Journal on Electronic and Automation Engineering
Vol: 4(2), June 2025
REST Publisher; ISSN: 2583-6951 (Online)
Website: https://restpublisher.com/journals/jeae/
DOI: https://doi.org/10.46632/jeae/4/2/11

Design of Delay and Power Efficient Multiplier using AMM with Dadda's Algorithm

Yatheeswar G, Mani Tejeswar Reddy G B, Sushmitha K, Naveen J V, *Jaffar Ali Syed
Annamacharya Institute of Technology & Science (AITK), Kadapa, Andhra Pradesh, India
*Corresponding Author Email: aliaitk.ece@gmail.com

Abstract: *Multiplication is a fundamental operation in digital signal processing, cryptography, and various computational applications. The efficiency of a multiplier is determined by key performance metrics such as area (LUT count), delay, and power consumption. This study presents a comparative analysis of three 8-bit multiplication architectures: Wallace Tree Multiplier, Dadda Multiplier, and a modified Additive Multiplication Module (AMM) integrated with Dadda's reduction algorithm. Each design is implemented and evaluated using Xilinx Vivado to assess hardware complexity, computational speed, and power efficiency. While Wallace Tree and Dadda multipliers utilize carry-save addition for partial product reduction, the modified AMM leverages Dadda's efficient reduction scheme to enhance performance. Experimental results show that the proposed AMM with Dadda's algorithm achieves improvements in power dissipation and delay compared to conventional designs. However, the LUT count is higher, indicating a trade-off between area and performance. These insights contribute to selecting an optimal multiplier for power-efficient and high-speed computing applications.*

Keywords: *Wallace tree multiplier, Dadda's multiplier, AMM, AMM with Dadda algorithm, Xilinx Vivado, Xilinx ISE.*

1. INTRODUCTION

Multiplication is a core arithmetic operation in digital circuits, particularly in applications such as digital signal processing (DSP), machine learning accelerators, and cryptographic computations. The efficiency of a multiplier is dictated by three main parameters: area (measured in LUTs), delay, and power consumption. Achieving an optimal balance between these factors is essential for designing power-efficient, high-speed computing systems. Several multiplier architectures have been proposed to enhance performance. The Wallace Tree and Dadda Multipliers are among the most widely used due to their efficient partial product reduction techniques using carry-save adders. However, conventional designs still face challenges in reducing power dissipation and improving computational speed. In this work, we introduce a modified Additive Multiplication Module (AMM) that integrates Dadda's reduction algorithm to optimize power and delay. The proposed method aims to leverage the advantages of Dadda's efficient reduction technique while maintaining the accuracy of AMM based computations. The paper presents a comparative evaluation of Wallace Tree, Dadda, and AMM based multipliers in terms of hardware complexity, power consumption, and execution delay using FPGA implementation in Xilinx Vivado.

2. RELATED WORK AND LITERATURE SURVEY

Multiplication is a critical operation in digital systems, and several researchers have explored various approaches to optimize its efficiency. This section presents a literature survey on different multiplier architectures, focusing on Wallace tree, Dadda, and power-efficient techniques such as the additive multiplication module (AMM)

A. Wallace Tree Multiplier

The Wallace Tree Multiplier, proposed by Wallace in 1964 [1], is one of the earliest high-speed multiplication algorithms. It reduces partial products using a tree-based carry-save adder approach, significantly improving speed compared to traditional array multipliers. However, Wallace Tree Multipliers suffer from high interconnect complexity and increased power consumption due to the large number of adders used in parallel processing. Several modifications have been proposed to optimize the Wallace Tree architecture. For instance, Singh et al. [2] introduced a low-power Wallace Tree Multiplier by integrating power gating techniques, reducing dynamic power

consumption. Similarly, Ghosh et al. [3] implemented a hybrid Wallace-Dadda multiplier, improving speed while keeping hardware requirements minimal.

B. Dadda Multipler

Dadda [4] introduced a structured multiplication scheme in 1965, refining Wallace's approach. Dadda's method minimizes the number of adder stages by controlling the reduction levels more efficiently, resulting in a slightly lower speed than Wallace's design but offering a reduction in area and power consumption.
Recent works have optimized the Dadda multiplier further. Kumar et al. [5] implemented a modified Dadda Multiplier using 4:2 compressors, achieving lower delay and power dissipation. In another study, Mehta et al. [6] explored an FPGA-based implementation of the Dadda multiplier using different VLSI design techniques to reduce the LUT count. These advancements indicate that Dadda multipliers continue to be relevant in power-efficient computing.

