RESEARCH PAPER

Implementation and Functional Verification of a 4×4 Dadda Multiplier Using Verilog

V. Vijayabhasker[1*], Gudipally Bharani Kumar[2], Mahesh Mudavath[3], K. Suresh Babu[4]
[1]Associate Professor, ECE (VLSI), Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088
Email: vijay.valiki@siddhartha.co.in
[2]M.Tech, ECE (VLSI), Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088
Email: 24TQ1D5701@siddhartha.co.in
[3]Associate Professor, ECE (VLSI), Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088
Email: dr.mahesh@siddhartha.org.in
[4]Assistant Professor, ECE (VLSI), Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088
Email: k.sureshbabu@siddhartha.org.in
***Corresponding Author:** V. Vijayabhasker, Associate Professor, ECE (VLSI), Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088
Email: vijay.valiki@siddhartha.co.in

Received: 25th October 2026; **Revised:** 28th October 2026; **Accepted:** 3rd November 2026; **Available Online:** 7th November 2026

ABSTRACT
Multipliers are basic units of arithmetic utilized in digital signal processing, microprocessors, and VLSI systems, with speed and area being important design factors. Other multiplier architectures such as the Dadda multiplier have been known to have partial product reduction optimality and lower hardware complexity. The proposed design produces the partial products with the help of AND logic, minimizes them with the help of a Dadda-based reduction strategy, and then uses a carry-propagate adder to obtain the final product. Icarus Verilog is used to implement and simulate the multiplier, and waveform analysis is done using GTKWave to confirm the correctness of the functional aspects. Due to a large amount of simulation, edge cases, and worst-case input testing, the results of the proposed design indicate that it yields correct output across all the input combinations it has been tested on. The ease of implementation and the accuracy of correct functioning make the proposed 4×4 Dadda multiplier an appropriate basic building block used to create higher-order multipliers in VLSI applications.
Keywords: Binary Multiplication, Dadda Multiplier, Functional Verification, Partial Product Reduction, VLSI Design
How to cite this article: *V. Vijayabhasker1* Gudipally Bharani Kumar2 Mahesh Mudavath3 K. Suresh Babu4. Implementation and Functional Verification of a 4×4 Dadda Multiplier Using Verilog. Int J Drug Deliv Technol. 2026;16(90s):1444-1448. DOI: 10.25258/ijddt.16.90s.144.*
Source of support: *Nil.*
Conflict of interest: *None*

INTRODUCTION

Multiplication is an arithmetic operation that is very much essential in the current digital systems such as microprocessors, digital signal processor (DSP), application-specific integrated circuit (ASIC), and field-programmable gate array (FPGAs). Filtering, convolution, image processing, encryption, and communication algorithms are some of the operations that are heavily dependent on the fast and efficient multiplication units [1]. This in turn has a substantial effect on the performance of these systems overall as the rate, area and the power efficiency of the multiplier architecture employed [2]. With the VLSI technology ever scaling to deep sub-micron and nanometer regimes, there has been a high demand to have high-performance arithmetic units with less silicon area and low power consumption. Traditional multiplier designs, including array multipliers, provide simple and regular designs, which are easy to achieve. But, they experience long critical path delays because of the serial propagation of carries and need a big number of adders, hence incurs high area and power usage as the operand size is expanded [3].

Parallel multiplier designs like Wallace tree and Dadda multipliers were proposed in order to overcome these shortcomings. Wallace tree multipliers [4] are multipliers that eliminate partial products immediately they can, through multiple reduction stages, and so are fast. Nevertheless, such an aggressive reduction approach results in many adders, anomalous interconnection, and routing complexity, as well as more power dissipation, which makes Wallace multipliers less viable in area- and power-constrained design [5]. The optimized version of the Wallace tree multiplier was the Dadda multiplier. The major concept of the Dadda algorithm is to reduce the number of reduction stages and adders as much as possible by deferring the reduction of partial products until they are needed. The Dadda method limits the height of each column based on pre-determined limits, instead of reducing the partial product matrix in the most aggressive way possible, which leads to a more balanced reduction tree [6]. The solution saves a lot of the hardware complications and at the same time, the speed performance is at par with rivals.

