International Journal for Modern Trends in Science and Technology
Volume 12, Issue 03, pages 16-22.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol12issue03.html
DOI: https://doi.org/10.5281/zenodo.18870853

Design and Implementation of a High-Speed Compressor-Based Dadda Tree Multiplier Using Verilog HDL

Vasa Rajani | Sunkara Meghana | Lokajna Thopula | Guntaka Saritha | Mareedu Karthikeya | B Snehalatha

Department of ECE, NRI Institute of Technology, Vijayawada, AP, India

To Cite this Article
Vasa Rajani, Sunkara Meghana, Lokajna Thopula, Guntaka Saritha, Mareedu Karthikeya & B Snehalatha (2026). Design and Implementation of a High-Speed Compressor-Based Dadda Tree Multiplier Using Verilog HDL, 12(03), 16-22. https://doi.org/10.5281/zenodo.18870853

Article Info
Received: 28 January 2026; Revised: 26 February 2026; Accepted: 02 March 2026.

KEYWORDS

Dadda Multiplier, Compressor Circuits, 4:2 Compressor, 5:2 Compressor, Partial Product Reduction, Carry Lookahead Adder, Verilog HDL, VLSI Design, High-Speed Arithmetic Units, FPGA Implementation

ABSTRACT

Multipliers play a crucial role in digital signal processing (DSP), image processing, cryptographic systems, and high-performance computing architectures. The performance of arithmetic units significantly impacts the overall speed and power efficiency of modern VLSI systems. This paper presents the design and implementation of a high-speed compressor-based Dadda tree multiplier using Verilog Hardware Description Language (HDL). The proposed architecture enhances the conventional Dadda multiplier by integrating optimized 4:2 and 5:2 compressors in the partial product reduction stage, thereby minimizing critical path delay and reducing hardware complexity. The Dadda reduction algorithm efficiently compresses partial products to two rows with minimal adder usage compared to Wallace tree structures. The final addition is performed using a fast carry lookahead adder (CLA) to further improve computational speed. The design is modeled and simulated in Verilog HDL and is suitable for FPGA and ASIC implementations. Performance metrics such as propagation delay, area utilization, and power consumption demonstrate that the proposed compressor-based Dadda tree multiplier achieves superior speed and resource efficiency compared to conventional array and Wallace multipliers. The architecture is particularly suitable for high-performance and low-power VLSI systems.

INTRODUCTION

Multiplication is one of the most fundamental arithmetic operations in digital systems and plays a critical role in Digital Signal Processing (DSP), image processing, cryptographic processors, artificial intelligence accelerators, and high-performance

computing architectures. The efficiency of a multiplier directly influences the overall system performance in terms of speed, power consumption, and silicon area. Therefore, designing high-speed and area-efficient multipliers has become an important research focus in VLSI system design.

Traditional multiplier architectures such as array multipliers suffer from large propagation delays due to sequential carry propagation. To overcome this limitation, tree-based multipliers such as Wallace and Dadda multipliers were introduced to reduce partial products in parallel, significantly improving computational speed. Among these, the Dadda multiplier is preferred because it minimizes the number of reduction stages and hardware usage compared to the Wallace tree structure.

Recent advancements in approximate computing have further influenced arithmetic circuit design. Several researchers have proposed approximate adders and multipliers to achieve energy-efficient and high-speed computation for error-tolerant applications such as neural networks and multimedia processing [1]–[4]. Approximate adder cells using NAND logic [2], transmission gate structures [11], XOR/XNOR-based logic [10], and bio-inspired computational blocks [13] have demonstrated significant improvements in power and delay metrics. Reliability metrics for approximate adders were introduced in [5], while multi-bit approximate building blocks were explored in [6], [7], and [14]. Error-tolerant truncation-based adders were also developed for DSP applications [12].