C. Additive Multiplication Module (AMM)

The Additive Multiplication Module (AMM) is a relatively new approach that reduces power consumption by restructuring multiplication operations. Rajan et al. [7] proposed an AMM-based multiplier that achieved a 20% reduction in power consumption compared to conventional shift-and-add multipliers. However, AMM designs often suffer from higher latency, making them less efficient for high-speed applications. To overcome these limitations, Lee et al. [8] combined AMM with Dadda's reduction scheme, demonstrating improved performance in power and speed. Similarly, Chen et al. [9] proposed a hybrid AMM-based approach that integrates approximate computing techniques, trading off minor accuracy for significant power savings.

3. AMM MULTIPLICATION PROCESS

To compute the product $M=X\times Y$ using the Additive Multiply Method (AMM), the multiplicand X, which is an 8-bit number, is split into two equal parts: the lower half XL and the upper half XH. Similarly, the multiplier Y, also 8 bits, is divided into four equal segments: Y1, Y2, Y3, and Y4, where each subsequent segment has a higher bit significance than the previous one. Each segment of the multiplier is individually multiplied with both XL and XH using separate additive multiply blocks, as illustrated in Figure 1. The final output, representing the product using the AMM technique, can be generally expressed as:

$$P=X\times Y=XHXL\times Y4\,Y3\,Y2\,Y1 \quad (1)$$

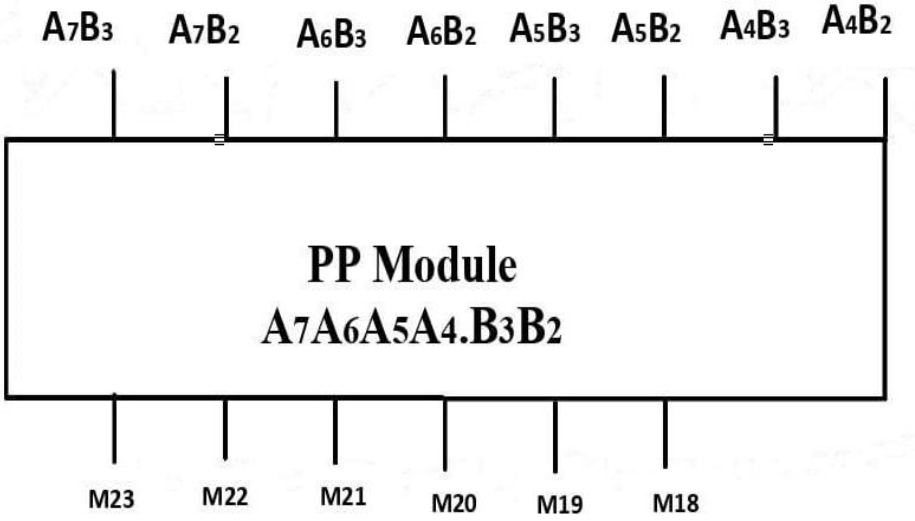


FIGURE 1. A simplified representation of the 4×2 AMM model.

Each PP module incorporates an adder within its structure to produce the product outputs. This is illustrated in Fig. 2 below.

