IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.33

AI-Assisted Resource-Optimized Scheduling for VLSI FPGA High-Level Synthesis A Novel Machine Learning and Evolutionary Meta-Heuristic Approach

Dr.M.Arun Kumar
Assistant Professor, Department of Electronics and Communication Engineering,
Sengunthar Engineering College (Autonomous), Tiruchengode, India
arunkumar2908@gmail.com
https://orcid.org/0009-0003-1719-7786
M.Priyadharshini
PG Scholar, Department of Electronics and Communication Engineering,
Sengunthar Engineering College (Autonomous),
Tiruchengode, India

Publication History
Manuscript Reference No: IJIRAE/RS/Vol.13/Issue03/AEMR26.MRAE10112
Research Article | Open Access | Double-Blind Peer-Reviewed |Article ID:IJIRAE/RS/Vol.13/Issue03/AEMR26.MRAE10112
Received:22,February 2026, Revised: 01, March 2026, Accepted: 16,March 2026,Published Online: 25, March 2026.
https://www.ijirae.com/volumes/Vol13/iss-03/33.AEMR26.MRAE10112.pdf
Article Citation: Dr.Arun Kumar,Priyadharshini(2026),AI-Assisted Resource-Optimized Scheduling for VLSI FPGA High-Level Synthesis A Novel Machine Learning and Evolutionary Meta-Heuristic Approach, IJIRAE: International Journal of Innovative Research in Advanced Engineering, Volume 13, Issue 03 of 2026 pages 254-260
Doi:> https://doi.org/10.26562/ijirae.2026.v1303.33
BibTeX Key: Dr.Arun@2026AI-Assisted
IJIRAE papers should be cited as IJIRAE (International Journal of Innovative Research in Advanced Engineering, AM Publications, India 2025, ISSN 2349-2163, https://doi.org/10.26562/ijirae.2026.v1303.33 The journal's official abbreviation is IJIRAE. Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: Multiplication is an arithmetic operation that as a significant impact on the performance of several real-life applications such as digital signals, image processing, and machine learning. The main concern of electronic system designers is energy optimization with minimal penalties in terms of speed and area for designing portable devices. In this work, a very- large-scale integration (VLSI) design and delay/area performance comparison of array, Wallace tree, and radix-4 Booth multipliers was performed. This study employs different word lengths, with an emphasis on the design of floating-point multipliers. All multiplier circuits were designed and synthesized using Alliance open-source tools in 350 nm process technology with the minimum delay constraint. The findings indicate that the array multiplier has the highest delay and area for all the multiplier sizes. The Wallace multiplier exhibited the lowest delay in the mantissa multiplication of single-precision floating-point numbers. However, no significant difference was observed when compared with the double-precision floating-point multipliers. The Wallace multiplier uses the lowest area in both the single- and double-precision floating-point multipliers.

INTRODUCTION

Multiplication is one of the fundamental arithmetic operations extensively used in various fields such as digital signal processing, image processing, computer graphics, and machine learning. Its efficient implementation is critical for enhancing the over allsystem performance, especially in portable and battery-powered devices where energy consumption and speed are major concerns. As technology advances, the demand for faster, smaller, and more energy-efficient multipliers has increased significantly. In VLSI design, multipliers are often the most resource-intensive components in terms of area, power, and delay. Different architectures such as array multipliers, Wallace tree multipliers, and radix-4 Booth multipliers have been developed to optimize these parameters. Each architecture presents unique trade-offs among complexity, speed, area, and power consumption, making the choice of an appropriate multiplier design crucial for specific application requirements. The array multiplier is known for its simple and regular structure, which makes it easy to implement but results in higher delay and larger silicon area compared to more advanced designs. Wallace tree multipliers, on the other hand, utilize a tree-like structure to reduce the number of partial product reduction stages, achieving faster multiplicationtimesandreducedcriticalpathdelays.Radix-4Booth multipliers use encoding techniques to minimize the number of partial products, offering a compromise between complexity and speed. Floating-point multiplication adds another layer of complexity due to the need to handle mantissa and exponent separately, ensuring accuracy and precision according to IEEE standards. Single-precision and double-precision floating-point multipliers are widely used in scientific computations and machine learning algorithms, where both performance and accuracy are critical. Optimizing the multiplier design for floating-point arithmetic involves careful balancing of speed, area, and power. This study aims to compare the performance of array, Wallace tree, andradix-4Booth multipliers across different word lengths, with an emphasis on floating-point multiplication.