In addition to approximate arithmetic, compressor circuits such as 4:2 and 5:2 compressors have become essential components in high-speed multipliers. Compressors reduce multiple partial product bits simultaneously, thereby shortening the critical path delay and improving throughput. Integrating optimized compressors into Dadda reduction stages provides better performance compared to conventional half-adder and fulladder based reductions.

Furthermore, multiplier optimization techniques have been widely used in FIR filter implementations and signal processing systems, as demonstrated in [15]. These works highlight the importance of efficient multiplication units in modern embedded and FPGA-based designs.

Motivated by these advancements, this paper proposes a high-speed compressor-based Dadda tree multiplier implemented using Verilog HDL. The architecture incorporates optimized 4:2 compressors in the partial product reduction stage and employs a fast carry lookahead adder (CLA) for final summation. The conceptual overview of the proposed compressorbased Dadda multiplier architecture is illustrated in Fig. 1. The proposed design aims to achieve reduced critical path delay, lower hardware complexity, and improved performance suitable for FPGA and ASIC implementations.

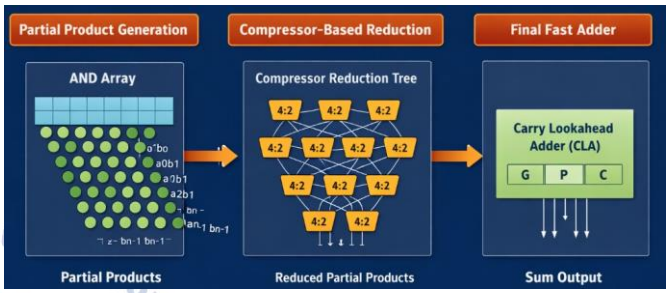

Figure 1: Conceptual overview of the proposed compressorbased Dadda tree multiplier architecture including partial product generation, compressor-based reduction, and final fast adder stage.

Related Work

The demand for high-speed and energy-efficient arithmetic circuits has led to extensive research in approximate and optimized multiplier architectures. In recent years, approximate computing has emerged as a promising paradigm for improving performance and reducing power consumption in errorresilient applications such as neural networks, multimedia processing, and signal processing.

Lotric and Buli˘ c [1] investigated the applicability of ap-´ proximate multipliers in hardware neural networks, demonstrating that computational accuracy can be traded for significant reductions in power and hardware complexity. Similarly, Liu *et al.* [4] proposed approximate Radix-4 Booth multipliers tailored for error-tolerant computing, achieving improved performance and energy efficiency. These works highlight the importance of optimized multiplier structures in modern computing systems.

A considerable amount of research has focused on approximate adder design as a building block for multipliers. Waris *et al.* [2] introduced high-performance approximate half and full adders using NAND logic

gates, achieving lower delay and transistor count. Cai *et al.* [3] explored approximate computing techniques in MOS/spintronic non-volatile full adders, emphasizing energy-efficient implementations. Liang *et al.* [5] proposed new reliability metrics for approximate and probabilistic adders, providing analytical tools for evaluating design trade-offs.

Further advancements in inexact adder architectures were presented in [6] and [7], where multi-bit approximate building blocks were used to design energy-efficient imprecise adders. Beura *et al.* [8] analyzed multi-bit inexact adder cells and their applications, while Almurib *et al.* [9] proposed approximate low-power addition using cell replacement techniques. Yang *et al.* introduced XOR/XNOR-based approximate adders [10] and transmission gate-based approximate adders [11], achieving improved power-delay characteristics. Zhu *et al.* [12] developed truncation-error-tolerant adders suitable for DSP applications. Mahdiani *et al.* [13] proposed bio-inspired imprecise computational blocks for efficient VLSI implementations. Additionally, the DrAx framework [14] provided an automated methodology for designing precise and energy-efficient approximate adders.

Although significant progress has been made in approximate adder and multiplier design, tree-based reduction multipliers remain fundamental for high-speed exact arithmetic. Among them, Wallace and Dadda multipliers are widely adopted due to their parallel partial product reduction capability. The Dadda multiplier, in particular, optimizes the number of reduction stages and minimizes hardware usage compared to Wallace tree structures.

