2024 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)

Optimizing Computational Efficiency: In-Memory Computing with Dynamic Switching

Kun-Lin Tsai and Chao-Ting Huang
Dept. of Electrical Engineering, Tunghai University, Taichung, Taiwan (R.O.C.)
E-mail: kltsai@thu.edu.tw; kimihuang4509@gmail.com

Abstract — **Modern computing architectures mainly utilize the Von-Neumann architecture. However, the increasing demand for processing vast volumes of data worsens the Von-Neumann bottleneck, which results in significant energy consumption and computational latency. In-memory computing provides an effective solution for Von-Neumann bottleneck by performing fundamental logical or arithmetic operations inside the memory directly. In this paper, an in-memory computing architecture with dynamic switching function is proposed to provide multiple logical or arithmetic operations. The 8+T SRAM cell and pass transistor logic circuits are utilized in the proposed in-memory computing architecture. Specifically, we introduce the design of a dynamic adder-subtractor capable of switching between addition and subtraction, thereby enhancing the efficiency of in-memory computing.**

I. Introduction

The widespread application of modern artificial intelligence and deep learning underscores the inefficiencies of the traditional Von Neumann architecture [1], particularly in managing high-speed, large-volume data transfers. The separation of the Arithmetic Logic Unit (ALU) and Memory Unit necessitates repeated data transfers, resulting in the well-documented Von Neumann bottleneck [2], which reduces system throughput and increases power consumption and performance degradation.

In-Memory Computing (IMC) [3][4] addresses this bottleneck by performing logical or arithmetic operations within memory units using simple circuits, thus reducing the frequency of data transfers. This paper aims to efficiently execute the six basic logical operations (AND, NAND, OR, NOR, XOR, XNOR) and two arithmetic operations (full adder and full subtractor) using 8+T Static Random Access Memory (SRAM) [8-11]. This architecture leverages Read Bit Lines (RBL, RBLB) for AND and NOR results and uses inverters for NAND and NOR results. Additionally, XOR and XNOR operations are achieved using two transistors, as referenced in previous studies [17-19]. For arithmetic operations, we adopt the full adder architecture [13-14], utilizing two sets of four transistors for Sum (S) and Carry-out (Co) outputs. By comparing the truth tables of the full adder and full subtractor, the design is enhanced to produce Difference (D) and Borrow-out (Bo) outputs for the full subtractor. Dynamic input switching allows for quick retrieval of full adder and full subtractor values within the memory, enabling multiple operations within a single memory unit.

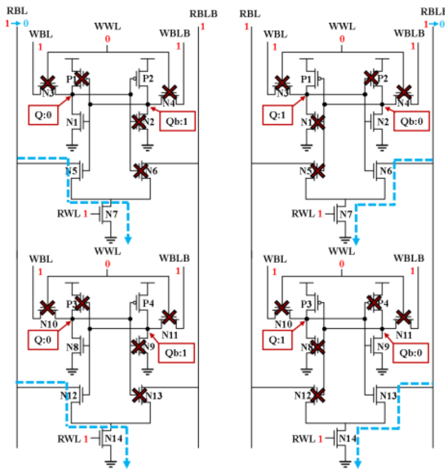

Fig. 1. The 8+T SRAM operations in case (0,0) and case (1,1)

II. 8+T SRAM and Pass Transistor Operations

Many researchers have proposed various ideas in the field of IMC. Agrawal *et al.* [10] have suggested using two cells of 8+T SRAM in series to obtain logical operation results. Fig. 1 shows the 8+T SRAM operations in case (0,0) and case (1,1). In the left part of Fig. 1, both cells are written with 0, and the 8+T SRAM begins read mode, activating RWL. Q1 and Q2 are 0, causing N6 and N13 to be cut off, keeping RBLB at a high potential. Qb1 and Qb2 are 1, causing N5 and N12 to conduct and discharge RBL to GND. Therefore, in case (0,0), RBL drops to low voltage, i.e., logic 0, and RBLB remains at high voltage, i.e., logic 1. The right diagram in Fig. 1 operates on the same principle as the left diagram described above. The conclusion is RBL remains at high voltage, i.e., logic 1, and RBLB also remains at high voltage.