It is because of these benefits that Dadda multipliers have received a lot of attention in recent years to find application in the high-speed and low-power VLSI systems [7]. Some studies have already proven that Dadda multipliers are superior to conventional array multipliers and offer similar or superior performance compared to Wallace multipliers which have lower area and power consumption [8]. The recent studies have concentrated on optimizing the performance of Dadda multipliers by doing architectural and circuit level optimization. This is one of the primary directions of using

optimized full adders and compressor-based reduction methods to achieve faster speeds and lower power consumption. Research documented in [9] and [10] indicates that substitution of the traditional full adders with hybrid or XOR-XNOR-based adder cells in the Dadda reduction tree is a significant way to minimise power dissipation and propagation delay without interfering with the functionality. Optimized adders assist in reducing switching activity and critical path delay and Dadda multipliers are therefore better used in low power VLSI applications.

The other significant current of research that is researched in [11] is the application of approximation adders and adjustable reduction techniques in the Dadda multiplier framework. With selective approximation of less important bits, massive energy savings and area can be attained with only slight effect on the accuracy of the output. These methods are especially effective in applications that are tolerant of error like image and signal processing, where minor computational errors are tolerated in favor of better performance. What is more, the material provided in [12] explores the concept of incorporating compressor-based architecture and sophisticated reduction methods in order to optimize Dadda multipliers further. All these studies point to the fact that Dadda multipliers provide a general design framework which can be customized to various performance, power, and area needs in current VLSI systems by the use of optimized adders, compressors and approximation methods.

It has been demonstrated through comparative studies that Dadda-based architectures are less delayed and more efficient than other tree-based multipliers, especially with optimized adders and compressors used [13], [14]. Besides, there have been investigations on accurate and approximate Dadda multipliers to be used in error tolerant applications which once again demonstrates the flexibility of Dadda multipliers in the contemporary digital design [15]. Based on these reasons, the implementation and functional verification of a 4x4 Dadda multiplier occurred in this paper using Verilog HDL. The work proposed is aimed at creating partial products with the help of AND logic, contracting them with the help of a reduction method based on Dadda, and the creation of the final result with the help of a carry-propagate adder. The complexity and specificity of the applied 4x4 Dadda multiplier enable it to be used as a simple arithmetic module in larger VLSI systems.

I. LITERATURE SURVEY

Multiplication is an arithmetic operation that is extremely important in all contemporary digital devices, such as microprocessors, digital signal processors (DSPs), application-specific integrated circuits (ASICs), and field-programmable gate arrays (FPGAs). The filtering, convolution, image processing, encryption and communication algorithms are some of the operations where fast and efficient multiplication units are very important [1]. This in turn affects the general efficiency of these systems in terms of speed, area, and power efficiency of the multiplier architecture employed [2]. With VLSI technology progressing to deep sub-micron and nanometer technology, the need has risen significantly to have high performance arithmetic units that minimally consume the silicon area and consume less power. The traditional multiplier designs including array multipliers provide regular designs that are easy to carry out. Nevertheless, they have long critical path delays since carries are propagated in a serial manner and need many adders, which cause an increase in area and power consumption with increase in the operand size [3].

In order to overcome these drawbacks parallel multiplier structures like Wallace tree and Dadda multipliers were proposed. Wallace tree multipliers [4] multiply partial products by reducing them as quickly as it can, by employing several reduction stages, making them fast. Nevertheless, this radical reduction method results in high adders, irregular interconnection, routing complexity, and power consumption, thus Wallace multipliers are not as useful in area- and power-constrained designs [5]. As an optimized version of the Wallace tree multiplier, the Dadda multiplier was suggested. The major concept behind this Dadda algorithm is that the reduction stages and adders are kept to a minimum by delaying the reduction of partial products until the point at which it is required. The Dadda method does not minimize the partial product matrix as hard as possible but rather limits the height of each column by predetermined values, which produces a more balanced reduction tree [6]. This implementation greatly minimizes the complexity of hardware and does not lose competitive speed performance.

Because of such benefits, Dadda multipliers have attracted a lot of research attention in recent years as alternatives to use in high-speed and low-power VLSI systems [7]. A number of studies have indicated that Dadda multipliers are better than standard array multipliers and offer a similar or better performance to Wallace multipliers at lower area and power consumption [8]. Recent work has been done on improving the performance of Dadda multipliers by architectural and circuit-level optimizations of their performance. A possible major direction is the application of optimized full adders and compressor-based reduction methods to enhance speed and minimize power usage. It has been shown in studies presented in [9] and [10] that a replacement of the traditional full adders with either hybrid or XOR-XNOR-based adder cells in the Dadda reduction tree will lower the power dissipation and propagation delay by a significant margin and still have the right functionality. These optimized adders are used to reduce switching activity and critical path delay and thereby Dadda multipliers are more appropriate to low-power VLSI applications.

