IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

DESIGN AND OPTIMIZATION OF 8-BIT ALU USING 180nm TECHNOLOGY

[1]Narayan A.Badiger, [2]Dnyaneshwar Jadhav, [3]Abhinandan Shetti, [4]Gayatri M, [5]Deeksha Mirji
[1]Associate Professor, [2 3 4 5] student
1 Department of Electronics & Engineering,
[1] S G Balekundri Institute of Technology, VTU Belagavi , Shivabasavnagar, Belagavi-590010

Abstract: The growing need for faster and more energy-efficient digital systems has made it crucial to design effective Arithmetic Logic Units (ALUs). Traditional ALU designs often struggle with issues related to power use, circuit complexity, and delays, especially as device sizes keep shrinking. This work details the design and implementation of an optimized 8-bit ALU using standard digital logic methods suitable for VLSI applications. The new architecture can perform a wide variety of arithmetic and logical operations while focusing on simpler hardware, lower delays, and better energy efficiency. A streamlined control unit is included to allow for smooth operation switching with minimal extra work. Simulation results show that the new ALU provides stable performance with lower power needs and reduced gate use compared to standard designs. These features make the proposed ALU a strong option for low-power embedded systems, high-speed processors, and modern digital computing platforms.

Index Terms - CMOS, 180nm, VLSI, Multiplexer, Logic Unit, Digital Design, ALU

I.Introduction

I. Introduction

The Arithmetic Logic Unit (ALU) is a key part of any digital processing system. It affects the speed, functionality, and efficiency of computations. As semiconductor devices get smaller and the demand for compact, low-power hardware rises, designing better ALUs has become crucial. Traditional ALUs are popular but often have drawbacks. They can consume more power, have longer delays, and use more gates if not designed properly. These issues are especially noticeable in embedded systems, portable devices, and real-time applications, where balancing efficiency and performance is essential. Boolean implemented using an adder–subtractor unit, basic Boolean operations, and multiplexer-controlled operation selection. The approach emphasizes reduced complexity, improved hardware utilization, and ease of implementation while maintaining full functionality suitable for VLSI, FPGA, and embedded system applications.This work focuses on designing a better 8-bit ALU using standard combinational and sequential logic components. Instead of using complex or impractical logic styles, the proposed architecture emphasizes simplicity, reduced hardware needs, and improved control logic to perform basic arithmetic and logical operations. By refining the internal data-path and cutting down on unnecessary transitions, the ALU achieves better performance and keeps power usage low. The design also aims for easy integration into modern VLSI systems and future low-power digital platforms. Overall, the presented ALU provides a practical and efficient solution for applications that need reliable computation, with minimized delay and lower circuit complexity.

II. Literature Survey

The design of ALUs has continued to evolve with the advancement of semiconductor technology and increasing demand for energy-efficient computing. Traditional ALU architectures used regular CMOS combinational logic, incorporating a ripple-carry adder and several multiplexers, which were functionally complete but often led to higher power dissipation and propagation delay. Sharma and Tiwari [6] proposed an early implementation of ALU using a structure based on a full adder and multiplexer, building the baseline for functional accuracy but leaving scope for improvement in power- and area-efficient design. As time progressed, the focus of research began to shift toward architecture and transistor-level logic optimization. Sarkar and Chatterjee [1] presented a design based on m-GDI logic, highlighting large savings in transistor count and significant power efficiency over conventional CMOS implementations.

Similarly, Subbulakshmi [2] proposed the concept of sliced ALU architecture, the key idea here lying in minimizing switching activity and hence dynamic power dissipation. This work emphasized how architectural fine-tuning could be achieved without compromising on the operational capability. Corresponding enhancements in computational efficiency were also illustrated by suggesting arithmetic operations. Kalaiselvi et al. [3] proposed an ALU based on the modular Booth encoder and Vedic multiplier, which demonstrated improved speed for multiplication workloads; because of this, it would be suitable for DSP and processor-based systems. Supritha et al. [7] further improved computational throughput by integrating a modified Booth multiplier into the arithmetic path and demonstrated improvement in delay, as well as switching efficiency. Low-power techniques remain a focus of critical research. Esther Rani et al. [12] proposed an area-optimized ALU architecture, targeting hardware redundancy reduction, while Amirthalakshmi and Raja [8] explored low-power CMOS design methodologies to reduce leakage currents and switching losses. Guan et al. [9] and studies in [10] evaluated ALU designs using logic minimization and optimized hardware sharing for improved scalability within embedded processors and ASIC platforms. The works that are more recent show a shift towards advanced digital platforms and AI-driven computation. Padmapriya et al. [12] proposed Cadence-optimized ALU which is appropriate for AI acceleration workloads. It has shown that power-efficient, ASIC-compatible designs are needed. Hybrid and approximate computing-based ALUs, proposed in recent research work [13], [14], [16], showed an increase in speed and power efficiency with slight deviation in accuracy, which can be appropriate to deploy in biomedical and IoT environments. These three factors mean that, despite considerable progress, an optimum balance between area, power consumption, and delay—particularly in mature nodes such as 180 nm CMOS technology—remains a design challenge. Motivated by this, the proposed work addresses a simplified and hardware-efficient 8-bit ALU architecture using standard digital logic, with an added focus on low-power and reliable operation for embedded and VLSI-based systems.