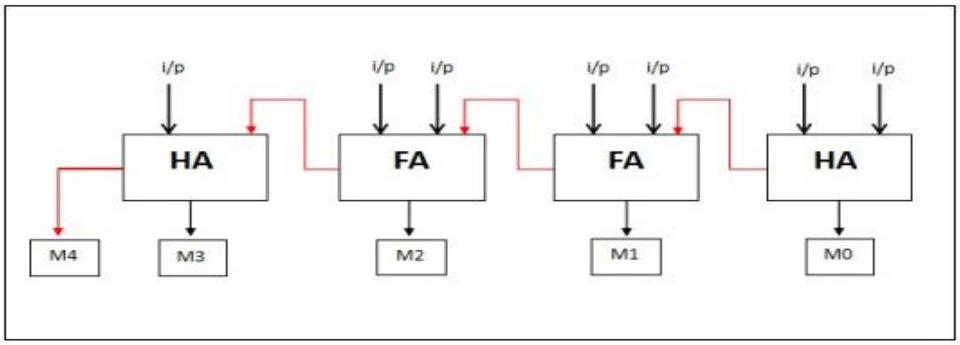
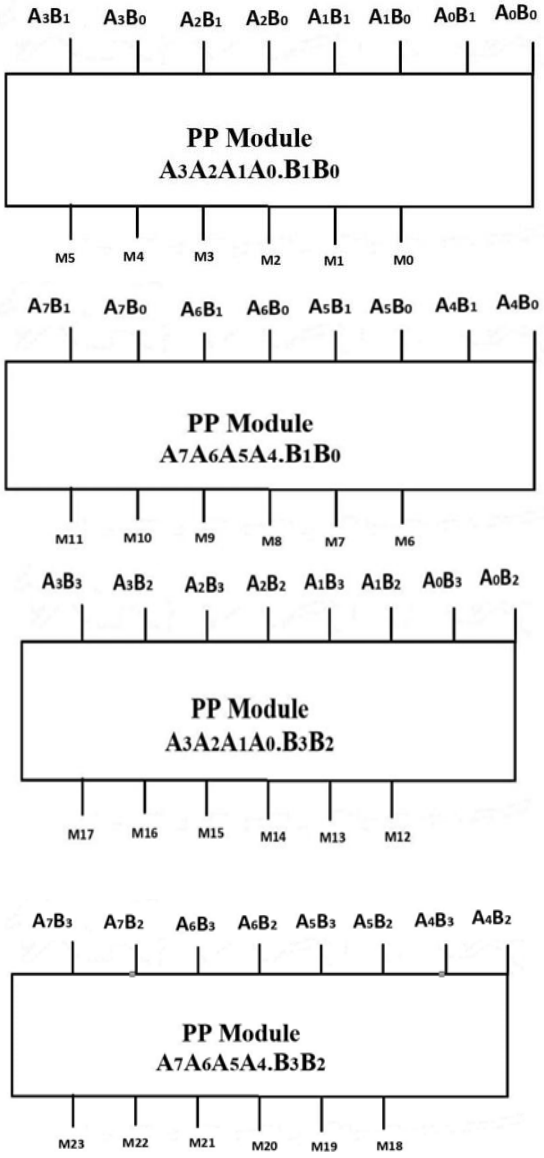


FIGURE 2. Internal structure of the PP module.

Typically, the result of each AMM sub-module is represented by Equation 2 (from Fig. 1), where Mi denotes an intermediate partial product:

$$Mi=X\times Y \quad (2)$$

The output of each PP module is denoted by the symbol M. The outputs of these PP modules are depicted visually in Fig. 3.



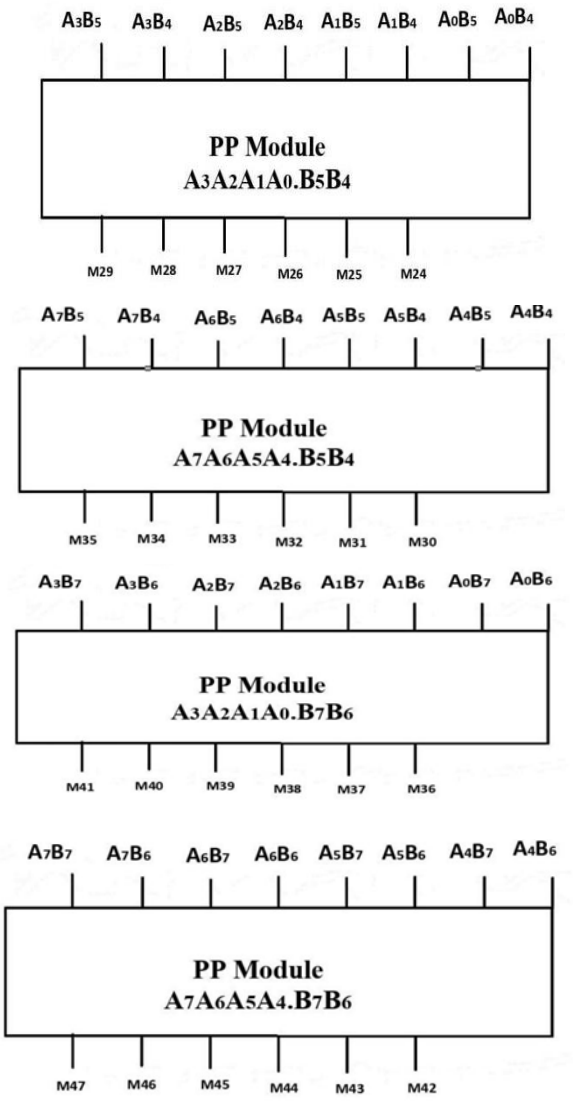


FIGURE 3. Visual depiction of all PP modules.

