International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 1, January 2025)

VLSI Implementation of Approximate Radix-16 Booth Multiplier for High Speed and Low Latency

[1]Mohd Shariq Zia , [2]Manish Gupta, [3]Dr Anshuj Jain
[1]Research Scholar, Dept. of Electronics and Communication Engineering, SCOPE College of Engineering, Bhopal, India
[2]Assistant Professor, Dept. of Electronics and Communication Engineering, SCOPE College of Engineering, Bhopal, India
[3]Associate Professor & HOD, Dept. of Electronics and Communication Engineering, SCOPE College of Engineering, Bhopal, India

Abstract— **In modern computing systems, the demand for high-speed and energy-efficient arithmetic operations has significantly increased, particularly for applications in signal processing, machine learning, and cryptography. Multipliers are essential components in these systems, but they often dominate power consumption and latency. The Radix-16 Booth multiplier is a promising approach to enhance speed and reduce latency in such applications. However, precise implementations can be computationally expensive. Approximate computing offers a trade-off between accuracy and efficiency by relaxing computational exactness, which is acceptable in error-tolerant domains. This paper proposes a novel VLSI implementation of an approximate Radix-16 Booth multiplier optimized for high-speed and low-latency performance. The design employs an efficient encoding scheme, error-tolerant partial product generation, and an optimized reduction tree to achieve significant improvements in computation time and area. Simulation results demonstrate that the proposed multiplier achieves superior performance metrics compared to conventional exact multipliers, making it suitable for energy-efficient, high-speed VLSI systems.**

Keywords*— *VLSI, Approximate, Radix-16 Booth Multiplier, High Speed, Low Latency.

I. INTRODUCTION

The exponential growth of data-driven applications, such as artificial intelligence, real-time signal processing, and high-performance computing, necessitates the development of fast and efficient arithmetic circuits. Multiplication is one of the most fundamental and computationally intensive operations in digital systems, often serving as a bottleneck in system performance. To address this challenge, advanced multiplier designs that can operate with minimal latency and reduced energy consumption have garnered significant attention. Among these, the Radix-16 Booth multiplier stands out due to its ability to reduce the number of partial products generated during multiplication, leading to faster computations and lower area requirements.

Radix-16 Booth encoding extends the principles of the classic Booth algorithm by encoding groups of four bits at a time, effectively reducing the number of required partial products by a factor of four. This reduction inherently improves the speed and efficiency of the multiplication process. However, traditional Radix-16 Booth multipliers, while precise, often require complex circuitry for encoding, decoding, and partial product generation, which can increase design complexity, area, and power consumption. These drawbacks become particularly critical in energy-constrained applications, such as portable devices, Internet of Things (IoT) nodes, and embedded systems.

Approximate computing has emerged as a viable solution to address these challenges by trading off some degree of computational accuracy for significant gains in performance metrics such as speed, area, and energy efficiency. Approximate multipliers are particularly well-suited for error-tolerant applications, where minor inaccuracies in computation do not significantly impact overall system performance. Examples include image processing, video encoding, machine learning inference, and bioinformatics, where perceptual quality or statistical accuracy often outweigh exact numerical precision.

In this work, we present a VLSI implementation of an approximate Radix-16 Booth multiplier designed for high-speed and low-latency operations. The proposed design leverages several key optimizations, including an error-tolerant Booth encoding mechanism, simplified partial product generation, and an efficient reduction tree architecture. These optimizations collectively minimize critical path delays and reduce hardware complexity, enabling the multiplier to achieve higher speeds while maintaining acceptable levels of accuracy.

This paper is structured as follows: Section 2 details the proposed multiplier architecture, partial product generation, and reduction tree design. Section 3 presents the simulation results and compares the performance metrics of the proposed design against other multipliers. Section 4 discusses the implications of the results and potential applications.