III. PROPOSED METHODOLOGY

The proposed method focuses on creating an efficient 8-bit Arithmetic Logic Unit with conventional digital logic parts. It minimizes hardware complexity and delay. The design uses a modular and structured approach. It starts by identifying the essential arithmetic and logical operations needed for an 8-bit processor environment. Inputs A, B, and Cin are processed based on three control lines, S0, S1, and S2 To implement the data-path, standard combinational circuits such as full adders, multiplexers, decoders, and logic gates are utilized. A dedicated arithmetic block is created to handle addition-based operations using cascaded full adders, while subtraction is achieved through two’s complement logic. Parallel to this, a logical operation block is developed using optimized gate-level implementations to ensure minimal propagation delay. Both blocks are interfaced through a compact multiplexer-based selection unit that chooses the final output based on the control signals.

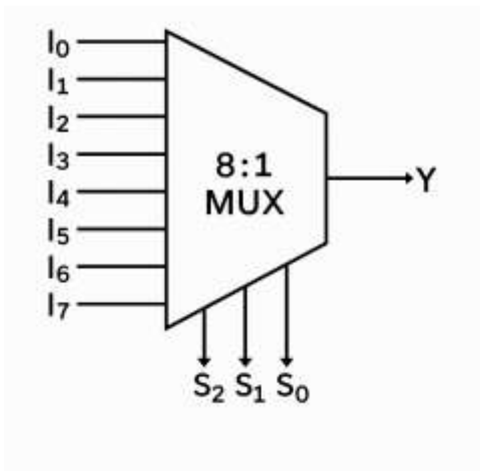

Fig.1. 8-bit ALU diagram

The block diagram of the proposed 8-bit ALU is divided into three main functional modules: the control unit, the arithmetic unit, and the logic unit, all built with standard digital logic. The ALU works on two 8-bit inputs and performs different arithmetic and logical functions based on the selection signals from the control unit. The control unit gets three selection lines (S0, S1, S2) and decodes them to produce the correct control signals for the data-path. Instead of figuring out reversible modes, S2 in this design chooses the operation category—either arithmetic or logical—while S0 and S1 select the specific operation within that category, like addition, subtraction, AND, OR, or XOR. This design allows for simple and efficient operation control with minimal hardware. The arithmetic unit uses standard full adders arranged in a ripple-carry setup to handle 8-bit addition and subtraction. It performs subtraction with two's complement logic, where input B is inverted and combined with the carry-in signal when necessary. This module is designed to reduce propagation delay and prevent unnecessary switching, which helps to keep power consumption low.

The logic unit carries out bitwise operations such as AND, OR, XOR, and NOT using optimized gate-level designs. Each operation is produced simultaneously, and the output is selected based on the control signals. This parallel setup ensures a quicker response time without adding complexity.A multiplexer at the output chooses between the arithmetic and logic results according to the signals from the control unit. This multiplexer uses standard digital multiplexers and guarantees that only the selected output goes to the final stage. The modular design makes it easy to scale and modify for additional operations if needed.

