



DESIGN AND VERIFICATION OF AN OPTIMIZED 32-BIT PIPELINED MULTIPLIER FOR HIGH PERFORMANCE COMPUTING

C.S. Chakradhar¹, M. Sreedhar²

¹PG Scholar, Department of ECE, JNTUA College of Engineering, India.

²Assistant Professor Ad-hoc, Department of ECE, JNTUA College of Engineering, India.

¹<https://orcid.org/0009-0005-8609-9615>, ²<https://orcid.org/0000-0003-1111-7450>

Email: sricr324@gmail.com, sreedhar470@gmail.com

ARTICLE INFO

Article History

Received: March 2, 2026

Reviewed: June 10, 2026

Accepted: July 10, 2026

Published: July 31, 2026

Keywords:

Modified Booth Encoding

Pipelining

Radix-8

Partial products reduction

Delay

ABSTRACT

Arithmetic operations are fundamental to digital signal processing systems, where multipliers often decide overall performance constraints. They are key components of many high-performance systems such as Microprocessors, FIR Filters, Digital Signal Processors etc. The most common way of performing signed multiplication in digital circuits is by using booth multipliers, but the existing algorithm has a drain on power consumption since it never coerces operations to the full precision. In this paper, a novel 32-bit pipelined multiplier is designed aimed at achieving high throughput and low power consumption for VLSI applications. Modified Booth Encoding (MBE) with Radix-8 and Wallace tree reduction for partial products reduction and along with a CLA adder for partial products addition is used. Furthermore, a linear Pipelining technique with flipflops is implemented to minimize critical path delay. The Register Transfer level (RTL) model was implemented using Verilog and synthesized using Xilinx Vivado. Performance analysis demonstrates the reduction of delay by 23% and power consumption by 61%.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

The Arithmetic Logic Unit (ALU) is the fundamental computational unit in modern digital computers and signal processors, and multiplication has been one of the most time-consuming and hardware-intensive operations [1-3]. With the growing need for high-speed and real-time data processing in various applications like image processing, machine learning accelerators, and telecommunications, the need to have high throughput and energy-efficient digital multipliers has taken centre stage [4-6]. Three consecutive stages, namely partial product generation, partial product reduction, and final addition, are what define the performance of a digital multiplier [7]. To maximize the former, Modified Booth Encoding (MBE) ranks among the most commonly used techniques to minimize the number of partial products. While Radix-4 MBE halves the partial product rows, higher bit-widths (e.g., 32-bit or 64-bit) still cause serious routing loads. Radix-8 MBE brings the number of partial product rows down to $N/3$ (only 11 rows in a 32-bit operand), bringing the area of the next reduction step down to a minimum [8-10]. The implementation of this higher radix, however, poses the architectural challenge of generating the hard multiple (3A), which needs extremely optimized Digital Signal Processing (DSP) logic and routing [11]. To alleviate the build-up of delay during the reduction stage, the Wallace tree architecture is favoured over Carry-Look Ahead trees, as it avoids propagating carries across the horizontal logic path [12]. Moreover, the pipelining implementation in the multiplier architecture decouples combinational logic paths containing registers, improving the maximum operating rate considerably while concentrating switching activity, thereby decreasing dynamic power loss [13], [14]. Based on these architectural considerations, this paper presents the design and implementation of a high-speed 32-bit pipelined multiplier. The primary objective is to overcome the routing bottlenecks of standard multipliers by integrating Radix-8 MBE with a Wallace tree reduction structure. This study further implements a Carry Look-Ahead (CLA) adder for the final stage and utilizes a multi-stage pipelining technique to minimize critical path delay. The proposed architecture is modeled in Verilog HDL and synthesized using Xilinx Vivado to analyze delay and power efficiency.