In Fig. 2, the operating principles described above also apply. Therefore, for case (0,1) and case (1,0), the conclusion is RBL will drop to low voltage, and RBLB will also drop to low voltage. Table 1 lists the truth table of AND and NOR functions, it can be observed that RBL exhibits AND logic while RBLB exhibits NOR logic.

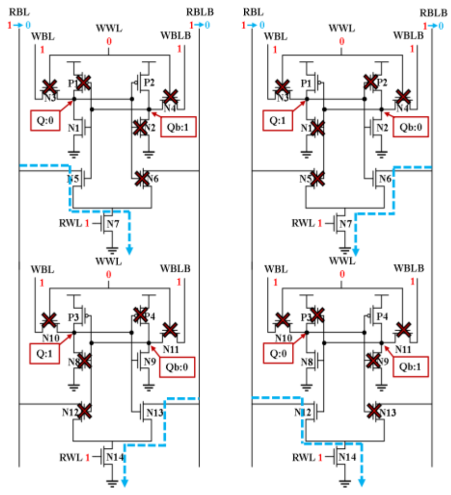

Fig. 2 The 8+T SRAM operations in case (0,1) and case (1,0)

Table 1 Logical Behavior Table of 8+T SRAM

Q1	Q2	RBL (AND)	RBLB (NOR)
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	0

This means that if an inverter is connected the resulting output will be NAND logic. Conversely, if an inverter is connected next to RBLB, the resulting output will be OR logic. In this manner, all four basic logical operations can be quickly obtained within the memory using RWL control.

Singh *et al*. [13] used Pass Transistor Logic (PTL) to generate the Sum and Cout of a full adder. This architecture first outputs the input values as XOR logic and connects them to an inverter to generate XNOR. These results are then used to produce the output of the full adder. Compared to CMOS, PTL uses fewer transistors to create various logic gates. In PTL, transistors act as switches between circuit nodes to determine logic levels without directly connecting to a voltage source, often resulting in lower output voltage. Multiple transistors in series usually require a stabilized power supply to restore the signal voltage.

For instance, Fig. 3 compares CMOS and PTL architectures for an AND gate. The PTL significantly reduces the number of required transistors. However, the XOR generation architecture in the referenced paper has flaws. Consequently, this study adopts only the full adder architecture from that reference, as illustrated in Fig. 4. Utilizing PTL, this design completes the full adder operation and significantly reduces transistor count compared to traditional implementations. Furthermore, PTL's design inherently lowers power consumption due to its lack of reliance on a stable voltage source.

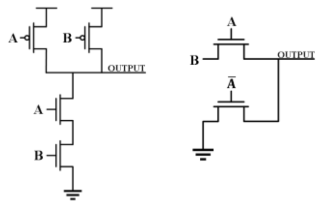

Fig. 3 AND Logic Gate in CMOS and PTL Architectures

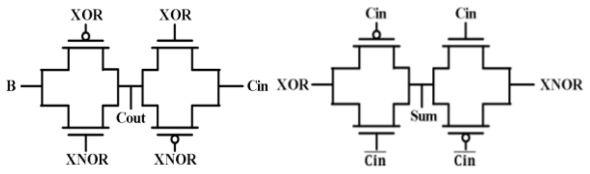

Fig. 4 Illustration of Full Adder Implementation Using PTL

III. Dynamic Switch in Memory Computing

This paper selects 8+T SRAM for its unique characteristics, enabling the generation of multiple logic gates with a few inverters. Separating the read and write lines addresses the data inversion issue found in 6T SRAM. While 8+T SRAM inherently produces four logic gates (AND, NAND, OR, NOR), most arithmetic operations necessitate XOR and XNOR gates. Referencing [7], this study generates XOR and XNOR gates by connecting two transistors to the Q1 and Q2 points in two cells.