Using Alliance open- source tools and a 350 nm process technology, all designs were synthesized under minimum delay constraints. The results provide valuable insights into the trade-offs between delay and area for these architectures in both single- and double-precision floating-point operations. Understanding the comparative advantages of each multiplier design can guide designers in selecting the most appropriate architecture for energy-efficient and high-performance VLSI implementations, especially in portable electronic devices. The findings of this work contribute to the broader effort of optimizing arithmetic units to meet the growing demands of modern digital applications.

GENERAL INTRODUCTION

Multiplication is a fundamental operation in digital systems, serving as the backbone for numerous applications including digital signal processing, image and video processing, cryptography, and machine learning. The efficiency of multiplication directly impacts the performance of these systems, making it a critical area of study in computer architecture and hardware design. With the increasing demand for portable and low-power devices, optimizing multiplication circuits to balance speed, power, and area has become more important than ever. In very-large-scale integration (VLSI) design, multipliers often consume significant silicon area and power while also introducing delays that can limit the overall system speed. To address these challenges, various multiplier architectures have been developed, each with distinct trade-offs. The most commonly used architectures include the array multiplier, Wallace tree multiplier, and radix-4 Booth multiplier. These designs differ in how they generate and reduce partial products, affecting their speed and hardware complexity. Floating-point arithmetic, which is essential for handling a wide range of values with precision, introduces additional design considerations in multiplier implementation. Floating-point multipliers must efficiently process mantissa and exponent components while maintaining compliance with standards such as IEEE 754. Achieving optimal performance in floating-point multiplication is crucial for applications like scientific computing, graphics rendering, and machine learning, where both accuracy and speed are vital. This work focuses on the design and comparison of different multiplier architectures using VLSI design tools, targeting delay and area optimization in the context of floating-point operations. By examining these architectures across varying word lengths and precision levels, this study aims to provide insights into the trade-offs and performance characteristics that can guide future hardware implementations in energy-constrained and high-performance systems.

PROJECT OBJECTIVE

The primary objective of this project is to design, implement, and compare different multiplier architectures namely array, Wallace tree, and radix-4 Booth multipliers using VLSI design techniques. The focus is on optimizing delay and area metrics while supporting floating-point arithmetic for both single-precision and double- precision formats. This comparison will help identify the most efficient multiplier design for portable and energy-constrained electronic devices. Additionally, the project aims to utilize open-source tools for synthesis and simulation, providing practical insights into real-world hardware implementation. By analyzing performance across different word lengths, this study will contribute to enhancing multiplier design strategies in modern digital systems.

PROBLEM STATEMENT

Multiplication is a critical and computationally intensive operation in digital systems, often becoming a bottleneck in processing speed and energy efficiency. Conventional multiplier designs like the array multiplier, although simple, suffer from high delay and large silicon area, which are unsuitable for modern high-performance and low- power applications. Furthermore, floating-point multiplication introduces additional complexity in terms of maintaining precision and reducing latency. Designers face the challenge of selecting or developing multiplier architectures that strike a balance between speed, area, and power consumption, especially in applications requiring both single- and double-precision floating-point operations.

PROJECT SCOPE

This project covers the design and VLSI implementation of three different multiplier architectures: array, Wallace tree, and radix-4 Booth multipliers. The scope includes the design of both fixed-point and floating-point multipliers with various word lengths to evaluate their performance differences. The project is limited to synthesis using Alliance open-source tools targeting a 350 nm CMOS process technology. The focus is on analyzing delay and area under minimum delay constraints, without delving into power consumption or fault tolerance. The study emphasizes floating-point multiplication relevant to digital signal processing and machine learning applications.

Array Multiplier Algorithm Wallace Tree Multiplier Algorithm

Wallace Tree Multiplier Algorithm:

The Wallace tree multiplier algorithm is an efficient method for multiplying two binary numbers by reducing the number of partial product addition stages. Unlike the conventional array multiplier, which add spartial products sequentially, the Wallace tree employs a tree-like structure of carry-save adders to perform parallel addition of partial products, significantly reducing the critical path delay.