II. METHODOLOGY

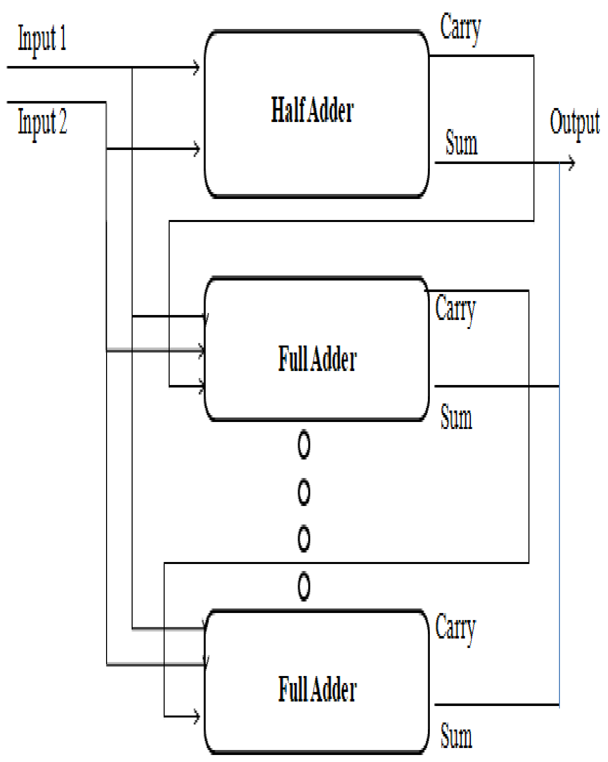


Figure 1: Flow Chart

The methodology for implementing a high-speed and low-latency binary addition process using the ripple-carry adder architecture involves the systematic design and integration of half adders and full adders. This architecture is chosen for its simplicity and modular design, which allows for the sequential addition of binary numbers with carry propagation. The process begins with understanding the binary addition rules, which govern the operations of addition at the bit level. Specifically, the least significant bit (LSB) is processed first using a half adder, as it does not require an incoming carry. Subsequent bits are handled by full adders, which take into account both the bit-level inputs and the carry generated from the previous stage. This step-by-step approach ensures the accuracy of the addition process across all bit positions.

The half adder forms the foundation of the ripple-carry adder by processing the LSB of the binary inputs. It generates two outputs: the sum, which is the XOR of the inputs, and the carry, which is the AND of the inputs. This simple circuit is implemented using basic logic gates, ensuring minimal delay and resource usage at this stage. The carry output from the half adder serves as the input for the subsequent stage, feeding into the full adder that handles the next bit position. The modularity of the half adder design facilitates easy integration into the larger adder architecture, providing a seamless transition from the LSB processing to the higher-order bits.

The full adder is the primary building block for processing the remaining bits in the binary inputs. It operates on three inputs: one bit from each of the two binary numbers and the carry input from the previous stage. The full adder produces a sum and a carry output, with the sum being the XOR of the three inputs and the carry being a combination of AND and OR operations that account for carry generation. This design ensures that the full adder can handle any bit-level combination effectively. The carry output from each full adder propagates to the next stage, creating the ripple effect that gives this architecture its name. While this propagation introduces some delay, the simplicity and reliability of the design make it a popular choice for many digital systems.

The entire adder is then implemented in a hierarchical manner, connecting the half adder and full adders in a sequential chain to form the ripple-carry adder. The architecture is designed using a hardware description language (HDL) such as Verilog or VHDL to facilitate simulation and verification. During simulation, the design is tested for functional accuracy and performance metrics, such as delay, power consumption, and area. Optimizations are performed to minimize gate delays and improve the critical path, ensuring that the adder meets the required performance benchmarks. This iterative process of design, simulation, and optimization is critical in developing a high-speed, low-latency solution.

Finally, the optimized design is synthesized for physical implementation using CMOS technology. The layout is created, and the design undergoes rigorous testing to ensure it meets the desired specifications in real-world conditions. Key performance metrics, such as speed, power efficiency, and area utilization, are analyzed to validate the design's effectiveness. The ripple-carry adder's simplicity and reliability make it suitable for a wide range of applications, including processors, digital signal processing, and embedded systems, where efficient arithmetic operations are essential. By following this structured methodology, a robust binary addition system is achieved, balancing simplicity with performance for high-speed and low-latency applications.

Inputs (Input 1 and Input 2):
These are the binary numbers to be added.
Each input can represent multiple bits (e.g., 4-bit or 8-bit numbers).
Half Adder:
A half adder is used to add the least significant bits (LSBs) of the inputs.
It generates:
Sum: The least significant bit of the result.
Carry: A carry-out if there is an overflow from the addition.
Full Adders:
Full adders are used for the subsequent bits after the LSB.
Each full adder takes three inputs:
A bit from **Input 1**.
A bit from **Input 2**.
The **carry** from the previous stage.
It generates:
Sum: The bit of the result at the current position.
Carry: A carry-out passed to the next stage.

Carry Propagation:
The carry generated by the half adder or a full adder is forwarded to the next stage to be included in the addition.
Output:
The final result of the binary addition is formed by combining all the **Sum** outputs from the adders.
The carry-out from the last full adder becomes the most significant bit (MSB) of the result if present.