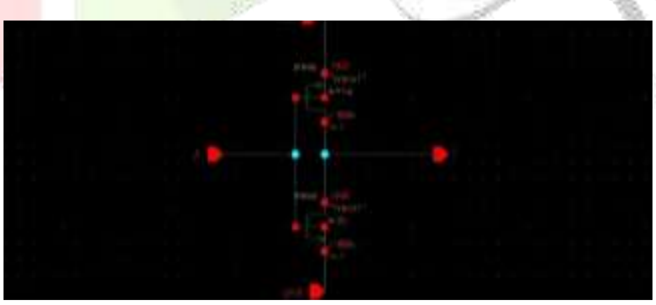

Fig 2. Inverter Schematic

This schematic illustrates a standard CMOS inverter created using one PMOS and one NMOS transistor arranged in a complementary structure. The PMOS device is placed at the top and connected to VDD, while the NMOS device is positioned at the bottom and connected to GND. Both transistor gates receive the same input signal A, ensuring opposite conduction states. When the input goes low, the PMOS switches on and pulls the output Y to a high level. but when the input is high, the NMOS turns on and changes the output to ground. The meeting point of the two transistors forms the inverter output node. This configuration provides strong noise immunity and low static power, making it ideal for digital logic design

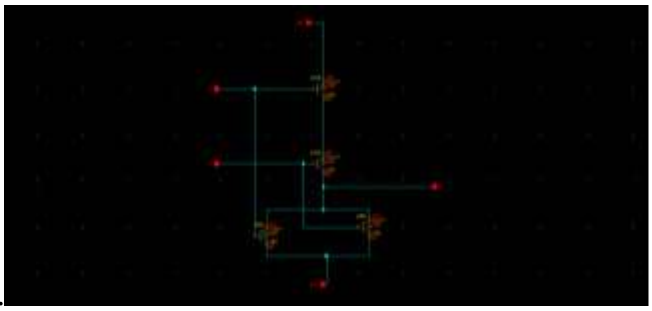

Fig 3. Nor Gate Schematic

This schematic represents a CMOS NOR gate built using two PMOS transistors in parallel at the top and two NMOS transistors in series at the bottom. Inputs A and B are connected to the gates of both PMOS and NMOS devices, ensuring complementary switching. When both inputs are low, the parallel PMOS network conducts and pulls the output node high. If either input becomes high, the series NMOS path activates and pulls the output to ground. This arrangement ensures the output is high only when both inputs are at logic zero, matching the NOR Boolean function. The layout also highlights clear separation of pull-up and pull-down networks for stable operation. Overall, this structure offers reliable logic behavior with low static power consumption typical of CMOS design.

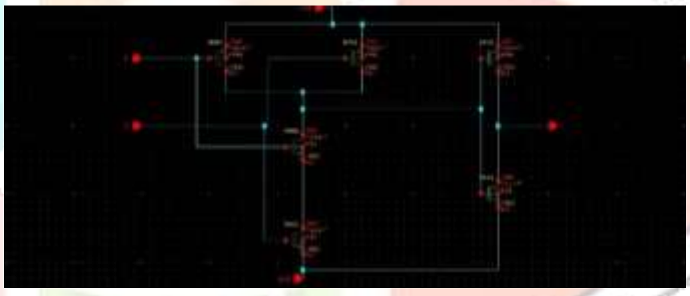

Fig 4. AND Gate Schematic

The schematic shows a CMOS AND gate designed in the Cadence Virtuoso environment. It uses a mix of PMOS and NMOS transistors arranged to produce a high output only when both inputs A and B are logic '1'. Two VPULSE sources provide dynamic digital inputs for timing and switching tests. The circuit runs on a 1.8V DC supply, while ground serves as the reference point. The output Y is seen at the drain connection shared by the transistor network. This schematic lets us evaluate the performance of the AND logic, including delay, power, and switching behavior. Such transistor-level design is essential in VLSI systems because logic-level operations form the foundation of complex digital circuits.

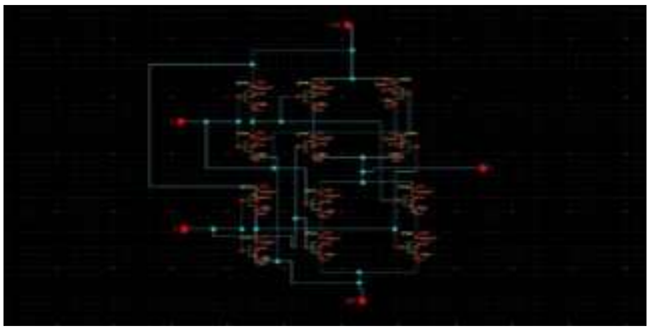

Fig 5. XOR Gate Schematic

The schematic shows a CMOS XOR gate built at the transistor level using the Cadence Virtuoso design suite. The circuit has several PMOS and NMOS transistors arranged in a complementary way to perform the exclusive OR function. This design produces a high output only when the two input signals are different, making it important for arithmetic and parity tasks. The network gets power from a VDD supply and connects to ground through VSS to support proper switching operation. Transistor stacking and series-parallel arrangements help generate the XOR logic efficiently. This schematic works well for examining delay, power loss, and device behavior in deep submicron VLSI design. XOR gates like this are key components in adders, ALUs, and digital signal processing systems.