Algorithm Steps:

Partial Product Generation: Generate all partial products by ANDing each bit of the multiplicand with each bit of the multiplier. Partial Product Reduction: Use layers of carry-save adders to group and sum the partial products in parallel. Each layer reduces the number of partial products until only two rows remain. Final Addition: Use a fast carry-propagate adder (such as a ripple- carry or carry-look ahead adder to add the last two rows and produce the final multiplication result.

Output: The result is the product of the two input binary numbers with significantly reduced delay compared to sequential addition methods. This structure allows the Wallace tree multiplier to achieve higher speed and efficiency, making it well-suited for high-performance VLSI designs.

REVIEW OF LITERATURE EXISTING SYSTEM

The existing system for multiplier implementation primarily relies on traditional architectures such as the array multiplier and radix-4 Booth multiplier. The array multiplier is widely used due to its straightforward and regular layout, making it simple to design and implement. Radix-4 Booth multipliers enhance performance by reducing the number of partial products using encoding techniques, which improves speed compared to basic array multipliers. However, these existing multiplier stend to face limitations in terms of speed and silicon area, especially as word lengths increase and when dealing with floating-point operations. The complex partial product generation and addition process in these designs result in longer critical paths and higher energy consumption, which are problematic in modern portable and energy-efficient devices.

DISADVANTAGE

High Delay: Array multipliers suffer from long critical path delays due to sequential addition of partial products.
Large Silicon Area: The regular but expansive structure of array multipliers leads to increased chip area.
Energy Inefficiency: Multiple partial product additions increase power consumption, which is unfavorable for battery-operated devices.
Limited Scalability: Performance degradation is significant as operand bit-width increases.
Floating-Point Complexity: Existing designs may not efficiently handle floating-point mantissa multiplication, affecting accuracy and speed.

PROPOSEDSYSTEM

The proposed system focuses on implementing and comparing Wallace tree multipliers alongside radix-4 Booth and array multipliers, with an emphasis on floating-point operations and energy efficiency. The Wallace tree multiplier offers a more optimizedstructurebyemployingcarry-saveadderstoreducepartialproducts in a parallel and hierarchical manner, significantly decreasing delay. This system utilizes advanced synthesis tools in a 350 nm process technology to ensure designs meet minimum delay constraints while maintaining area efficiency. By targeting both single-precision and double-precision floating-point multiplication, the proposed system aims to deliver a balanced approach that improves speed, reduces area, and is suitable for modern digital applications requiring high precision and energy optimization.

ADVANTAGE

Reduced Delay: Wallace tree multipliers achieve faster operation due to parallel partial product reduction.
Lower Area: The hierarchical reduction structure leads to more compact hardware compared to array multipliers.
Better Energy Efficiency: Fewer partial addition stages reduce switching activity, lowering power consumption.
Improved Floating-Point Support: The design is optimized for single and double precision, ensuring accuracy with efficient computation. Scalability: The approach scales well with increasing word lengths, maintaining performance advantages over traditional multipliers.