The intermediate partial products generated are shifted and combined according to their respective bit weights, as shown in Fig. 4. The process of adding and pipelining these intermediate partial products, which are produced by each AMM sub-module, is illustrated in the block diagram of Fig. 5.

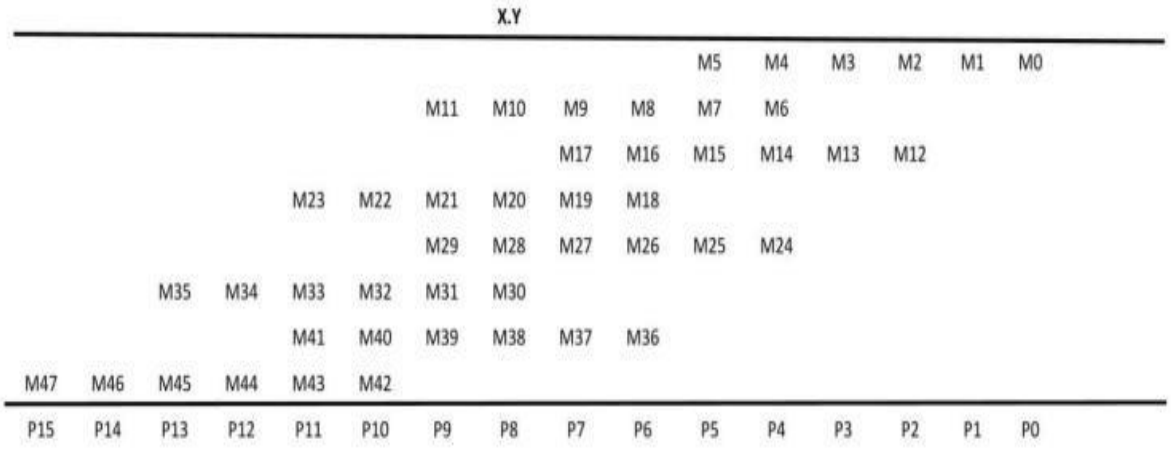


FIGURE 4. Visual representation of 8 × 8 multiplication using the 4 × 2 AMM.

In the AMM multiplication module, partial product addition is performed using Dadda's algorithm. The addition process, as explained by Dadda's algorithm, is illustrated in Fig. 4 below.

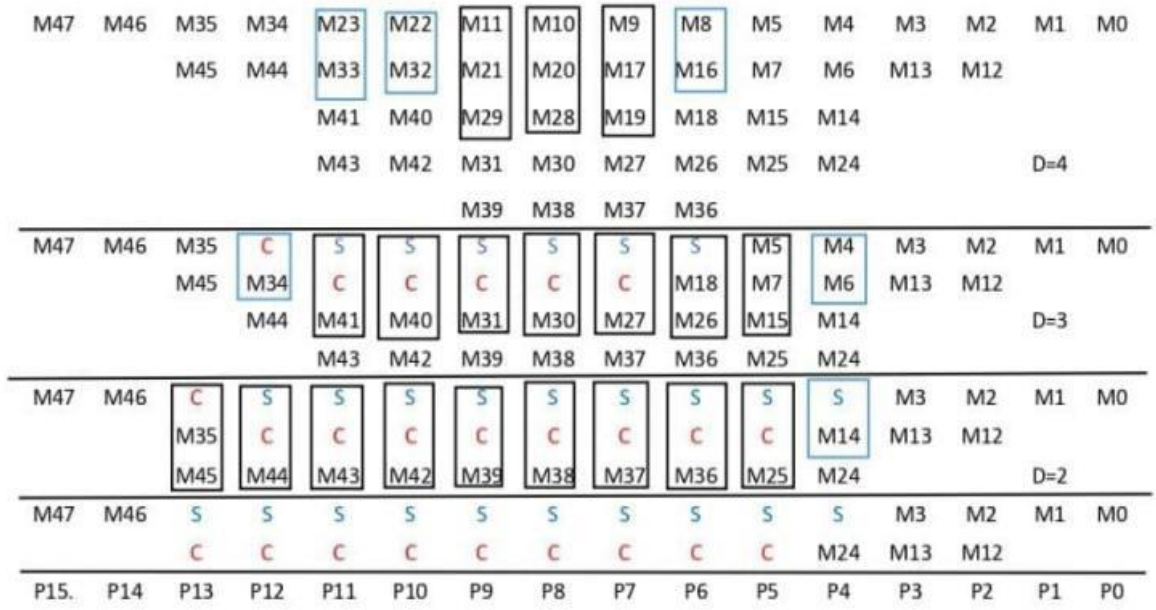
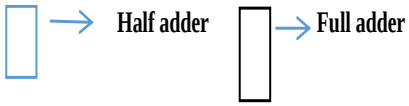