These XOR and XNOR gates are then integrated into the PTL adder architecture to yield a two-bit adder output. By comparing the truth tables of addition and subtraction, the architecture can be slightly modified to produce subtraction outputs. Finally, incorporating input switching functionality allows for dynamic switching between addition, subtraction, and logic operations within the memory.

A. Logical Operations

According to one of the papers discussed in the Related Studies, we can understand that through the discharge characteristics of 8+T SRAM, an AND logic gate result can be obtained at the RBL end, and a NOR logic gate result can be obtained at the RBLB end. However, if an inverter is added to each end, NAND and OR results can be achieved, as shown in Fig. 5. Additionally, the extra inverter acts as a buffer, making the output logic more stable.

Next, we will generate XOR and XNOR based on the methodology described in reference [7]. As shown in Fig. 6, connecting the Q1, Q2, and Qb1 points from Fig. 5 using the CMOS inverter architecture yields the XNOR result. The circuit logic is as follows: when Q2 is 1, the NMOS transistor conducts and the PMOS transistor is cut off, making Q1 represent the XNOR outcome. Conversely, when Q2 is 0, the PMOS transistor conducts and the NMOS transistor is cut off, making Qb1 represent the XNOR outcome. An additional

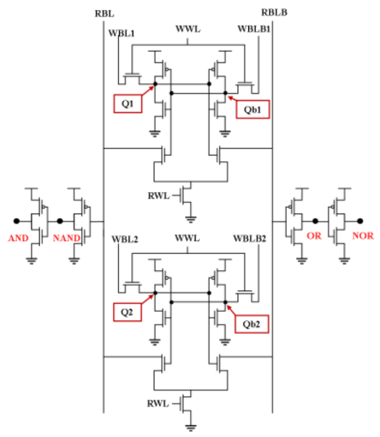

Fig.5 8+T SRAM Generating AND, NAND, OR, NOR

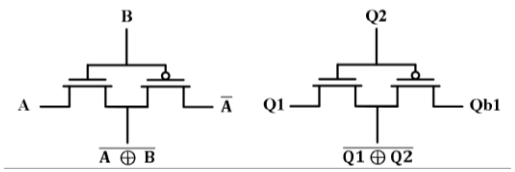

Fig.6 The XNOR integrated with the 8+T SRAM circuit.

inverter is then used to produce the XOR output. The derived truth table for XOR and XNOR logic with respect to Q1 and Q2 follows from this circuit logic.

B. Addition and Subtraction Operation

Upon completing the six basic logical operations, we utilize the previously generated XOR and XNOR results to perform addition and subtraction operations. We first examine the circuit logic for Cout, as illustrated in Fig. 7. When the XOR result of Q1 and Q2 is 1 and the XNOR result is 0, Cout takes the Cin value. Conversely, when the XOR result is 0 and the XNOR result is 1, Cout takes the Q2 value. The corresponding truth table is presented in

Table 2 references in alphabetical order or in order of appearance in the paper [6]. Next, let's discuss the circuit logic for the Sum. As shown in Fig.8, when Cin is 1 and $\overline{Cin}$ is 0, Sum directly takes the XNOR value of Q1 and Q2. Conversely, when Cin is 0 and $\overline{Cin}$ is 1, Sum directly takes the XOR value of Q1 and Q2. The corresponding truth table is presented in Table 3.