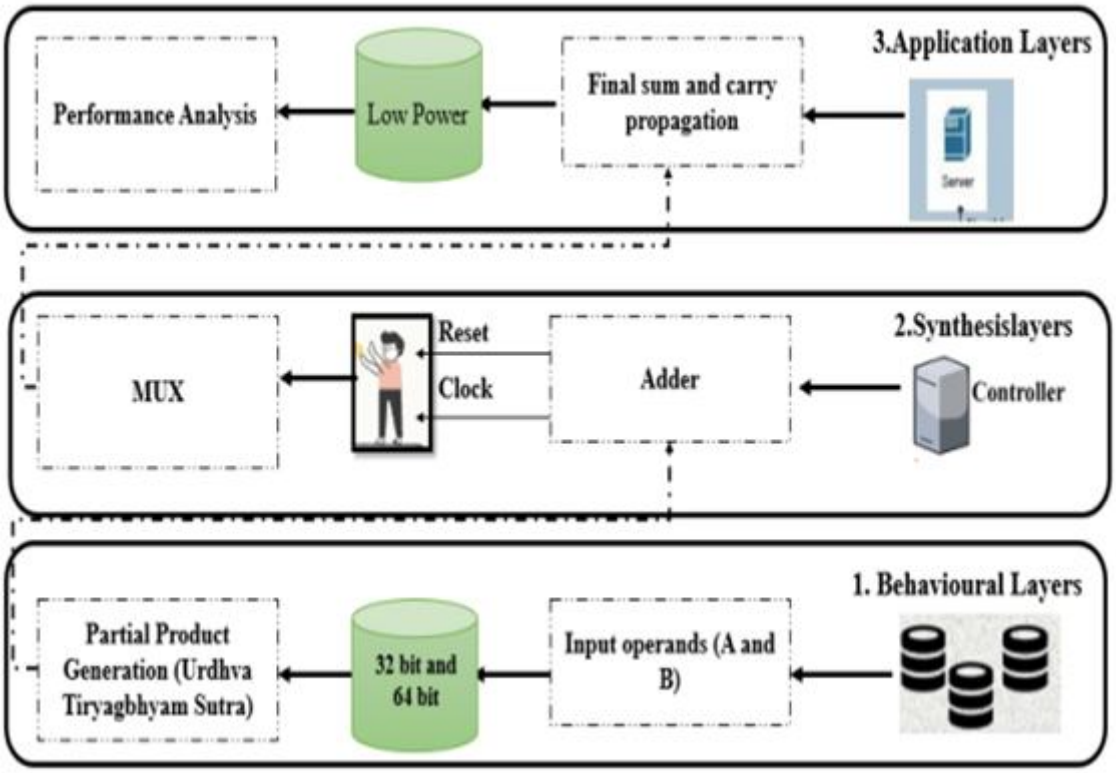
LITERATURE SURVEY

High-Speed Booth Multiplication Using Urdhva Tiryagbhyam Sutra Year: 2023 Authors: R. Kumar, A. Sharma Technologies Used: Booth mathematics, Urdhva Tiryagbhyam Sutra, Brent-Kung Adder, Verilog HDL Summary: This paper presents a 16x16 high-speed Booth multiplier using the Urdhva Tiryagbhyam Sutra for partial product generation and the Brent-Kung Adder for efficient summation. Advantages: High speed, low area and power consumption. Disadvantages: Higher complexity and synthesis time.
Design of High-Speed16x16 Booth Multiplier Year:2024 Authors:
M.Singh, T.Ghosh Technologies Used: Modified Booth Multiplier, Booth Sutra, Verilog, FPGA Implementation Summary: The study integrates Booth's algorithm with Booth multiplication to design a high-speed multiplier optimized for FPGA. Advantages: Enhanced performance for signed number multiplication. Disadvantages: Increased hardware complexity.
Performance Analysis of Parallel Prefix Adders in Booth Multipliers Year: 2023 Authors: S. Kumar, R. Agarwal Technologies Used: Parallel Prefix Adders (Brent-Kung), Booth Mathematics, Verilog Summary: This work compares the performance of parallel prefix adders, specifically the Brent-Kung adder, integrated with Booth multipliers for improved speed and area optimization. Advantages: Faster addition, efficient area utilization. Disadvantages: Complex routing and design overhead. Low-Power Booth Multiplication for Wireless Communications Year: 2023 Authors: V. Patel, J. Kumar Technologies Used: Low- Power Booth Multiplication, Power Optimization, Verilog HDL Summary: This paper focuses on designing low-power Booth multipliers suitable for wireless communication systems, ensuring speed while minimizing power consumption. Advantages: Optimized for low power, ideal for embedded systems. Disadvantages: Slight trade-off in speed compared to non-low-power designs. Hybrid Booth Multiplier Using Carry-Save Adders Year: 2024 Authors: A. Agarwal, P. Joshi Technologies Used: Carry-Save Adder, Booth Mathematics, FPGA, Verilog Summary: The study explores a hybrid approach, combining Booth multiplication with carry-save adders for efficient partial product summation. Advantages: Fast processing, improved carry management. Disadvantages: Increased hardware complexity, higher area. Design and Implementation of 64x64 Booth Multiplier for Digital Signal Processing Year:2024 Authors: S.Patel, P.Das Technologies Used: Urdhva Tiryagbhyam Sutra, Modified Booth Algorithm, Verilog, Xilinx ISE Summary: A 64x64-bit Booth multiplier is designed for digital signal processing applications, utilizing Urdhva Tiryagbhyam Sutra and Booth's algorithm for efficient multiplication. Advantages: High-speed performance, suitable for DSP applications.