.

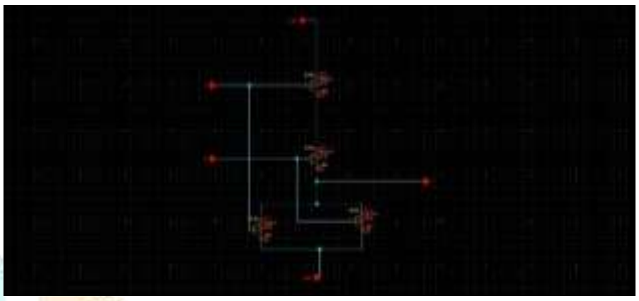

Fig 6. OR Gate Schematic

Figure illustrates the transistor-level implementation of a CMOS OR gate designed in Cadence Virtuoso. The pull-up network has parallel PMOS transistors, which keep the output high when any input is active. The pull-down network uses series NMOS devices, which together drive the output low only when both inputs are zero. This complementary setup ensures proper logic behavior while lowering static power use. The layout allows for adjustable transistor sizing, making it possible to optimize for speed, power, or area. The schematic also includes clearly defined VDD and GND rails for dependable biasing and signal integrity.

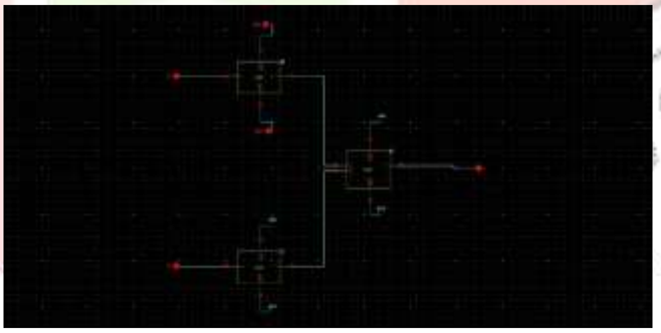

Fig 7. Demorgan’s Law Schematic

Figure shows the transistor-level realization of De Morgan’s theorem for an OR function. It uses basic CMOS logic blocks in Cadence Virtuoso. The circuit first inverts the individual inputs with two NOT gate stages, which ensures signal complementation. After inversion, these signals are combined with a NAND gate. This takes advantage of the principle that A + B = (A’B’)’. This structure simplifies the design by reusing standard cells and keeps the logical behavior correct. The placement of VDD and GND rails guarantees stable operation and proper logic transitions. This implementation demonstrates the practical use of De Morgan’s law in CMOS-based digital logic design.

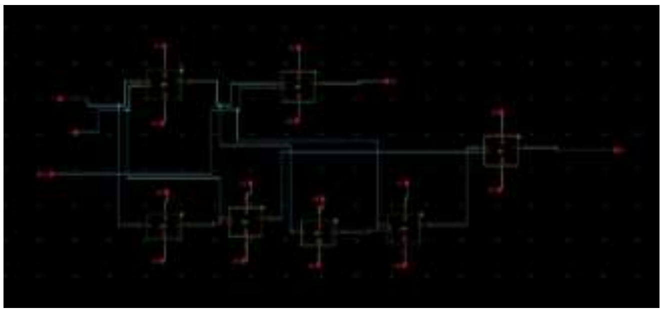

Fig 8. Full Subtractor Schematic

Figure presents a transistor-level schematic that implements a composite digital logic function using CMOS design in Cadence Virtuoso. The circuit includes XOR, AND, OR, and NOT blocks to create a multi-stage Boolean expression. Each logic stage connects to the next to progressively derive the final output, ensuring proper signal propagation and logic sequencing. CMOS inverters provide intermediate inversion, maintaining full logic swing and noise immunity. The modular structure allows for design reuse, scalability, and easier verification. VDD and GND rails provide stable switching and reliable operation across all logic blocks.