Table.2 Truth Table of Cout Using PTL Architecture

Q2	Cin	XOR	XNOR	Cout
0	0	1	0	0(Cin)
0	0	0	1	0(Q2)
0	1	1	0	1(Cin)
0	1	0	1	0(Q2)
1	0	1	0	0(Cin)
1	0	0	1	1(Q2)
1	1	1	0	1(Cin)
1	1	0	1	1(Q2)

Table. 3 Truth Table of Cout Using PTL Architecture

XOR	XNOR	Cin	$\overline{Cin}$	Sum
0	0	1	0	0(XNOR)
0	0	0	1	0(XOR)
0	1	1	0	1(XNOR)
0	1	0	1	0(XOR)
1	0	1	0	0(XNOR)
1	0	0	1	1(XOR)
1	1	1	0	1(XNOR)
1	1	0	1	1(XOR)

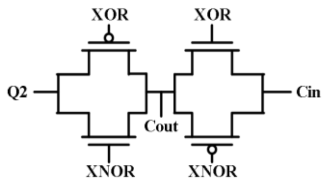

Fig.7 Circuit Diagram of Cout in PTL Architecture

Comparing the truth tables of the full adder and full subtractor reveals that the Difference in the full subtractor and the Sum in the full adder are identical. This suggests that the PTL full adder architecture generating the Sum for addition can also produce the Difference for subtraction.

Next, we consider integrating Cout and Bout. Based on the previously introduced PTL adder circuit logic, when Q1 and Q2 yield XNOR = 1, Bout equals Q2. Conversely, when Q1 and Q2 yield XOR = 1, Bout equals Cin, the inverse logic of addition. Thus, by interchanging Q2 and Cin in the PTL architecture generating Cout, as shown in Fig. 9, Bout can be obtained.

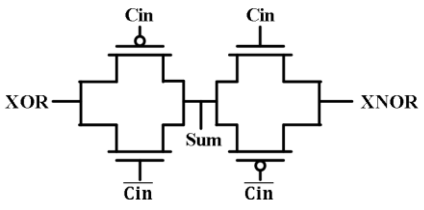

Fig.8 Circuit Diagram of Sum in PTL Architecture

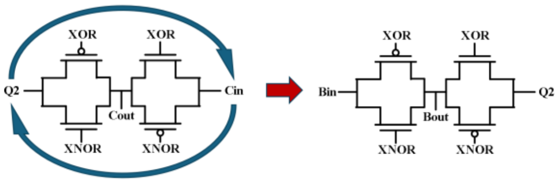

Fig.9 Generating Bout in PTL Architecture

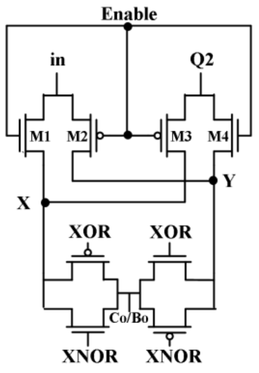

Fig.10 Dynamic Switching with PTL Architecture for Generating Co/Bo

Given that both addition and subtraction can be integrated within the PTL architecture, we aim to enable dynamic switching between the Q2 and Cin/Bin points in the Cout/Bout generation. Therefore, we have designed the following architecture. For readability, both Cin and Bin are referred to as "in," as shown in Fig. 10. When the Enable input is 0, M2 and M3 conduct, while M1 and M4 are cut off, allowing the data from the "in" terminal to reach point Y through M2, and the data from Q2 to reach point X through M3. This means that "when Enable = 0, the PTL architecture performs addition to generate Cout." Conversely, when the Enable input is 1, M1 and M4 conduct, while M2 and M3 are cut off, allowing the data from the "in" terminal to reach point X through M1, and the data from Q2 to reach point Y through M4. This means that when Enable = 1, the PTL architecture performs subtraction to generate Bout.

The entire design and operation of the circuit have been introduced above. Utilizing the 8+T SRAM in combination with the PTL adder-subtractor allows for efficient and rapid logical and arithmetic operations within the memory. With this, the complete in-memory computing architecture has been thoroughly explained.

IV. Experimental Results

A. *Logical Operation Results*

The preceding sections thoroughly introduce the architecture and operation of this study. Next, we will present the results and validate the functionality. All simulations and measurements were conducted using HSPICE, utilizing the 180-nanometer (nm) virtual process from TSRI at 25°C and 1.8V.