The other significant research trend covered in [11] is the application of approximate adders and configurable reduction strategies in the Dadda multiplier model. With carefully chosen approximations of minor bits, it is possible to obtain great energy savings and area reduction with a small effect on output precision. They are especially effective in applications that do not need accuracy in computing, like image and signal processing, where minimal computational inaccuracies can be tolerated in favor of increased speed. Moreover, the article in [12] examines the combination of compressor-based designs and additional reduction algorithms to optimize Dadda multipliers even more. All these studies point out that Dadda multipliers provide the ability to design a flexible framework that can be optimized with optimized adders, compressors, and approximation schemes to fit a variety of performance, power and area needs in the modern VLSI systems.

It has been demonstrated that Dadda-based architectures have lower delay and better efficiency than other tree-based multipliers, especially when it is optimized by the use of

adders and compressors [13], [14]. Furthermore, the concept of correct and approximate Dadda multipliers has been discussed as applicable in error-tolerant applications, which further emphasize on their applicability in contemporary digital designs [15]. This paper is inspired by these factors and thus creates the implementation and functional validation of a 4x4 Dadda multiplier in the Verilog HDL. The suggested work is aimed at the generation of partial products with the help of AND logic, reduction of these products with the help of a Dadda-based reduction strategy, and the generation of the final product with the help of a carry-propagate adder. The ease and precision of used 4x4 Dadda multiplier render it appropriate as a simple arithmetic module in bigger VLSI systems.

II. Proposed Work

The given work is dedicated to the development and practical testing of a 4x4 Dadda multiplier with the help of Verilog HDL. The Dadda multiplier is a multiplier architecture in a tree form that minimizes the products that need to be produced by reducing the adders and the reduction stages. The general design of the proposed multiplier will be based on three major steps including partial product creation, Dadda reduction and the final carry-propagate addition. Fig. 1.1 represents the block diagram of the proposed architecture.

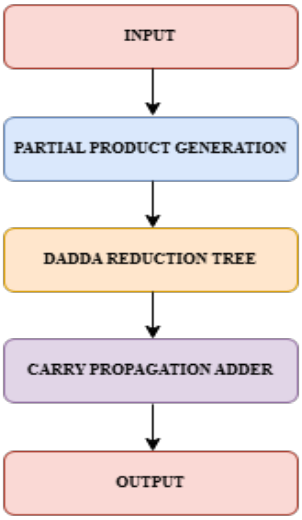


Fig. 1. Block Diagram

A. Partial Product Generation

In the first stage, partial products are generated by In the proposed 4×4 Dadda multiplier, partial products are generated by performing bitwise AND operations between the bits of the multiplicand $A = A_3A_2A_1A_0$ and the multiplier $B = B_3B_2B_1B_0$. This process results in sixteen partial products of the form A_iB_j, where $i,j \in \{0,1,2,3\}$. These partial products are arranged in a matrix according to their respective bit positions, forming the initial structure for multiplication. Mathematically, the partial products are given by:

$$pp_{ij} = A_i \cdot B_j$$

for $i,j = 0,1,2,3$

Fig. 2 illustrates the partial product matrix for the 4×4 multiplication. Each row corresponds to one bit of the multiplier, and each column represents a specific weight in the final product. The partial product matrix serves as the input to the Dadda reduction stages, where the number of bits in each column is reduced efficiently.

					A3	A2	A1 A0
					x B3	B2	B1 B0
				A3B0	A2B0	A1B0	A0B0
			A3B1	A2B1	A1B1	A0B1	
		A3B2	A2B2	A1B2	A0B2		
	A3B3	A2B3	A1B3	A0B3			
P7	P6	P5	P4	P3	P2	P1	P0

Fig. 2. 4×4 multiplications (partial product matrix).

B. Dadda Reduction Technique

The technique of Dadda reduction is used in order to minimize the height of the partial product matrix and the number of adders used. The Dadda algorithm is also used to reduce partial products only when the height of the column reaches a specified limit as opposed to Wallace tree multipliers, which reduce partial products at each stage. The strategy leads to the decrease in the number of adders, minimization in the complexity of wiring, and enhancement of the efficiency of the area.