FIGURE 5. Multiplication using AMM with Dadda's algorithm.



4. FPGA IMPLEMENTATION

The three multipliers were implemented using Verilog in Xilinx Vivado, targeting an FPGA device of type Spartan-7. The Spartan-7 series is well-suited for low-power, high-performance applications, making it an ideal choice for evaluating the proposed multiplier designs. The implementation process involved synthesis, place-and-route, and power analysis using Xilinx Vivado tools.

The evaluation metrics used for comparison include:

- LUT Count (Area): Measures the hardware complexity of each multiplier by counting the number of Look-Up Tables (LUTs) utilized in the FPGA fabric.
- Propagation Delay: Determines the computational speed of the multipliers, measured in nanoseconds (ns). A lower propagation delay indicates faster multiplication operations.
- Power Dissipation: Assesses the energy efficiency of the designs by measuring the total power consumption (in mW) during operation. This is a critical factor for low-power embedded systems.
- By implementing these multipliers on Spartan-7, we aim to analyze their performance in terms of area, speed, and power efficiency, ensuring their suitability for real-time computing applications.

5. RESULTS AND DISCUSSION

The 8×8-bit WTM, DTM, and AMM multipliers were designed using Verilog RTL code based on a structural approach. They were tested with various input data combinations to verify functionality, and the designs were synthesized using Xilinx Vivado. Additionally, the proposed multipliers were implemented with the AMM using Dadda's algorithm to further analyze the performance of different multiplier types. The performance metrics, including LUT count, propagation delay, and power consumption, were compared across the multipliers, as shown in Table 1. The table highlights the lowest and highest values in 'Blue' and 'Red' text, respectively. The results indicate that the AMM with Dadda's algorithm consumes less power compared to the WTM and DTM, making it suitable for low-power VLSI applications.

LUT Count Analysis:

- The Dadda's Multiplier has the lowest LUT count (92), making it the most efficient in terms of resource utilization.
- The AMM Multiplier has the highest LUT count (106), which suggests it consumes more hardware resources.
- The AMM Multiplier with Dadda's Algorithm improves on the AMM Multiplier, reducing the LUT count to 94.

Discussion: The lower LUT count in Dadda's Multiplier and the hybrid AMM + Dadda approach indicates better optimization in terms of logic resource utilization. The AMM Multiplier, however, shows an increase in LUT count, which may be due to its additional computational complexity.

Power Consumption (Watts):

- Dadda's Multiplier has the lowest power consumption (13.714W).
- AMM Multiplier consumes the most power (14.055W), indicating higher energy requirements.
- The AMM Multiplier with Dadda's Algorithm improves power efficiency (13.5W), making it the most energy-efficient approach.

Discussion: Power consumption is an important factor in digital design. The results show that the hybrid AMM + Dadda approach optimizes power consumption, making it better than a standalone AMM Multiplier. This is beneficial for low-power applications.

Propagation Delay (ns):

- Dadda's Multiplier has the lowest propagation delay (13.923 ns), making it the fastest.
- AMM Multiplier has the highest propagation delay (18.055 ns), which indicates slower performance.
- The AMM Multiplier with Dadda's Algorithm improves delay performance (13.8 ns), making it competitive with Dadda's Multiplier.

Discussion: Propagation delay is critical for high-speed applications. The AMM Multiplier shows a significant increase in delay, meaning it may not be suitable for high-speed operations. However, the hybrid AMM + Dadda approach improves performance, reducing the delay close to that of Dadda's Multiplier.