II. LITERATURE REVIEW

II.1 BOOTH'S ALGORITHM

First introduced by Andrew D. Booth in 1950 in his crystallography work, the algorithm of Booth transformed the binary form of multiplication by identifying that mechanical shifting operations are computing faster than sequential additions [1], [15], [16]. The algorithm supports both signed and unsigned binary numbers equally using two complement representation. As it performs the operation, every bit of the multiplier yields a multiple of the multiplicand and this is added to the partial product accumulated. The overall latency of a standard multiplier is therefore dominated strongly by the number of addition operations to be made. To further increase the level of performance, parallel counting methods like Modified Booth Encoding (MBE) algorithm were created to provide a staple standard in the modern design of VLSI chips. Even though the standard Booth algorithm can efficiently and beautifully deal with signed multiplication, the performance of this algorithm is very dependent on the bit pattern of the multiplier. The algorithm is very efficient when the multiplier operand is a long sequence of continuous ones or zeros, since all it does in this case is to shift the partial product. Nevertheless, in the presence of runs of ones and zeros alternating (e.g. 01010101), the algorithm has to make an addition or subtraction operation at almost every bit in the sequence [1].

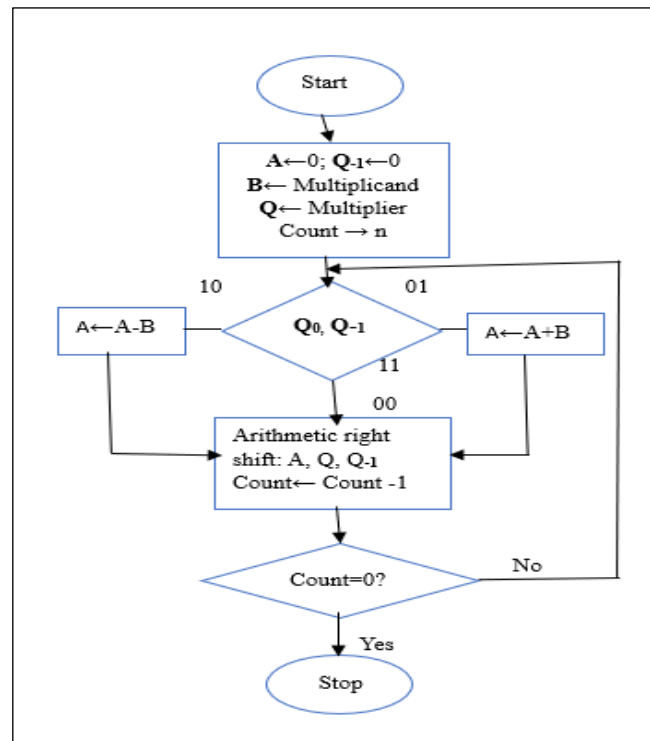


Figure 1: Flow Chart of Booth Multiplier.
Source: Authors, (2026).

II.2 MODIFIED BOOTH'S ALGORITHM (RADIX-4)

Radix-4 does an excellent job of reducing the number of partial products by a half compared to normal Booth algorithm [17]. It does this through multiplier bits evaluation in groups of three with overlapping functions. This 3-bit scanning window defines the current operands values, with the two leftmost bits as the current operands' values, and the third bit being the carry-in of the past two variables.

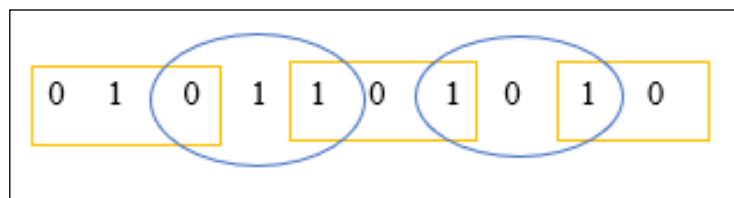


Figure 2: Grouping of multiplier bits in radix-4.
Source: Authors, (2026).

The encoding process reads starting at the Least Significant Bit (LSB) through the Most Significant Bit (MSB). In order to process the preliminary evaluation, the LSB is pre-padded with a ghost zero at position of $i-1$. The next step element of 3-bit window coincides with the MSB of the last group that was examined. The initial technique is this complementary procedure, which describes the arithmetic weights of the binary inputs, into a set of these five operative co-efficients: -2 , -1 , 0 , $+1$, and $+2$ [17].

Table 1: Radix-4 booth encoding.