1) First Dadda Reduction Stage: Under first Dadda stage, the partial product columns that are not allowed to be of a certain height are noted and minimized with the help of half adders and full adders. One bit columns (where there are three or more bits per column) are selectively eliminated to ensure that the height of the column meets the Dadda constraint of this step. Fig. 3 illustrates the first stage of reduction of the Dadda process, in which partial products are clumped into columns and reduced. In this step the height of the matrix is abridged and the multiplication remains accurate.

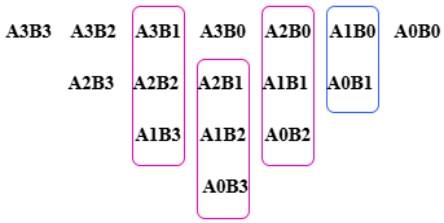


Fig. 3. First Dadda reduction stage.

2) Second Dadda Reduction Stage: When the first reduction has been completed, some columns might still be too high to allow the next stage. The next Dadda reduction step involves additional reduction of the column with the help of full adders and half adders in order to make the column height as small as possible to reach the next possible level as specified by the Dadda algorithm. The second stage of Dadda reduction is shown in Fig. 4 and the additional bits are cut off. This step then further reduces the partial product matrix and is ready to be reduced.

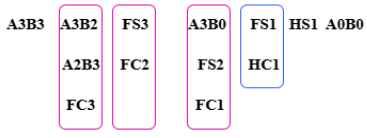


Fig. 4. Second Dadda reduction stage.

3) Final Dadda Reduction Stage: In the last Dadda step the partial product matrix has been reduced to two rows; a sum row and a carry row. This is done by applying the remaining adders to columns that have over two bits in them. There is no column on the end of this stage higher than two. Fig. 5 is the last reduction stage of Dadda. These two resultant rows

are then fed to the last carry-propagate adder to give the final multiplication accomplishment.

A3B3 FS5 HS3 FS4 HS2 HS1 A0B0

FC5 HC3 FC4 HC2

Fig. 5. Final Dadda reduction stage.

C. *Final Carry-Propagate Adder*

In the last phase, they are added with the help of a carry-propagate adder (CPA), the remaining two rows in the Dadda reduction stage. This adder gives the end result of the multiplication which is 8 bit. This step is necessitated by the use of a carry-propagate adder, which provides proper carry propagation and also provides the correct output to all input combinations including worst-case inputs. Despite the fact that the Dadda algorithm is optimal in reducing stage, the last step of carry-propagate addition is required to get the right product. This phase is realized behaviorally in Verilog HDL and this provides simplicity and correctness with being compatible with synthesis and simulation tools.

III. Results and Discussion

A. *Simulation Results*

The 4x4 Dadda multiplier was proposed and was implemented in Verilog HDL and tested, simulated using Icarus Verilog. The behavior of the multiplier with various combinations of inputs was observed by working on the waveforms of the circuit with the help of GTKWave. The waveform in Fig. 6 depicts the simulation waveform of the proposed design, which has indicated the relation between the input operands A[3:0], B[3:0] and the product output P[7:0]. To test the accuracy of the multiplier, the inputs were changed after varying periods of time. At first, both inputs were initialized to zero and the output did give the right response of zero. After this, various sets of input values were used and the respective output was recorded. An example is, in case the inputs were A=3and B=2, the output P=6. On the same note, right results were achieved with A=5, B=3 and P=15, and with A=11, B=6 and P=66. It was also confirmed that the multiplier is valid under the worst-case combination of inputs A=15and B=15. The output obtained was P=225 which was the correct output as indicated by the waveform indicating the adequate carry propagation as well as the correct reduction of partial products. The fact that unknown or undefined values do not occur in the course of the simulation is a sign of stable and reliable work of the offered architecture.

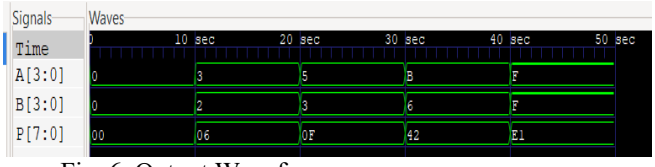


Fig. 6. Output Waveform

B. *Output Verification and Discussion*

The simulation is a sign of stable and reliable work of the offered architecture. The chosen test cases used in the course of the simulation and their results are summarized in Table 1. The results obtained precisely coincide with values of the multiplication as expected thus confirming the functional accuracy of the suggested Dadda multiplier.