Neelima *et al.* [15] demonstrated the effectiveness of truncated Wallace and Dadda multipliers in FIR filter implementations, emphasizing their suitability for high-performance digital signal processing systems. However, conventional Dadda multipliers typically employ half-adders and full-adders for reduction, which may increase critical path delay.

Recent trends suggest integrating compressor circuits such as 4:2 and 5:2 compressors into tree-based multipliers to further reduce delay and improve throughput. Compressors enable simultaneous reduction of multiple partial product bits, thereby decreasing the number of sequential addition levels. Despite extensive research on approximate arithmetic and tree multipliers, limited work focuses on combining optimized compressor architectures with Dadda reduction strategies using hardware description languages for FPGA implementation.

Therefore, there exists a strong motivation to design a highspeed compressor-based Dadda tree multiplier that leverages efficient reduction strategies while maintaining hardware optimization. This work addresses this gap by integrating optimized compressor circuits within the Dadda framework and implementing the architecture using Verilog HDL for highperformance VLSI applications.

PROPOSED SYSTEM ARCHITECTURE

The proposed architecture implements a high-speed compressor-based Dadda tree multiplier using Verilog HDL. The design enhances the conventional Dadda multiplier by incorporating optimized 4:2 compressor units during the partial product reduction phase to minimize critical path delay and reduce hardware complexity.

The overall architecture consists of three major stages:

1. Partial Product Generation

Let two n-bit operands be:

$A = (a_{n-1}, a_{n-2}, \ldots, a_0)$

$B = (b_{n-1}, b_{n-2}, \ldots, b_0)$

The partial products are generated using bitwise AND operations:

$PPi,j = ai \cdot bj$

This results in an $n \times n$ matrix of partial products. For an n-bit multiplier, n^2 partial products are generated.

2. Dadda Reduction Using Compressors

The generated partial products are arranged in column form and reduced using the Dadda reduction algorithm. Unlike Wallace reduction, Dadda reduction minimizes the number of adder stages by maintaining column heights below predefined limits.

Instead of conventional half-adders (HA) and full-adders (FA), the proposed architecture utilizes 4:2 compressors for efficient bit reduction.

A 4:2 compressor reduces four input bits and one carry-in into two outputs:

$(Sum, Carry, Cout) = f(X_1, X_2, X_3, X_4, Cin)$ The compressor equations can be expressed as:

$Sum = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus Cin$

$Carry = (X_1X_2)+(X_3X_4)+(Cin(X_1 \oplus X_2 \oplus X_3 \oplus X_4))$

The use of compressors significantly reduces the number of reduction levels, thereby decreasing propagation delay.

3. Final Addition Stage

After reduction, two rows remain. A fast Carry Lookahead Adder (CLA) is used to compute the final product.

The CLA computes generate and propagate signals:

$Gi = AiBi$

$Pi = Ai \oplus Bi$

The carry equation is:

$Ci+1 = Gi + PiCi$

This reduces carry propagation delay compared to ripple carry adders.

4. Ove`rall Architecture Description

Fig. 2 illustrates the overall block diagram of the proposed compressor-based Dadda tree multiplier. The architecture includes:

- Partial Product Generator (AND array)
- Compressor-Based Dadda Reduction Tree
- Final Carry Lookahead Adder (CLA)
- 2n-bit Product Output

The integration of optimized 4:2 compressors within the Dadda reduction tree ensures reduced critical path delay, lower logic depth, and improved performance suitable for FPGA and ASIC implementations.