Block	Recoded digit		Operation on 'A'
000	0		0A
001	+1		+1A
010	+1		+1A
011	+2		+2A
100	-2		-2A
101	-1		-1A
110	-1		-1A
111	0		0A

Source: Authors, (2026).

The initial step produces a partial product matrix with the MSB determining the arithmetic weight, which is 0 (positive) with unsigned operations and 1 (negative) with two's complement arithmetic. This is implemented as logical AND operations with the input vectors. At this final step, these partial products in completed form are reduced to two separate vectors (sum and carry) now used to calculate the final product (P).

III. PROPOSED METHODOLOGY

III.1 RADIX-8 MODIFIED BOOTH'S ALGORITHM

Although Radix-4 reduces the partial products by half, a 32-bit operand gives 16 rows. Hence, a significant limitation remains. The large volume causes a large routing bottleneck, and huge critical path delay and limits the design to low-frequency operations [8]. In order to eradicate this bottleneck, sophisticated partial product reduction schemes are necessary, and the long combinational path way is to be subdivided into more stages.

A. Modified Booth Encoding and PP Generation

To achieve the proposed architecture, Radix-8 MBE has been implemented that further reduces the operands into 11 PP rows [7], [8], [18], [19]. In this architecture, the bits of the multiplier are divided into groups of four, with one bit overlap between the two groups. The number of bits to shift the multiplicand left is strictly calculated as

$$\text{Shift} = \text{group index} \times 3 \quad (1)$$

(e.g. if Group 2 has bits 1110, then $-1A \ll 2(6)$), where A is the multiplicand.

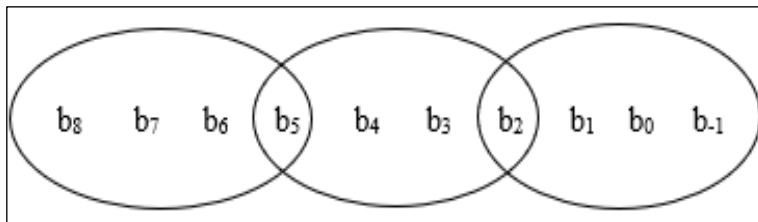


Figure 3: Grouping of multiplier bits with radix-8.

Source: Authors, (2026).

The multiplicand (A) is multiplied using a control signal or an instruction depending on the bits of the multiplier of each group. These rules are usually ordered between $\{-4, +4\}$ radix-8 as in the table below.

Table 2: Radix-8 booth encoding.

Group	Recoded digit	Operation on 'A'
0000	0	0A
0001	+1	+1A
0010	+1	+1A
0011	+2	+2A
0100	+2	+2A
0101	+3	+3A
0110	+3	+3A
0111	+4	+4A
1000	-4	-4A
1001	-3	-3A
1010	-3	-3A
1011	-2	-2A
1100	-2	-2A
1101	-1	-1A
1110	-1	-1A
1111	0	0A

Source: Authors, (2026).

Upon the generation of the 11 PPs, they are entered into the pipeline register that is usually D-flipflop 1. It is essentially a storage device, which stores and synchronizes the data to use in the following clock cycle. The 11 PP rows are then loaded into our proposed Wallace tree 3:2 compressor reduction method. These 11 PP rows are trimmed to leave less load on the final CLA adder which makes the product easier to reduce and calculate.

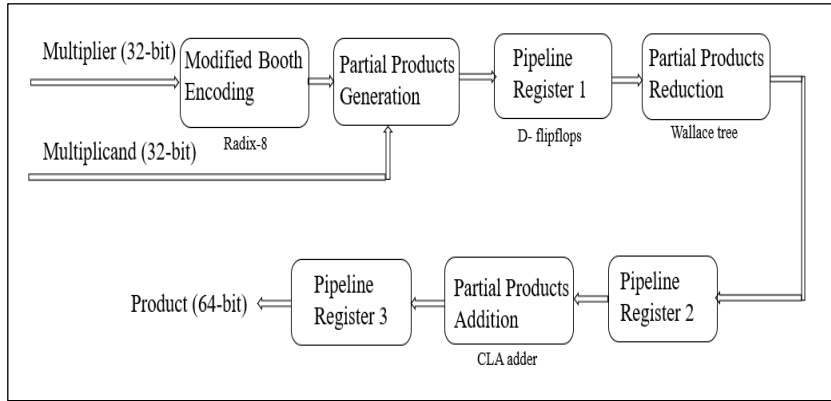


Figure 4: Modified Multiplier block diagram with pipelining and Wallace tree.
Source: Authors, (2026).