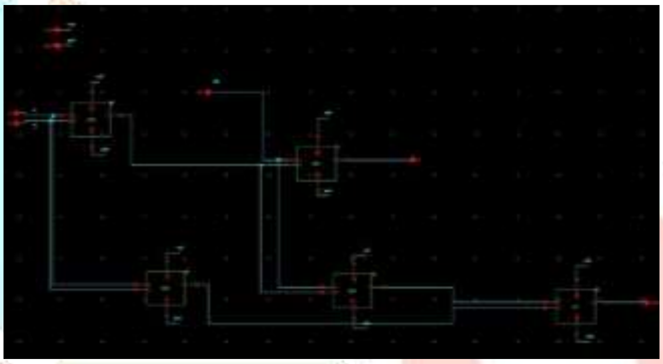

Fig 9. Full Adder Schematic

The figure shows the transistor-level schematic of a CMOS full adder created in Cadence Virtuoso. The design uses cascaded XOR gates to produce the sum output. The carry signal comes from AND and OR logic blocks. The modular structure allows for efficient computation of the sum and carry operations with minimal propagation delay. CMOS logic provides full swing outputs, low static power use, and better noise resistance. Interconnects between stages are optimized for signal quality, and the use of separate VDD and GND rails ensures stability. This implementation shows a practical and scalable method for designing arithmetic circuits in VLSI systems.

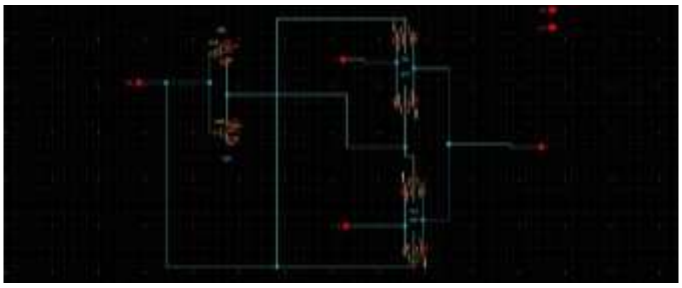

Fig 11 2:1 MUX Schematic

Figure shows the transistor-level schematic of a 2:1 multiplexer created with CMOS logic in Cadence Virtuoso. The circuit chooses between two input signals based on a single select line (S), performing the function Y = S·B + S’·A. The design uses complementary transmission gates made from parallel PMOS and NMOS devices to provide full rail-to-rail output and reduce signal loss. The select line and its inverted form control the switching network, allowing effective data routing with low static power use.

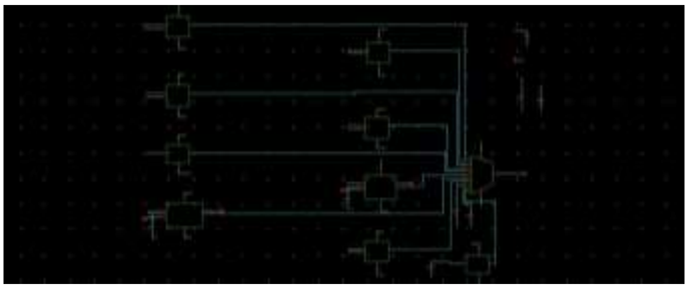

Fig 12. 8:1 MUX -Schematic with all Input Operations

The 8:1 multiplexer shown in the figure uses CMOS logic and is implemented in Cadence Virtuoso. It has eight data input lines (D0-D7), three select inputs (S2, S1, S0), and one output terminal. The select lines decide which single input will go to the output during each operation. To create the needed control signals, the select inputs are decoded with a mix of inverters and AND-based selection logic. This setup ensures that only one data path is active at a time, while all others stay in a high-impedance or OFF state. Each data path is controlled using complementary pass transistor logic, where PMOS and NMOS transistors function together as a transmission gate. When the condition for a specific input is high, the transmission gate activates and directly connects that input to the output line with minimal signal loss.
The complementary design of the gate ensures full voltage swing and reduces threshold drop issues that usually occur in NMOS-only pass-transistor logic. The multiplexer maintains low static power consumption because only one transmission gate is active at any time, and no direct path exists between VDD and GND during steady states. The use of a modular and symmetric layout also minimizes routing congestion and improves scalability for applications with higher-bit multiplexing.

This design is useful in VLSI systems where selecting signals, sharing resources, and routing data are critical. The 8:1 MUX can be expanded to wider data buses by connecting or grouping multiple units. It can also function efficiently at high frequencies because of its low propagation delay and transistor-level optimization.