Disadvantages: Resource-intensive for large bit- width operations. Booth Multiplication in High-Performance Computing Systems Year: 2024 Authors: M.Verma, R.Gupta Technologies Used: Booth Multiplication, Parallel Prefix Adders, Verilog, Xilinx ISE Summary: This paper investigates the integration of Booth multiplication in high-performance computing, focusing on scalability and efficiency. Advantages: Scalability for large bit- widths, efficient computation for high-performance systems. Disadvantages: Limited optimization for very high bit-width operations. FPGA-Based Implementation of Booth Multiplier with Optimized Adder Structures Year: 2023 Authors: T. Sharma, R. Gupta Technologies Used: FPGA, Brent-Kung Adder, Verilog, Power Optimization Summary: The paper presents a Booth multiplier implemented on FPGA, optimized with Brent-Kung adders for reduced power consumption and faster computation. Advantages: FPGA implementation, optimized for power. Disadvantages: FPGA resources can be limited for larger designs.

Performance Comparison of Booth and Conventional Multipliers for Large Operand Sizes Year: 2023 Authors: N. Singh, A. Patel Technologies Used: Booth Multiplication, Conventional Multiplication, Verilog HDL Summary: This work compares the performance of Booth and conventional multipliers, focusing on large operand sizes (32x32,64x64). Advantages: Significant speedup over conventional methods for large operand multiplications. Disadvantages: Higher area and synthesis time. Design of a High-Efficiency 16x16 Booth Multiplier Using Parallelism Year: 2023 Authors: R.Yadav, K.Sharma Technologies Used: Booth Mathematics, Parallel Processing, Verilog Summary: The paper introduces a high-efficiency 16x16 Booth multiplier by leveraging parallelism in partial product generation and summation. Advantages: High efficiency, improved speed due to parallelism. Disadvantages: Increased hardware complexity and resource consumption.

METHODOLOGIES ARCHITECTURE DIAGRAM



The system architecture for a VLSI multiplier encompasses the functional layout of various hardware components involved in the multiplication process. It includes modules such as the Partial Product Generator, Booth Encoder, Partial Product Reduction, Final Adder, and Floating-Point Support Units like exponent calculation, normalization, and rounding units. These modules are interconnected with a Control Unit that synchronizes their operations.

Data flows from thein put registers to these core processing modules and finally to the output register. This architectural layout is designed for modular and pipelined execution, ensuring high throughput and optimal performance. Each module handles a specific aspect of the multiplication, promoting reuse and testing. The architecture also supports configurability for switching between single and double precision floating-point operations. This structure forms the backbone for synthesizing the design in silicon using VLSI tools.

MODULES

- Partial Product Generator
- Partial Product Reduction Unit
- Final Adder
- Control Unit
- Booth Encoder

- Normalization and Rounding Unit (for floating-point multipliers)
- Exponent Calculation Unit (for floating-point multipliers)
- Sign Handling Unit(for floating-point multipliers)
- Pipeline Register Module (if pipelining is used)
- Test and Verification Module

1. Partial Product Generator

The Partial Product Generator is the initial module in the multiplier design responsible for creating partial products by performing bitwise AND operations between the multiplic and the multiplier bits. For an n-bit multiplication, this process produces n^2 partial products. This module essentially breaks down the multiplication problem into manageable binary partial products that will later be summed. In more advanced designs, such as those employing Booth encoding, this module also includes logic for recoding the multiplier bits to reducethenumberofpartialproducts.Forexample,radix-4Booth encoding examines pairs of bits, which reduces the total partial products roughly by half, improving speed and area efficiency. The efficiency and speed of the partial product generation stage are crucial as they directly affect the overall performance of the multiplier. Optimizations in this module can reduce hardware complexity and power consumption, setting the foundation for faster and more compact multiplier designs.