B. Pipelining

Pipelining is a structural optimization technique that improves the throughput in a calculation by inserting storage elements (pipeline registers) between diverse processing steps [13], [20]. These types of D-flip-flop registers merely grab a bit of in-between data and then one can select blocks of combinational logic to be isolated synchronously. Each set of operands must be processed in more than the usual Radix-4 algorithm in the proposed architecture, whereby 32-bit data makes sure that the data is continually flowing in a stream with multiple clock cycles necessary to complete each set of operands.

C. Wallace Tree Structure

This structure is firstly introduced by Chris Wallace in 1964, [12] to compress through Carry-Save Adders (CSAs) rather than the classical ripple-carry propagation [21-23]. In this compression step, 3:2 compressors are to be used. The length of two vectors Sum and a Carry is squeezed into three (n-bit) input vectors (e.g. A, B and C) using a CSA. A Carry Save Adder computes the sum of 3 or more n-bit numbers and gives 2n-bit numbers and will result two numbers of same length called Sum(S) and Carry(C). For three input bits A, B, and C that come from the partial product rows, the sum and carry is:

$$i. \quad \text{Sum (S)} = A \oplus B \oplus C \quad (2)$$

This is an XOR operation. It checks if the number of 1's is odd.

$$ii. \quad \text{Carry (C}_{out}) = (A \cdot B) + (B \cdot C) + (C \cdot A) \quad (3)$$

Where these two sum and carry vectors are 64-bit vectors each that will be fed to stage 2 pipeline and the values will be held and synchronized and eventually presented to the Carry Look Ahead (CLA) adder to add the final 64-bit product.

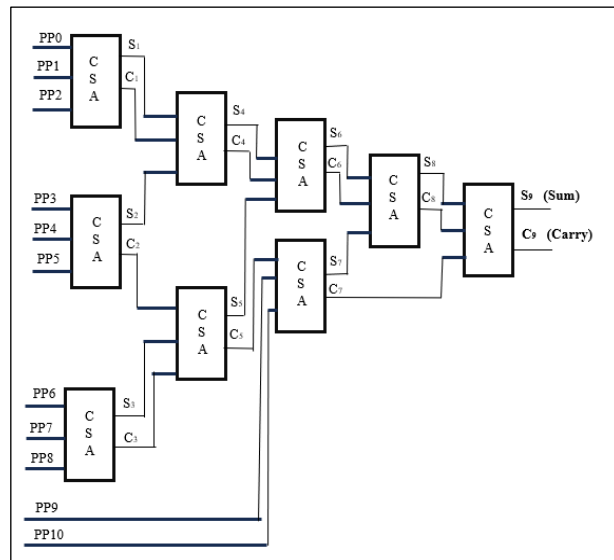


Figure 5: Partial Products Reduction by 3:2 compressor Wallace tree with CSA.
Source: Authors, (2026).

D. Carry Look-Ahead(CLA) Adder

A CLA adder is used to compute carry bits simultaneously with the other bits instead of having to wait until previous computations are done with the result. In each bit position, each logic tests two conditions on whether the bit is being calculated depending on the inputs A_i and B_i [8], [24], [25].