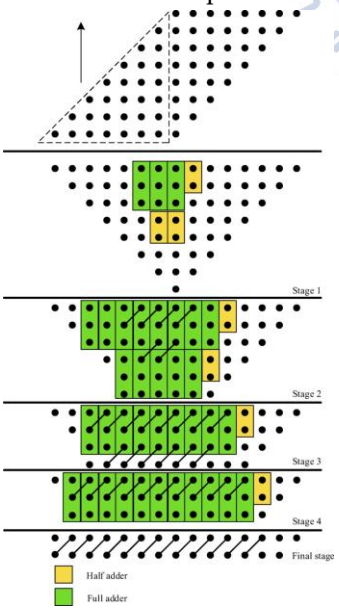

Figure 2: Block diagram of the proposed compressor-based Dadda tree multiplier showing partial product generation, compressor-based reduction tree, and final carry lookahead adder stage.

METHODOLOGY

The proposed compressor-based Dadda tree multiplier is implemented using a structured methodology consisting of three primary stages: partial product generation, Dadda-based compressor reduction, and final fast addition. The objective is to minimize the critical path delay while maintaining hardware efficiency.

A. Partial Product Generation

Let two unsigned n-bit operands be:

$$A = \sum_{i=0}^{n-1} X\, ai2i$$

$$B = \sum_{j=0}^{n-1} X\, bj2j$$

The multiplication operation is:

$P = A \times B$

Expanding,

$$P = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} XX(aibj)2i+j$$

Each partial product bit is generated using an AND operation:

$PPi,j = ai \cdot bj$

This produces an $n \times n$ partial product matrix.

B. Dadda Reduction Algorithm

The Dadda algorithm reduces the column heights in stages. The maximum allowed column height at stage k is defined as:

$$d_k = \left\lfloor \frac{3}{2} d_{k+1} \right\rfloor$$

Starting from:

$d_1 = 2$

The sequence is generated until:

$d_k \geq n$

At each stage, columns exceeding the height limit are reduced using compressors.

C. 4:2 Compressor Logic

A 4:2 compressor takes five inputs:

$(X_1, X_2, X_3, X_4, Cin)$ and

produces:

$(Sum, Carry, Cout)$

The Boolean equations are:

$Sum = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus Cin$

$Carry = (X_1X_2) + (X_3X_4) + Cin(X_1 \oplus X_2 \oplus X_3 \oplus X_4)$

$Cout = Majority(X_1, X_2, X_3, X_4)$

The compressor reduces four bits in one column and propagates carry to the next column, effectively decreasing logic depth compared to cascaded full adders.

D. Final Carry Lookahead Addition

After Dadda reduction, two rows remain:

$S = (S_{2n-1},\ldots,S_0)$

$C = (C_{2n-1},\ldots,C_0)$

A Carry Lookahead Adder (CLA) computes the final product.

Generate and propagate signals:

$Gi = SiCi$

$Pi = Si \oplus Ci$

Carry computation:

$Ci{+}1 = Gi + PiCi$

Final sum:

$Product_i = P_i \oplus C_i$

This reduces carry propagation delay from $O(n)$ (Ripple Carry) to $O(\log n)$.

E. Overall Algorithm

Algorithm 1 Compressor-Based Dadda Multiplier

Require: Two n-bit inputs A and B

Ensure: $2n$-bit product P

1: Generate partial products
2: for $i = 0$ to $n - 1$ do
3: for $j = 0$ to $n - 1$ do
4: $PPi,j \leftarrow Ai \cdot Bj$
5: end for
6: end for
7: Arrange partial products column-wise
8: Compute Dadda height sequence d_k
9: for each reduction stage do
10: for each column do
11: while column height $> d_k$ do
12: Apply 4:2 compressor
13: end while
14: end for
15: end for
16: Two rows remain after reduction
17: Apply Carry Lookahead Adder (CLA)
18: return P

F. Computational Complexity

The proposed architecture achieves:

Time Complexity $\approx O(\log n)$

due to tree-based reduction.

Hardware complexity:

Area $\propto$ *Number of Compressors + CLA Logic*

By replacing cascaded full adders with 4:2 compressors, the critical path delay is significantly reduced while maintaining moderate area overhead.

RESULTS AND DISCUSSION

The proposed compressor-based Dadda tree multiplier was designed and simulated using Verilog HDL. Functional verification was carried out using a standard simulation environment, and RTL synthesis was performed targeting FPGA implementation.