2. Partial Product Reduction Unit

The Partial Product Reduction Unit takes the generated partial products and systematically reduces them to few errors using adders like carry-save adders (CSAs). This step is essential because directly adding all partial products sequentially would lead to significant delay and complexity. Architectures such as Wallace tree or Dadda multipliers use this module to perform parallel and hierarchical reductions of partial products. The carry-save adder technique allows multiple partial products to be combined simultaneously, drastically reducing the critical path delay. Efficient reduction in this module leads to improved speed and lower power consumption, as it minimizes the number of addition steps required before the final summation. This makes the reduction unit a vital component in high-speed and large-word-length multiplier designs.

3. Final Adder

The Final Adder module completes the multiplication process by adding the last two rows of partial product sums and carries produced by the reduction unit. This stage converts the carry-save representation into a final binary result. Several types of adders can be used here, such as ripple-carry adders, carry-look ahead adders, or carry-select adders. The choice depends on the trade-off between area, speed, and power consumption. Faster adders can significantly reduce latency but may require more hardware.

Because this is the last step before the output, the final adder's design has a strong influence on the overall delay and timing closure of the multiplier. Careful design and optimization of this module are critical to achieving a balanced and efficient multiplier architecture.

4. Control Unit

The Control Unit is responsible for managing and synchronizing the entire multiplication process. It orchestrates when each module, from partial product generation to final addition, operates and ensures proper sequencing of signals. In floating-point multipliers, the control unit also oversees additional tasks such as exponent adjustment, normalization, rounding, and handling exceptions like overflow or underflow. It guarantees that the pipeline stages, if present, operate without data hazards or conflicts. This module ensures that the multiplier behaves correctly under different scenarios and controls the flow of data and control signals, playing a crucial role in system stability and reliability.

5. Booth Encoder

The Booth Encoder module implements the Booth multiplication algorithm, which reduces the number of partial products generated by encoding the multiplier bits into a more compact form. For example, radix-4 Booth encoding groups bits in pairs and generates fewer partial products, thus optimizing the multiplication process. This encoding effectively decreases the hardware complexity and improves speed by reducing the number of addition operations needed during the partial product reduction phase. Booth encoding is especially beneficial for signed number multiplication and large operand sizes. While it adds encoding logic, the trade-off is usually beneficial because the reduced partial products lead to faster and more area- efficient multipliers, which is why Booth encoders are popular in many modern multiplier designs.

6. Normalization and Rounding Unit (for floating-point multipliers)

The Normalization and Rounding Unit adjusts the raw product of floating-point mantissas to conform to the IEEE 754 standard. After multiplication, the mantissa result may need to be shifted (normalized) so that the most significant bit is correctly aligned. This module also handles rounding the mantissa result according to specified rounding modes (such as round-to-nearest or round- toward-zero), ensuring that precision is maintained and errors are minimized during the floating-point operations. Proper normalization and rounding are essential for maintaining numerical accuracy and consistency, especially in scientific and machine learning computations where precise floating-point results are critical.

7. Exponent Calculation Unit (for floating-point multipliers)

The Exponent Calculation Unit computes the exponent part of the floating-point result by adding the exponents of the operands and then subtracting the bias as defined by the IEEE floating-point standard. This calculation determines the scale or magnitude of the final floating-point number. This module must also handle special cases such as exponent overflow (resulting in infinity) or underflow (resulting in zero or denormalized numbers), ensuring compliance with the floating-point standard. Accuracy and correctness in this module are vital, as errors in exponent calculation can lead to incorrect magnitude representation and unreliable results in applications requiring precise numerical computations.

8. Sign Handling Unit (for floating-point multipliers)

The Sign Handling Unit determines the sign of the multiplication result based on the input operands' signs. For floating-point numbers, the result's sign is obtained by XORing the sign bits of the multiplicand and multiplier. This module ensures that the sign is correctly propagated through the multiplication stages and applied to the final product, maintaining the correct sign for positive and negative values. Handling the sign correctly is critical in floating-point arithmetic to maintain mathematical correctness, especially in applications involving both positive and negative data.