And then, we will explain the results of the logical operations. As seen in the various figures, before 5 microseconds (μs), the WWL is activated to allow data to be written into the memory. Subsequently, the WWL is deactivated, and the RWL is activated to enable data to be read and correctly generate the logical output. Therefore, the logical output results before 5 microseconds (μs) are not the final results.

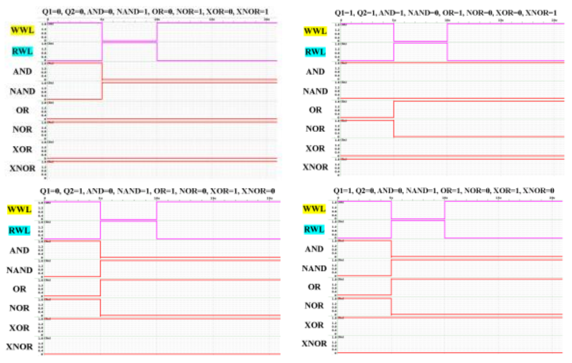

Fig.11 Logical Outputs Generated by 8+T SRAM in Various States

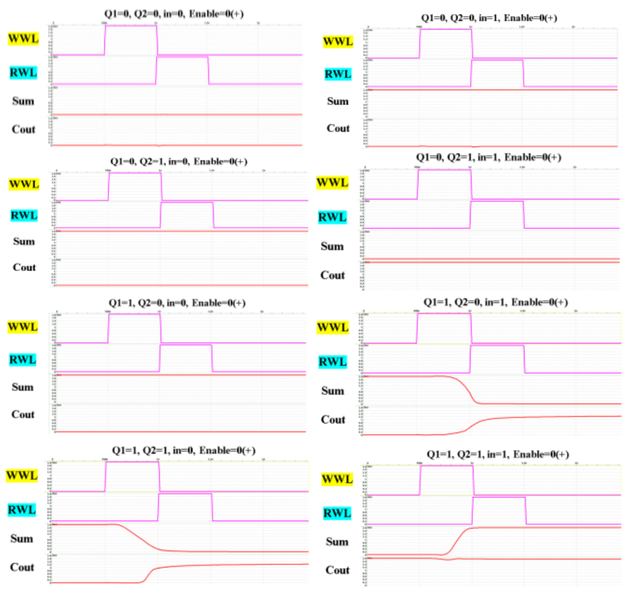

Fig.12 Addition Results Generated by the Proposed Architecture in Various States

The results from the above figures match those of the basic logic gates.

B. *Addition and Subtraction Operation Results*

Next, we will explain the results of the addition operations in the Fig. 12. By comparing the experimental results with the truth table of the full adder, it can be observed that the two results match each other.

Next, we will explain the results of the subtraction operations in the Fig.13. By comparing the experimental results with the truth table of the full subtractor, it can be observed that the two results match each other.

C. *Delay Time Results*

The propagation delay, measured in picoseconds (ps), is calculated from the activation of the WWL or RWL until the voltage reaches VDD/2 and from the initiation of operations (XOR, XNOR, Sum/Diff, Cout/Bout, AND, NAND, OR, NOR) until their output reaches VDD/2.

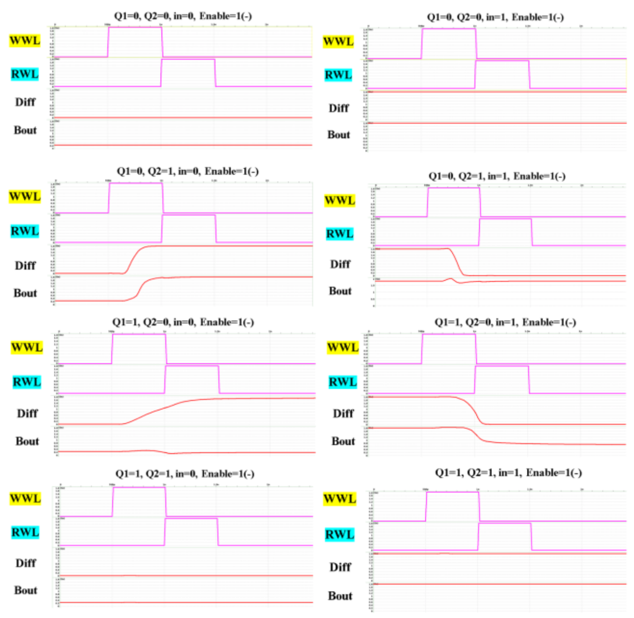

Fig. 13 Subtraction Results Generated by the Proposed Architecture in Various States