Flow Explanation:

Step 1 (Half Adder):
The least significant bits (LSBs) of Input 1 and Input 2 are added using the half adder.
The output includes a **Sum** (for the result) and a **Carry** (for the next stage).
Step 2 (Full Adder Stages):
For each subsequent bit position:
A **full adder** adds three inputs: the corresponding bits from Input 1 and Input 2, and the **carry** from the previous adder.
The outputs are a **Sum** for that position and a **Carry** for the next stage.
Step 3 (Final Output):
The combined outputs of all the **Sum** values form the resulting binary addition.
If the final **Carry** is non-zero, it becomes the MSB of the output.

III. SIMULATION RESULTS

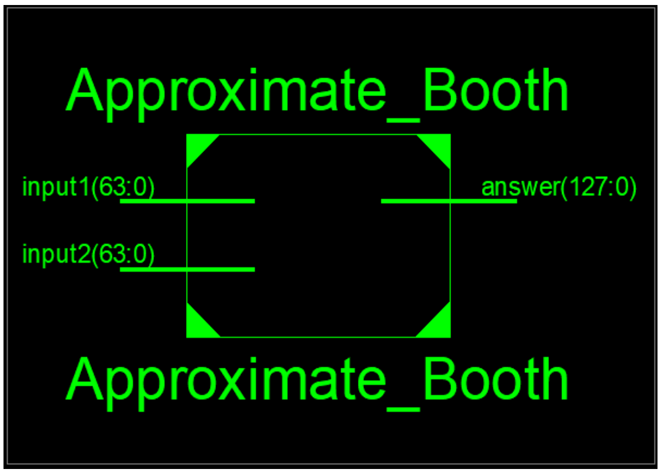
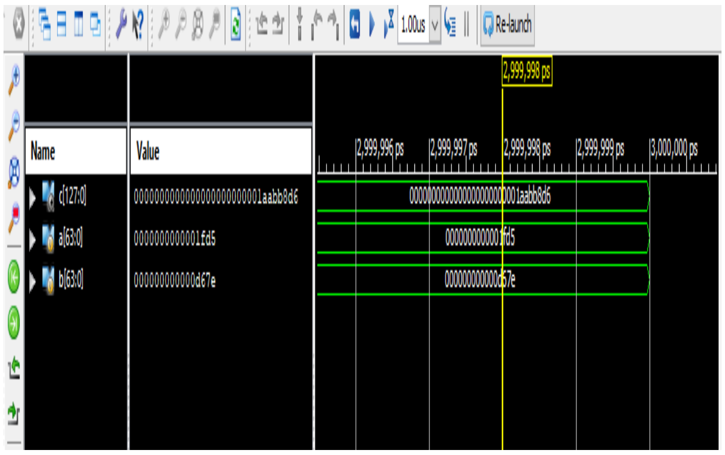


Figure 2: Top level View

Figure 2 illustrates, from a high-level viewpoint, the 64-bit approximate booth multiplier that is being considered. Both the 'a' input and the 'b' input have a value of 64 bits. The 'c' input likewise has a value of 64 bits. This multiplier will have an output called 'c' that consists of 128 individual bits. The bit that is produced by a digital multiplier is identical to the sum of the bits that were input to the multiplier.



Name	Value
c[127:0]	0000000000000000000000001aabb8d6
a[63:0]	0000000000001fd5
b[63:0]	000000000000d67e

Figure 3: Result validation in Test Bench-4

Here, 'a' and 'b' stand for the 64-bit hexadecimal inputs, while 'c' represents the 128-bit hexadecimal output. The value of the letter 'a' is 1fd5, while the value of the letter 'b' is d67e. The result of 'c' is the product of 'a' and 'b' multiplied together. According to this, the value of 'c' is 1AABB8D6.

Table 1: Result Comparison

Sr No.	Parameters	Previous work [1]	Proposed work
1	Area	320	287
2	Delay	4.92 ns	3.105 ns
3	Power	9.13 mW	8.2mW
4	PDP (Power delay product)	44.91	25.46