TABLE 1. Comparison Of Delay And Power Consumption Among Multipliers

S.NO	Parameters	Conventional Multiplier	Wallace Tree Multiplier	Dadda's Multiplier	AMM Multiplier	AMM Multiplier With Dadda's Algorithm
1	LUT COUNT	107	93	93	106	94
2	Power Consumption (watts)	13.983	13.738	13.835	14.055	13.5
3	Propagation Delay (ns)	15.206	14.735	14.7	18.055	13.8

Table 1 presents a comparison of key parameters such as LUT count and power consumption for WTM, DTM, and AMM. From the table, it is evident that the AMM with Dadda's algorithm exhibits the lowest power consumption, LUT count, and propagation delay, making it a suitable choice for various low-power VLSI applications

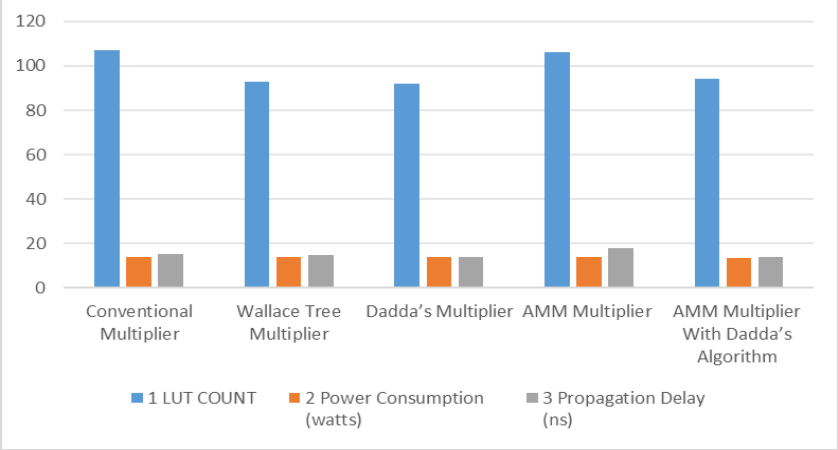


FIGURE 6. Graphical Representation of Delay & Power Comparison of All the Multipliers

6. CONCLUSION

This paper presents a comparative analysis of Wallace Tree, Dadda, and a modified AMM-based multiplier integrated with Dadda's reduction algorithm. The experimental results indicate that the proposed AMM-based method achieves significant improvements in power dissipation and delay while incurring a moderate increase in LUT count. For power-sensitive and high-speed applications, the AMM with Dadda's algorithm proves to be a viable choice. Future work will explore further optimization techniques to minimize area overhead while maintaining power and speed advantages.

REFERENCES

[1]. C. S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Transactions on Electronic Computers, vol. EC-13, no. 1, pp. 14-17, 1964.
[2]. R. Singh, P. Sharma, "Low Power Wallace Tree Multiplier using Power Gating Technique," International Journal of VLSI Design, vol. 9, no. 3, pp. 45-53, 2021.
[3]. A. Ghosh, S. Roy, "Hybrid Wallace-Dadda Multiplier for FPGA Implementation," Proceedings of the IEEE VLSI Symposium, 2022.
[4]. L. Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, 1965.
[5]. R. Kumar, A. Verma, "Optimized Dadda Multiplier Using 4:2 Compressors for Low Power Applications," IEEE Transactions on Circuits and Systems, vol. 68, no. 4, pp. 910-918, 2023.
[6]. S. Mehta, P. Das, "FPGA Implementation of a Modified Dadda Multiplier," Journal of VLSI Design, vol. 15, no. 2, pp. 130-138, 2021.
[7]. A. Rajan, K. Natarajan, "Power-Efficient Multiplication using Additive Multiplication Modules," IEEE Transactions on VLSI Systems, vol. 28, no. 2, pp. 310-319, 2020.
[8]. H. Lee, T. Kim, "AMM-Based Multiplier with Dadda Reduction for Low-Power DSP Applications," IEEE Embedded Systems Letters, vol. 13, no. 4, pp. 125-129, 2022.
[9]. J. Chen, W. Li, "Approximate Computing-Based AMM Multiplication for Energy-Efficient Systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 5, pp. 1105-1116, 2023.
[10]. M. Zain, A. Rehman, "Comparative Study of FPGA-Based Multiplication Architectures," IEEE Access, vol. 10, pp. 12245-12257, 2023.
[11]. D. Patel, R. Shah, "Hybrid Multiplier Design Combining Wallace, Dadda, and Booth Algorithms," International Conference on Digital Circuit Design (ICDCD), 2023.