Table 4 Delay Time of the Proposed Architecture in This Paper (Unit: ps)

	Addition	Subtraction
AND	84.1875	82.9674
NAND	61.6961	60.2848
OR	63.1859	64.0917
NOR	85.8013	86.7333
XOR	249.7036	301.3825
XNOR	165.0656	238.0201
Sum/Diff	477.7527	492.1859
Cout/Bout	466.3265	630.9141

Table 5 Overall Power Consumption (Unit: μW)

Cin/ Bin	ADD/ SUB	Q1/Q2 Individual Write Values			
		(0,0)	(0,1)	(1,0)	(1,1)
in=0	ADD	48.6280	80.9560	84.4149	304.558
	SUB	49.5960	286.847	459.489	320.609
in=1	ADD	49.5421	80.9561	489.497	286.461
	SUB	49.5962	280.869	469.211	317.727
AVG		49.3405	182.407	375.653	307.338

Table 4 shows the maximum delay times for the proposed architecture in each state. In some states, there is no delay time. For Sum/Diff, Cout/Bout, XOR, and XNOR outputs, delays occur only in cases (1,0) and (1,1). In other states, the output values before and after WWL activation to VDD/2 are the same, resulting in no delay. For AND and NAND, there is no delay in case (1,1); for OR and NOR, there is no delay in case (0,0).

D. Power Consumption Calculation

This paper measures the average power consumption for each state when Q1, Q2, and Cin/Bin are individually written. The unit of measurement is fixed at microwatts (μW). Table 5 shows that overall power consumption peaks in the (1,0) state. This is likely due to the PTL architecture's primary drawback of lacking full swing at the output, leading to static power consumption. This issue can be mitigated using dual-edge-triggered flip-flops [20]. However, the architecture's advantage lies in performing both addition and subtraction through the exchange of two transistors, significantly reducing the required area. Future research will focus on reducing power consumption

V. Conclusion

The remarkable growth of artificial intelligence and deep learning has led to unprecedented data volumes, straining existing hardware. In-memory computing has thus become a critical research focus. While most research targets logical operations, the increasing demand for arithmetic operations is evident. This study explores using 8+T SRAM to enable both logical and arithmetic operations. Using Pass Transistor Logic (PTL) circuits, we designed an adder generating Sum and Carry-out outputs. Additionally, by examining adder and subtractor truth tables, we developed a dynamically switchable adder-subtractor.

Integrating computational functions within memory mitigates data transfer and processing delays, enhancing overall system performance. The dynamically switchable adder-subtractor offers high flexibility and scalability, addressing diverse computational needs and aligning with the performance demands of modern computing systems.

This research aims to provide valuable insights into advancing in-memory computing technologies and offers new perspectives on enhancing computer system performance and efficiency. Future research will optimize and apply this technology for broader applications and significant breakthroughs.

References

[1] Backus, J. (1978). Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. *Communications of the ACM*, 21(8), 613-641.

[2] Zou, X., Xu, S., Chen, X., Yan, L., & Han, Y. (2021). Breaking the von Neumann bottleneck: architecture-level processing-in-memory technology. *Science China Information Sciences*, 64(6), 160404.

[3] Elliott, D. G., Stumm, M., Snelgrove, W. M., Cojocaru, C., & McKenzie, R. (1999). Computational RAM: Implementing processors in memory. *IEEE Design & Test of Computers*, 16(1), 32-41.

