.; et al. Reconfigurable Nonvolatile Logic Operations in Resistance Switching Crossbar Array for Large-Scale Circuits. Adv. Mater. 2016, 28, 9758–9764.

[58] Chang, Y.; Zhou, F.; Fowler, B.W.; Chen, Y.; Hsieh, C.; Guckert, L.; Swartzlander, E.E.; Lee, J.C. Memcomputing (Memristor + Computing) in Intrinsic SiOx-Based Resistive Switching Memory: Arithmetic Operations for Logic Applications. IEEE Trans. Electr. Devices 2017, 64, 2977–2983.

[59] Cheng, L.; Zhang, M.Y.; Li, Y.; Zhou, Y.X.; Wang, Z.R.; Hu, S.Y.; Long, S.B.; Liu, M.; Miao, X.S. Reprogrammable logic in memristive crossbar for in-memory computing. J. Phys. D Appl. Phys. 2017, 50, 505102.

[60] Teimoory, M.; Amirsoleimani, A.; Shamsi, J.; Ahmadi, A.; Alirezaee, S.; Ahmadi, M. Optimized implementation of memristor-based full adder by material implication logic. In Proceedings of the 2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS), Marseille, France, 7–10 December 2014; pp. 562–565.

[61] Rohani, S.G.; Taherinejad, N.; Radakovits, D. A Semiparallel Full-Adder in IMPLY Logic. IEEE Trans. Very Larg. Scale Integr. (VLSI) Syst. 2019; 28, 297–301.

[62] Kim, K.M.; Williams, R.S. A Family of Stateful Memristor Gates for Complete Cascading Logic. IEEE Trans. Circuits Syst. I Regul. Pap. 2019, 66, 4348–4355.

[63] Siemon, A.; Drabinski, R.; Schultis, M.J.; Hu, X.; Linn, E.; Heittmann, A.; Waser, R.; Querlioz, D.; Menzel, S.; Friedman, J.S. Stateful Three-Input Logic with Memristive Switches. Sci. Rep. 2019, 9, 14618.

[64] Xu, L.; Yuan, R.; Zhu, Z.; Liu, K.; Jing, Z.; Cai, Y.; Wang, Y.; Yang, Y.; Huang, R. Memristor-Based Efficient In-Memory Logic for Cryptologic and Arithmetic Applications. Adv. Mater. Technol. 2019, 4, 1900212.

[65] Siemon, A.; Menzel, S.; Bhattacharjee, D.; Waser, R.; Chattopadhyay, A.; Linn, E. Sklansky tree adder realization in 1S1R resistive switching memory architecture. Eur. Phys. J. Spec. Top. 2019, 228, 2269–2285.

[66] Shen, W.; Huang, P.; Fan, M.; Han, R.; Zhou, Z.; Gao, B.; Wu, H.; Qian, H.; Liu, L.; Liu, X.; et al. Stateful Logic Operations in One-Transistor-One- Resistor Resistive Random Access Memory Array. IEEE Electr. Device Lett. 2019, 40, 1538–1541.

[67] Revanna, N.; Swartzlander, E.E. Memristor based adder circuit design. In Proceedings of the 2016 50th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 6–9 November 2016; pp. 162–166.

[68] Wang, Z.; Li, Y.; Su, Y.; Zhou, Y.; Cheng, L.; Chang, T.; Xue, K.; Sze, S.M.; Miao, X. Efficient Implementation of Boolean and Full-Adder Functions with 1T1R RRAMs for Beyond Von Neumann In-Memory Computing. IEEE Trans. Electr. Devices 2018, 65, 4659–4666.

[69] Cheng, L.; Li, Y.; Yin, K.S.; Hu, S.Y.; Su, Y.T.; Jin, M.M.; Wang, Z.R.; Chang, T.C.; Miao, X.S. Functional Demonstration of a Memristive Arithmetic Logic Unit (MemALU) for In-Memory Computing. Adv. Funct. Mater. 2019, 29, 1905660.

[70] Kim, Y.S.; Son, M.W.; Song, H.; Park, J.; An, J.; Jeon, J.B.; Kim, G.Y.; Son, S.; Kim, K.M. Stateful In-Memory Logic System and Its Practical Implementation in a TaOx-Based Bipolar-Type Memristive Crossbar Array. Adv. Intell. Syst. 2020, 2, 1900156.

[71] P.D. Tougaw and C. S. Lent, “Logical devices implemented usingquantum cellular automata,” J. Appl. Phys., American Institute ofPhysics, vol. 75, pp. 1818- 1824, 1994.

[72] A. Vetteth, K. Walus, V. S. Dimitrov, and G. A. Jullien, "Quatum-dotcellular automata carry-look-ahead adder and barrel shifter,” in Proc.IEEE Emerging Telecommunications Technologies Conference, Dallas,TX, Sept. 2002.

[73] Wei wang, konradwalus and g. A. Jullien, ”QuantumDot Cellular Automata Adders”, 461 - 464 vol.2 2003 Third IEEE Conference on Nanotechnology. IEEENANO.