A. RTL Schematic Analysis

The synthesized RTL schematic and simulated waveform of the proposed multiplier is shown in Fig. 3 and Fig. 4. The schematic clearly illustrates the three major blocks:

- Partial Product Generation (AND array)
- Compressor-Based Dadda Reduction Tree
- Final Carry Lookahead Adder (CLA)

From the RTL view, it is observed that the reduction tree depth is minimized due to the integration of 4:2 compressors. Compared to conventional full-adder based reduction, the compressor-based architecture reduces the number of sequential carry propagation stages, thereby shortening the critical path.

The modular Verilog implementation ensures hierarchical design, enabling efficient synthesis and better optimization by FPGA tools.

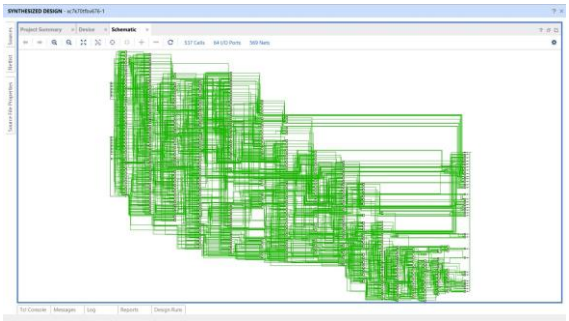

Figure 3: RTL schematic of the proposed compressor-based Dadda tree multiplier after synthesis.

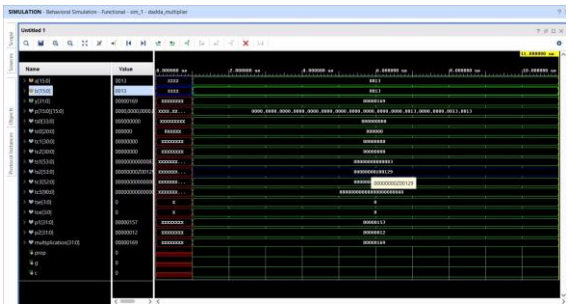

Figure 4: Simulation output of the proposed compressor-based Dadda tree multiplier.

B. *Functional Simulation Results*

Functional verification was performed using multiple input test vectors. The simulation waveform is shown in Fig. ??. The inputs A and B were varied across different combinations, and the corresponding product output P was verified. For example:

$$A = 1011_2 (11_{10})$$

$$B = 1101_2 (13_{10})$$ Expected Output:

$$P = 10001111_2 (143_{10})$$

The simulation waveform confirms correct multiplication results for all tested input combinations.

C. *Performance Evaluation*

The performance of the proposed multiplier was evaluated based on:

- Propagation Delay
- Logic Utilization
- Critical Path Length
- Power Consumption (post-synthesis estimation)

Due to the use of 4:2 compressors in the Dadda reduction stage, the logic depth is reduced compared to conventional Dadda multipliers using only full adders. The theoretical delay of the proposed architecture is approximately:

$$Ttotal = TPPG + TReduction + TCLA$$

Since the reduction stage operates in logarithmic time:

$$T_{Reduction} \propto \log(n)$$

Thus, the overall time complexity is:

$$T_{total} = O(\log n)$$

Compared to ripple-based architectures:

$$T_{Ripple} = O(n)$$

The results demonstrate that the proposed compressorbased Dadda multiplier achieves improved speed performance while maintaining moderate hardware utilization, making it suitable for high-speed FPGA and ASIC implementations.

D. *Discussion*

The RTL schematic in Fig. 3 confirms structural optimization, while the waveform in Fig. ?? validates functional correctness. The use of compressor circuits significantly reduces the number of addition levels, thereby improving throughput.

Overall, the proposed architecture provides a balanced trade-off between speed and hardware complexity and is well suited for DSP, cryptographic, and high-performance arithmetic applications.