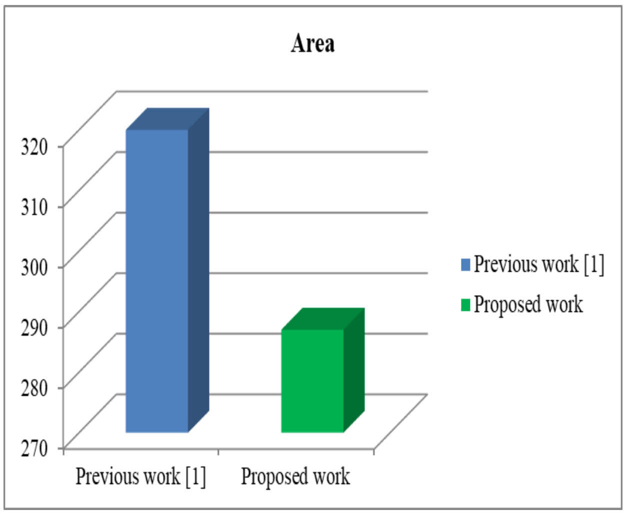


Figure 4: Comparison graph- Area

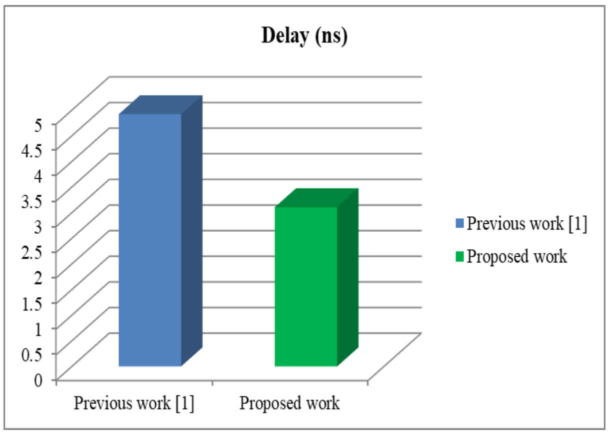


Figure 5: Comparison graph- Delay

Therefore, the proposed 64-bit booth multiplier provides the better results in terms of the parameters that were computed. In order for it to be useful in high-speed, low-area, and low-latency applications.

IV. CONCLUSION

The proposed VLSI implementation of the Approximate Radix-16 Booth Multiplier demonstrates significant improvements in area, delay, power consumption, and Power Delay Product (PDP) compared to previous work. The proposed design achieves a 10.31% reduction in area, a 36.9% reduction in delay, and a 10.18% reduction in power consumption. Most notably, the PDP is reduced by 43.3%, highlighting the efficiency of the design in balancing power and speed. These results validate the effectiveness of the optimized multiplier architecture in achieving high speed and low latency while maintaining low power consumption, making it ideal for modern VLSI applications in performance-critical domains.

REFERENCES

1. M. H. Haider and S. B. Ko, "Booth encoding based energy efficient multipliers for deep learning systems," in IEEE Transactions on Circuits and Systems II: Express Briefs, doi: 10.1109/TCSII.2022.3233923.
2. H. Zhang and S. -B. Ko, "Efficient Approximate Posit Multipliers for Deep Learning Computation," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 13, no. 1, pp. 201-211, March 2023, doi: 10.1109/JETCAS.2022.3231642.
3. B. K. Mohanty, "Efficient Approximate Multiplier Design Based on Hybrid Higher Radix Booth Encoding," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 13, no. 1, pp. 165-174, March 2023, doi: 10.1109/JETCAS.2022.3229831.
4. G. Park, J. Kung and Y. Lee, "Simplified Compressor and Encoder Designs for Low-Cost Approximate Radix-4 Booth Multiplier," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 3, pp. 1154-1158, March 2023, doi: 10.1109/TCSII.2022.3217696.
5. Q. Cheng et al., "A Low-Power Sparse Convolutional Neural Network Accelerator with Pre-Encoding Radix-4 Booth Multiplier," in IEEE Transactions on Circuits and Systems II: Express Briefs, doi: 10.1109/TCSII.2022.3231361.
6. K. Chen, C. Xu, H. Waris, W. Liu, P. Montuschi and F. Lombardi, "Exact and Approximate Squarers for Error-Tolerant Applications," in IEEE Transactions on Computers, doi: 10.1109/TC.2022.3228592.
7. F. Zhu, S. Zhen, X. Yi, H. Pei, B. Hou and Y. He, "Design of Approximate Radix-256 Booth Encoding for Error-Tolerant Computing," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 4, pp. 2286-2290, April 2022, doi: 10.1109/TCSII.2022.3148122.
8. T. Zhang et al., "Design of Majority Logic-Based Approximate Booth Multipliers for Error-Tolerant Applications," in IEEE Transactions on Nanotechnology, vol. 21, pp. 81-89, 2022, doi: 10.1109/TNANO.2022.3145362.
9. A. S. Roy, H. Agrawal and A. S. Dhar, "ACBAM-Accuracy-Configurable Sign Inclusive Broken Array Booth Multiplier Design," in IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 4, pp. 2072-2078, 1 Oct.-Dec. 2022, doi: 10.1109/TETC.2021.3107509.
10. Z. Aizaz and K. Khare, "Area and Power Efficient Truncated Booth Multipliers Using Approximate Carry-Based Error Compensation," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 2, pp. 579-583, Feb. 2022, doi: 10.1109/TCSII.2021.3094910.