IV. Conclusion

In this work, we designed and implemented an 8-bit Arithmetic Logic Unit using basic logic gates, an adder-subtractor structure, and control operations with multiplexers. The design performs various arithmetic and logical operations with a compact and organized circuit. By applying De Morgan's law in the logic section, we reduced the circuit's complexity while keeping it functionally correct, which led to a simpler gate-level realization. Using a multiplexer for operation selection provides flexibility and enables one hardware unit to support multiple operations without wasting resources. The proposed ALU is efficient, easy to implement on FPGA or ASIC platforms, and ideal for low-power digital systems and academic projects. Overall, the design shows a practical and optimized way to build an ALU while ensuring clarity, modularity, and operational accuracy.

REFERENCES

[1] A. Sarkar and S. Chatterjee, "8-bit ALU design using m-GDI technique," in *Proc. Int. Conf. Trends Electron. Inform. (ICOEI)*, 2020.

[2] N. Subbulakshmi, "ALUSGDI: Low-power arithmetic logic unit based sliced architecture," *Heliyon*, 2023.

[3] C. M. Kalaiselvi, et al., "A modular technique of Booth encoding and Vedic multiplier," *Scientific Reports*, 2023.

[4] D. K. J. Rajendiran, C. Ganesh Babu, K. Priyadharsini, et al., "Certain investigation on hybrid neural

network method for classification of ECG signal with suitable FIR filter," *Scientific Reports*, vol. 14, no. 15087, 2024. DOI: 10.1038/s41598-024-65849-w.

[5] M. K. Thomsen, R. Glück, and H. B. Axelsen, "Reversible arithmetic logic unit for quantum arithmetic," Dept. Comput. Sci., Univ. Copenhagen, Denmark.

[6] A. Sharma and R. Tiwari, "Low-power 8-bit ALU design using full adder and multiplexer," in *IEEE WiSPNET Conf.*, 2016.

[7] B. Supritha, K. Mannem, B. V. Reddy, K. Jamal, and M. O. V. P. Kumar, "High-speed and efficient ALU using modified Booth multiplier," in *IEEE I-SMAC*, 2018.

[8] T. M. Amirthalakshmi and S. S. Raja, "Design and analysis of low-power 8-bit ALU on reversible logic for nano processors," *J. Ambient Intell. Humaniz. Comput.*, 2018.

[9] Z. Guan, W. Li, W. Ding, Y. Hang, and L. Ni, "An arithmetic logic unit design based on reversible logic gates," in *IEEE Conf. Proc.*, 2011.

[10] "Design and performance evaluation of energy-efficient 8-bit ALU," *Semantic Scholar Conference Papers*, 2021–2023.

[11] S. Anusha, M. M. Rao, and N. S. Reddy, "Design, analysis, implementation and synthesis of 16-bit reversible ALU using Xilinx 12.2," *Int. J. Eng. Res. Appl.*, vol. 4, no. 4, pp. 86–91, Apr. 2014.

[12] T. Esther Rani, M. Asha Rani, and R. Rao, "Area-optimized low-power arithmetic and logic unit," *Journal of Electronics and Communication Engineering*, vol. 5, no. 1, pp. 13–22, 2015.

[13] S. Padmapriya et al., "Sustainable Low-Power ALU and Multiplexer based AI Accelerator — Cadence optimization," conference paper, 2024.

[14] S. Padmapriya, et al., "Sustainable low-power ALU and multiplexer-based AI accelerator—Cadence optimization," in *IEEE Conf.*, 2024.

[15] "Design and analysis of low-power and high-speed 8-bit ALU using hybrid full adder," *ResearchGate Preprint*, 2023.

[16] D. K. J. Rajendiran, C. Ganesh Babu, and K. Priyadharsini, "Examination on heterogeneous systolic array (HSA) design for deep learning accelerations with low-power computations," *Sustainable Computing*, vol. 44, p. 101042, 2024. DOI: 10.1016/j.suscom.2024.101042.

[17] D. K. J. Rajendiran, G. Babu, K. Priyadharsini, and M. Ramkumar, "Hybrid Radix-16 Booth encoding and rounding-based approximate Karatsuba multiplier for FFT computation in biomedical signal processing," *Integration*, vol. 98, p. 102215, 2024. DOI: 10.1016/j.vlsi.2024.102215