Table 1. Simulation Output

Besides the chosen test cases, the suggested design was fully checked in all 256 possible combinations of inputs of the 4-bit operands. The simulation outcomes proved that all the input pairs are operating correctly, which proves the robustness and reliability of the proposed multiplier.

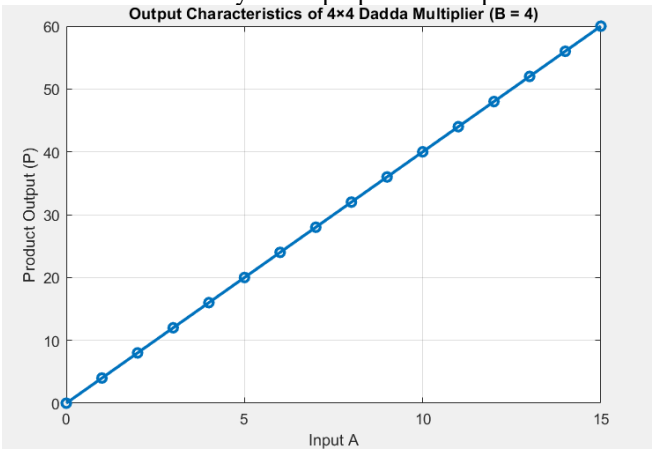


Fig. 7. Output characteristics of the proposed 4×4 Dadda multiplier for sequential variation of input A with fixed input B = 4.

The effect that the 4x4 Dadda multiplier proposed on the output output was further examined by changing one of the input operands at a time with the other held constant. In this analysis, the multiplicand input was changed between 0 and 15, and the input of the multiplier was set at 4. The respective product output was noted and graphically represented as in Fig. 7. Based on the graph, it can be seen that the output of the products is linearly proportional to the input operand. This straight-line property attests to the fact that the proposed multiplier yields proportional and correct values of output with respect to every input transition. This operation confirms the accuracy of multiplication operation in the whole range of input values and indicates a stable operation of Dadda multiplier architecture. The fact that there were no deviations in the linearity implies that the partial product generation, reduction and final addition stages are functioning properly. This finding also gives more testament to the validity and accuracy of the suggested 4x4 Dadda multiplier to sequential variations of inputs.

C. *Performance Comparison*

Table 2 shows performance comparison of various 4x4 multiplier architecture in power consumption, propagation delay as well as power-delay product (PDP). The array multiplier has the worst power consumption and delay because it has a sequence carry propagation. The Wallace multiplier is faster with the disadvantage of increased hardware complexity due to parallel reduction of the partial products. Vedic multiplier has moderate gains in power and delay because of algorithmic effectiveness.

Table 2. Performance Comparison

A[3:0]	B[3:0]	P[7:0] (Hex)	Decimal
3	2	06	6

4x4 Multiplier	Power (µw)	Propagation Delay (µs)	Power delay Product(pJ)

The given Dadda multiplier has the lowest propagation delay and the lowest product of power-delay of any of the compared architectures. This is a result of the streamlined partial product reduction strategy of the Dadda algorithm that reduces the adders and reduction steps. The low PDP means that it is more energy efficient and the proposed Dadda multiplier can be used in high-speed and low-power VLSI. The results of the simulation are clear on why the Dadda reduction technique may be used in reducing the height of the partial product and at the same time maintaining accurate

multiplication. The suggested architecture demonstrates a higher level of computational efficiency compared to the traditional array multiplier because it has fewer sequential addition operations. The appropriate management of worst-case inputs also demonstrates the appropriateness of the proposed 4x4 Dadda multiplier as a fundamental building block in higher-bit multiplier in VLSI systems.