[4] Gauchi, R., Kooli, M., Vivet, P., Noel, J. P., Beigné, E., Mitra, S., & Charles, H. P. (2019, October). Memory sizing of a scalable SRAM in-memory computing tile based architecture. In *2019*

IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC) (pp. 166-171). IEEE.

[5] Singh, J., Pradhan, D. K., Hollis, S., & Mohanty, S. P. (2008). A single ended 6T SRAM cell design for ultra-low-voltage applications. *IEICE Electronics Express*, 5(18), 750-755.

[6] Zhang, J., Wang, Z., & Verma, N. (2017). In-memory computation of a machine-learning classifier in a standard 6T SRAM array. *IEEE Journal of Solid-State Circuits*, 52(4), 915-924.

[7] Athe, P., & Dasgupta, S. (2009, October). A Comparative Study of 6T, 8T and 9T Decanano SRAM cell. *In 2009 IEEE Symposium on Industrial Electronics & Applications* (Vol. 2, pp. 889-894). IEEE.

[8] Ramani, A. R., & Choi, K. (2011, May). A novel 9T SRAM design in sub-threshold region. In 2011 *IEEE INTERNATIONAL CONFERENCE ON ELECTRO/INFORMATION TECHNOLOGY* (pp. 1-6). IEEE.

[9] Kulkarni, J. P., Goel, A., Ndai, P., & Roy, K. (2010). A read-disturb-free, differential sensing 1R/1W port, 8T bitcell array. *IEEE transactions on very large scale integration (VLSI) systems*, 19(9), 1727-1730.

[10] Agrawal, A., Jaiswal, A., Lee, C., & Roy, K. (2018). X-SRAM: Enabling in-memory Boolean computations in CMOS static random access memories. *IEEE Transactions on Circuits and Systems I: Regular Papers, 65(12),* 4219-4232.

[11] Song, S., & Kim, Y. (2021, October). Novel in-memory computing circuit using muller C-element. In *2021 18th International SoC Design Conference (ISOCC)* (pp. 81-82). IEEE.

[12] B. Holdsworth . (2002) Digital Logic Design.

[13] Singh, S. S., Leishangthem, D., Shah, M. N., & Shougaijam, B. (2020, November). A Unique design of hybrid full adder for the application of low power VLSI circuits. In *2020 4th international conference on electronics, communication and aerospace technology (ICECA)* (pp. 260-264). IEEE.

[14] Mahendran, G. (2024). CMOS full adder cells based on modified full swing restored complementary pass transistor logic for energy efficient high speed arithmetic applications. *Integration, 95*, 102132.

[15] Devi, S., Suhas, N. S., & Vijay, K. A. (2017, February). Design of full subtractor using DPL logic and MTCMOS technique to reduce the leakage current and area. In *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)* (pp. 1-4). IEEE.

[16] Ding, L., Zhang, Z., Pei, T., Liang, S., Wang, S., Zhou, W., ... & Peng, L. M. (2012). Carbon nanotube field-effect transistors for use as pass transistors in integrated logic gates and full subtractor circuits. *ACS nano, 6(5),* 4013-4019.

[17] Sharma, T., & Kumre, L. (2017, October). A comparative performance analysis of CMOS XOR XNOR circuits. In *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)* (pp. 473-478). IEEE.

[18] Frontini, L., Shojaii, S., Stabile, A., & Liberali, V. (2012, December). A new XOR-based Content Addressable Memory architecture. In *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)* (pp. 701-704). IEEE.

[19] Babu, H., KV, R. R., & Dhanabal, R. (2014, February). Comparative performance analysis of XOR-XNOR function based high-speed CMOS full adder circuits. In *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)* (pp. 432-436). IEEE.

[20] Hossain, R., Wronski, L. D., & Albicki, A. (1994). Low power design using double edge triggered flip-flops. *IEEE transactions on very large scale integration (VLSI) systems*, *2*(2), 261-265.