- i) Generate (G_i) : $A_i \cdot B_i$ (both i/p's are 1)
- ii) Propagate (P_i) : $A_i \oplus B_i$ (one i/p is 1)

If we have both G and P values for a bit pair, we can calculate the Carry out (C_{out}) as follows:

$$C_{out} = G_i + P_i \cdot C_{in} \quad (4)$$

To calculate the C_{in} , assume the initial C_0 is Zero. Then the carry 1 for bit 1 is

$$C_1 = G_1 + (P_0 \cdot C_0). \quad (5)$$

After the 64-bit product is produced, it will be sent to final pipeline stage for steadiness and synchronizing for next clock cycle. The whole process of the proposed architecture is controlled by the positive-edge triggered clock signal [14], additionally in the three stages in the following way:

Clock Cycle 1: MBE block samples the multiplicand and multiplier. On the first clock cycle, the 11 latched PP rows are calculated and latched to Pipeline Register 1, and

Clock cycle 2: The Wallace tree reduces them to two vectors (Sum and Carry), which are latched to the Pipeline Register 2 memory. At the same time, Stage 1 initiates the next group of operands processing.

Clock Cycle 3: The CLA adder calculates the final 64-bit product of the two vectors which gets stabilized in the last output register at the third clock step.

IV. RESULTS AND DISCUSSIONS

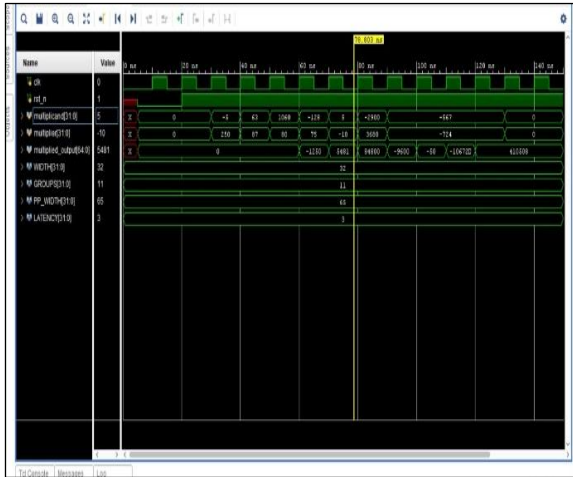


Figure 6: Output in signed values.
Source: Authors, (2026).

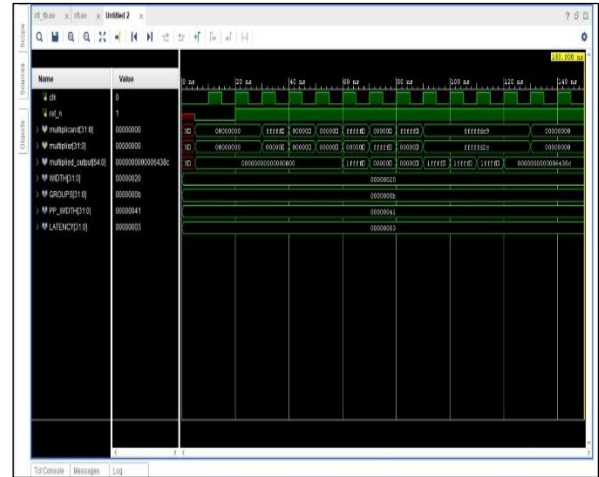


Figure 7: Output in hexadecimal values.
Source: Authors, (2026).

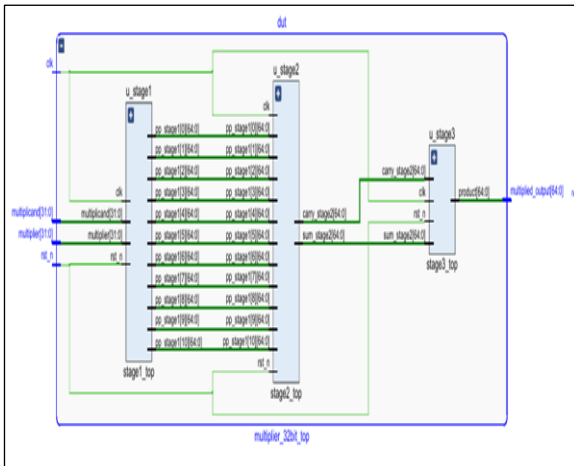


Figure 8: Schematic diagram 1.
Source: Authors, (2026).