IV. CONCLUSION

This paper has discussed the design, implementation as well as the functional verification of 4x4 Dadda multiplier. The proposed architecture uses effective partial product generation and subsequent Dadda-based reduction algorithm and a subsequent carry-propagate adder to derive the multiplication output. The Dadda algorithm can successfully minimize hardware hardware complexity and still be very computationally efficient by controlling the number of reduction stages and adders. The proposed multiplier was functionalally correct and was tested extensively on basic, mid-range and worst-case input combinations. The findings validate the assertion that the suggested design yields correct outputs on all the cases experimented. Comparative performance analysis with traditional multiplier architectures including Array, Wallace and Vedic multipliers depicts that the Dadda multiplier has lower propagation delay and low power-delay product which implies better energy efficiency. Because of its accuracy, lower complexity, and enhanced performance attributes, the suggested 4x4 Dadda multiplier would be very suitable in application as a basic building block in larger multiplier structures and multiplier arithmetic units in digital signal processing and embedded systems implementations.

REFERENCES

[1] S. Chanda, K. Guha, S. Patra, and A. Karmakar, "A 32-bit energy efficient exact Dadda multiplier," *Proc. IEEE 5th Int. Conf. Computing, Communication and Networking Technologies*, pp. 1–6, 2019.

[2] H. Afzali-Kusha and M. Vaeztourshizi, "Design exploration of energy-efficient accuracy-configurable Dadda multipliers with improved lifetime based on voltage overscaling," *IEEE Trans. VLSI Systems*, vol. 28, no. 7, pp. 1671–1684, 2020.

[3] A. Kumar, A. Kumar, and D. Nand, "Design and study of Dadda multiplier using 4:2 compressors and parallel prefix adders for VLSI circuit designs," *Proc. IEEE Int. Conf. VLSI Design*, pp. 1–5, 2021.

[4] V. Manu and A. M. V. Prakash, "Design and implementation of sixteen-bit low power and area efficient Dadda multiplier," *Proc. IEEE Int. Conf. Recent Trends in Electronics, Information and Communication Technology*, pp. 1–6, 2019.

[5] K. C. Pathak, A. D. Darji, and J. N. Sarvaiya, "Low power Dadda multiplier using approximate almost full adder and majority logic based adder compressors," *Proc. IEEE Region 10 Conf. (TENCON)*, pp. 1–6, 2022.

[6] S. Nagaleela, G. Shanthi, and B. Manisha, "Design of DADDA multiplier using high performance and low power full adder," *Proc. IEEE Int. Conf. Emerging Trends in Engineering*, pp. 1–6, 2023.

[7] B. R. Kumar, P. Munaswamy, and B. C. Naik, "Implementation of low power and high speed Dadda multiplier using XOR-XNOR cell based hybrid logic full adder," *Proc. IEEE Int. Conf. Smart Technologies*, pp. 1–5, 2023.

[8] M. H. Riaz, S. A. Ahmed, and Q. Javaid, "Low power 4×4 bit multiplier design using Dadda algorithm and optimized full adder," *Proc. IEEE Int. Conf. Advances in Electrical Engineering*, pp. 1–5, 2018.

[9] D. Jangalwa and M. Nagabushanam, "Design and analysis of 8-bit multiplier for low power VLSI applications," *Proc. IEEE Mysore Subsection Int. Conf.*, pp. 1–6, 2022.

[10] M. Munawar, Z. Shabbir, and M. Akram, "Area, delay, and energy-efficient full Dadda multiplier," *Journal of Circuits, Systems and Computers*, vol. 32, no. 12, pp. 1–20, 2023.

[11] B. S. Kandula, R. Devi, and M. K. Devi, "Performance analysis of VLSI architecture of Dadda multiplier using divide and conquer strategy with approximate adders," *Proc. IEEE Int. Conf. Emerging Trends in Engineering and Technology*, pp. 1–6, 2024.

[12] **U**. Rachanna, S. Shirol, S. Sonal, and M. Srushti, "Design and implementation of Dadda multiplier in 45-nm technology," *Springer Int. Conf. Communication and Signal Processing*, pp. 121–130, 2024.

[13] J. Nessa, S. Dash, and M. C. Tripathy, "High speed low power 16×16 Dadda multiplier by utilizing reversible gates," *Proc. IEEE Int. Conf. VLSI and Embedded Systems*, pp. 1–6, 2025.

[14] S. Chanda, K. Guha, S. Patra, and L. M. Singh, "An energy efficient 32-bit approximate Dadda multiplier," *Proc. IEEE Calcutta Conf.*, pp. 1–6, 2020.

[15] A. A. Rather, B. Khurshid, S. A. Banday, and A. A. Algarni, "Design of high-performance, accurate and approximate Dadda-tree multipliers for image processing applications," *Scientific Reports*, vol. 15, pp. 1–15, 2025.