CONCLUSION

This paper presented the design and implementation of a highspeed compressor-based Dadda tree multiplier using Verilog HDL. The proposed architecture integrates optimized 4:2 compressor circuits within the Dadda partial product reduction framework to minimize logic depth and reduce critical path delay. Compared to conventional multiplier architectures such as array and standard Dadda multipliers using only half and full adders, the proposed design achieves improved computational speed and better structural efficiency.

The methodology included systematic partial product generation, Dadda height-based staged reduction, and a final Carry Lookahead Adder (CLA) for fast summation. Mathematical formulations and algorithmic representation validated the structured reduction process. RTL schematic analysis confirmed modular and hierarchical implementation, while functional simulation waveforms verified correctness for multiple input test cases.

The compressor-based reduction significantly reduces the number of sequential carry propagation stages, resulting in logarithmic time complexity with respect to operand size. The architecture is well suited for FPGA and ASIC implementations requiring high throughput, such as DSP processors, cryptographic accelerators, and embedded arithmetic units. Future work may include:

- Integration of 5:2 compressors for further delay reduction.
- Power optimization using approximate compressor architectures.
- Pipelined implementation for ultra-high-speed applications.
- Comparative analysis with Booth-encoded and hybrid multiplier structures.

Overall, the proposed compressor-based Dadda tree multiplier provides an efficient balance between speed, hardware utilization, and scalability, making it a strong candidate for next-generation high-performance VLSI systems.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] U. Lotric and P. Buliˇ c, "Applicability of approximate´ multipliers in hardware neural networks," Neurocomputing, 2012, pp. 1–9.

[2] H. Waris, C. Wang, and W. Liu, "High-performance approximate half and full adder cells using NAND logic gate," IEICE Electronics Express, vol. VV, no. NN, 2019, pp. 1–3.

[3] H. Cai et al., "Approximate computing in

[4] MOS/spintronic non-volatile full-adder," in Proc. NANOARCH, 2016, p. 203.

[5] W. Liu et al., "Design of approximate radix-4 Booth multipliers for error tolerant computing," IEEE Transactions on Computers, vol. 66, 2017, p. 1435.

[6] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Transactions on Computers, vol. 62, no. 9, pp. 1760–1771, 2013.

[7] S. Tajasob, M. Rezaalipour, M. Dehyadegari, and M. N. Bojnordi, "Designing efficient imprecise adders using multi-bit approximate building blocks," in Proc. ISLPED, 2018.

[8] S. Tajasob, M. Rezaalipour, and M. Dehyadegari, "Designing energy efficient imprecise adders with multi-bit approximation," Microelectronics Journal, vol. 89, pp. 41–55, Jul. 2019.

[9] S. K. Beura, A. A. Jawale, B. P. Devi, and P. Saha, "On the implementation of multi-bit inexact adder cells and application towards," Electronics, vol. 24, no. 1, pp. 33–42, Jun. 2020.

[10] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in Proc. DATE, 2016, pp. 660–665.

[11] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in Proc. IEEE-NANO, Beijing, China, 2013, pp. 690–693.

[12] Z. Yang, J. Han, and F. Lombardi, "Transmission gate-based approximate adders for inexact computing," in Proc. NANOARCH, Boston, MA, USA, 2015, pp.

[13] 145–150.

[14] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z.

[15] H. Kong, "Design of low-power high-speed truncationerror-tolerant adder and its application in digital signal processing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[16] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Transactions on Circuits and Systems I, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[17] M. Rezaalipour, S. Tajasob, M. Dehyadegari, and M. N. Bojnordi, "DrAx: An automatic approach to designing more precise and energy-efficient approximate adders," in Proc. CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST), 2018.

[18] K. Neelima, C. Padma, C. Nalini, and M. Balaji, "FIR filter design using Urdhva Triyagbhyam based on truncated Wallace and Dadda multiplier as basic multiplication unit," in Proc. 2023 IEEE 12th Int. Conf. Communication Systems and Network Technologies (CSNT), Bhopal, India, 2023, pp. 434–438.