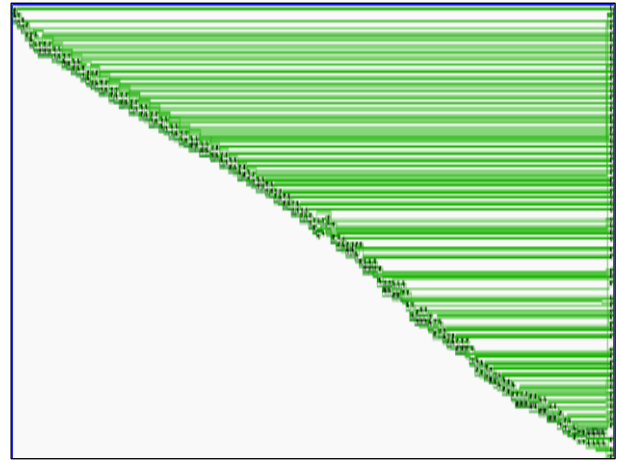


Figure 9: Schematic diagram 2.
Source: Authors, (2026).

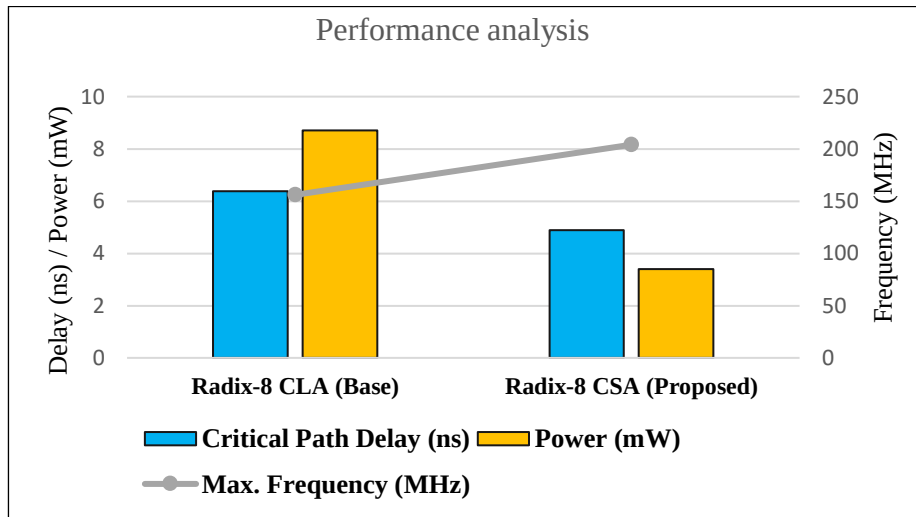


Figure 10: Performance analysis of multipliers.
Source: Authors, (2026).

The synthesis results demonstrate highly efficient hardware utilization. The Critical path delay was recorded at 4.895 ns, yielding a maximum operating frequency ($F_{\max}$) at 204 MHz. The total on-chip power at this frequency was restricted to 3.39 mW. A detailed power breakdown reveals that dynamic power dominates, with pipeline registers accounting for 52.32% of the total power and the combinational logic accounting for 46.16%.

V. CONCLUSIONS

This paper presented the design, implementation, and verification of a high-performance 32-bit multiplier utilizing a 3-stage pipelined Radix-8 Modified Booth Encoding (MBE) architecture. The proposed design successfully mitigated traditional routing congestion and minimized the critical path delay by replacing conventional linear reduction schemes with a Carry Save Adder (CSA)-based Wallace tree structure, which accelerated the compression of the 11 partial product rows. Synthesis results demonstrate that the architecture achieves a delay of 4.895 ns and power consumption of 3.39 mW, representing a 23% reduction in delay and a 61% improvement in power efficiency compared to standard designs. These results confirm that the proposed multiplier architecture is highly suitable for integration into modern DSP systems and high-speed VLSI applications.

VI. AUTHOR'S CONTRIBUTION

Conceptualization: C.S. Chakradhar, M. Sreedhar.

Methodology: C.S. Chakradhar, M. Sreedhar.

Investigation: C.S. Chakradhar, M. Sreedhar.

Discussion of results: C.S. Chakradhar, M. Sreedhar.

Writing – Original Draft: C.S. Chakradhar, M. Sreedhar.

Writing – Review and Editing: C.S. Chakradhar, M. Sreedhar.

Resources: C.S. Chakradhar, M. Sreedhar.

Supervision: C.S. Chakradhar, M. Sreedhar.

Approval of the final text: C.S. Chakradhar, M. Sreedhar.

VII. REFERENCES

- [1] N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Boston, MA: Addison-Wesley, 2010.
- [2] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd ed. Pearson, 2003.
- [3] IEEE Standard for Verilog Hardware Description Language, IEEE Std 1364-2005, 2006.
- [4] Texas Instruments, "Digital Signal Processing Solutions and Architectural Advantages," TI Application Report, SPRU041, 2021.
- [5] Y. C. Tseng, et al., "Area-Power-Delay-Efficient Multi Modulus Multiplier Based on Area-Saving Hard Multiple Generator Using Radix-8 Booth-Encoding Scheme," Electronics, vol. 13, no. 2, 2024.
- [6] M. Waris et al., "AxBMs: Approximate Radix-8 Booth Multipliers for High-Performance FPGA-based Accelerators," IEEE Transactions on Circuits and Systems II: Express Briefs, 2021.
- [7] A. Sharma and P. Kumar, "Design and Analysis of Multipliers using Radix-8 Booth Encoder," International Journal of Engineering Research & Technology (IJERT), vol. 10, no. 5, 2021.
- [8] R. Kumar, et al., "Design and implementation of RADIX-8 based 32-bit pipelined multiplier by using CLA," Journal of Engineering and Applied Sciences, 2017.
- [9] R. Orugu, "FPGA Design and Implementation of Approximate Radix-8 Booth Multiplier," 3rd International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), IEEE, 2024.

- [10] S. R. Kuang et al., "Factored Systolic Arrays Based on Radix-8 Multiplication for Machine Learning Acceleration," IEEE Transactions on Computers, 2024.
- [11] Xilinx Inc., "Ultra Scale Architecture DSP Slice User Guide," UG579 (v1.10), AMD Adaptive Computing, 2021.
- [12] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Transactions on Electronic Computers, vol. EC-13, no. 1, pp. 14-17, 1964.
- [13] M. Patel and S. Gupta, "Design of energy-efficient multiplier based on 32-bit architectures," International Journal of Advanced Technology and Engineering Exploration, 2022.
- [14] S. Rao, "A High-Speed 32-bit Signed/Unsigned Pipelined multiplier," International Journal Microelectronics and Computer Science, 2023.
- [15] O. L. MacSorley, "High-speed arithmetic in binary computers," Proceedings of the IRE, vol. 49, no. 1, pp. 67-91, 1961.
- [16] A. D. Booth, "A signed binary multiplication technique," The Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, no. 2, pp. 236-240, 1951.
- [17] K. Reddy, "Implementation of Radix-4 32-bit Booth Multiplier using VHDL," IJERT, vol. 12, no. 6, 2023.
- [18] K. L. Reddy et al., "Compressor-Based Precise Blended Wallace Tree Multiplier Architecture for Signal Processing Applications," 4th International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT), IEEE, 2022.
- [19] T. N. V. Krishna et al., "Enhancing the Efficiency of Wallace Tree Multipliers Through Optimized ECSLA Design," IEEE International Conference, 2023.
- [20] M. V. Ramana et al., "Delay Enhancement of Wallace Tree Multiplier with Binary to Excess-1 Converter," IEEE International Conference, 2023.
- [21] V. Kumar, "Design and Implementation of Pipelined Wallace-Tree Multiplier," International Journal of Computer Engineering and Technology (IJCET), 2024.
- [22] L. Dadda, "Some schemes for parallel multipliers," Alta Frequenza, vol. 34, pp. 349-356, 1965.
- [23] P. S. Rao et al., "VLSI Design of Majority Logic based Wallace Tree Multiplier," IEEE International Conference, 2023.
- [24] M. Ali, "Design of Power and Area Efficient Approximate Multipliers for DSP Applications," Journal of Electronic Testing, 2022.
- [25] M. Pedram et al., "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 2017.