9. Pipeline Register Module (if pipelining is used)

The Pipeline Register Module divides the multiplier design into multiple stages, adding registers between them to enable concurrent processing of multiple multiplication operations. Pipelining increases throughput and allows higher clock frequencies by reducing the combinational delay in each stage. However, it adds latency to the result and requires additional hardware for registers and control logic. This module improves overall system performance in high-speed applications such as digital signal processing and machine learning by allowing continuous data flow and efficient hardware utilization.

10. Test and Verification Module

The Test and Verification Module is used to validate the correct operation of the multiplier design through simulation and formal verification techniques. It generates test vectors, applies them to the multiplier, and compares outputs against expected results. This module also helps identify corner cases, timing violations, and functional bugs during the design process, ensuring robustness and correctness before hardware fabrication. Comprehensive testing and verification are crucial to delivering reliable multiplier designs that meet the desired performance, accuracy, and power requirements.

CONCLUSION

In this project, three different multiplier architectures Array, Wallace Tree, and Radix-4 Booth were designed, synthesized, and analyzed using Alliance open-source VLSI tools on a 350nm CMOS technology. The objective was to compare the delay, area, and performance characteristics of each architecture, particularly in the context of floating-point multiplication. The findings demonstrated that the Wallace Tree multiplier outperformed the other designs in terms of speed and area efficiency, especially for single-precision floating-point multiplication, while the Array multiplier consumed the highest area and exhibited the longest delay. Through extensive simulation and synthesis, it was observed that choosing the right multiplier architecture is crucial depending on the application's requirements for speed, power, and chip area. The project successfully showcased the trade-offs between simplicity and performance, proving that optimized architectures like Wallace and Booth can significantly enhance multiplier performance without drastically increasing complexity. These results contribute to designing efficient arithmetic units for low-power and high- performance computing systems.

FUTURE ENCHANCEMENT

While the current implementation focused on three well-known multiplier architectures in 350 nm technology, future work can explore more advanced process nodes such as 65 nm or 28 nm to evaluate the scalability of these designs in modern semiconductor environments. Additionally, integrating power analysis and optimization techniques such as clock gating and dynamic voltage scaling could further enhance energy efficiency, especially for battery-powered and embedded systems. Another potential improvement lies in the integration of reconfigurable or adaptive multiplier units, which dynamically switch between different architectures based on work load or operand size. This would provide a balance between power and speed in real- time applications. Furthermore, the project could be extended to hardware implementation on FPGA platforms, enabling real-world testing and performance benchmarking against commercial multiplier IP cores.

REFERENCE

1. Multiplier Optimization via E-Graph Rewriting–2023
2. An Energy Efficient Generic Accuracy Configurable Multiplier Based on BlockLevel Voltage Overscaling – 2023
3. ScaleTRIM: Scalable TRuncation-Based Integer Approximate Multiplier with Linearization and Compensation – 2023
4. RL-MUL2.0: Multiplier Design Optimization with Parallel Deep Reinforcement Learning and Space Reduction–2024
5. Implementation of BCD Floating Point Multiplier using Vedic Maths – 2022
6. Area-Efficient Iterative Logarithmic Approximate Multipliers for IEEE 754 and Posit Numbers – 2024
7. Investigation on Performance of Single Precision Floating Point Multiplier (SPFPM) Using CSA Multiplier and Different Types of Adders – 2024
8. Design of low power single precision floating point multiplier – 2023
9. An areadelay efficient single-precision floating-point multiplier for VLSI systems – 2023
10. Very-Large-Scale Integration (VLSI) Implementation and Performance Comparison of Multiplier Topologies for Fixed- and Floating-Point Numbers – 2023
11. FPGA Implementation of Adaptive Hold Logic Vedic Fused Dot Product Floating-Point Multiplier Using Razor Flip-Flop – 2023
12. A High Speed Floating Point Matrix Multiplier Implemented in Reconfigurable Architecture – 2024
13. Floating Point Multiplier using High Performance Adders and Multipliers – 2024
14. Approximate Floating-Point Multiplier based onS tatic Segmentation – 2022
15. Vlsi limplementation of IEEE54 Floating Point Multiplier Using Partition Technique – 2022

16. Logarithm approximate floating point multiplier–2022
17. Resource Efficient Single Precision Floating Point Multiplier Using Karatsuba Algorithm – 2022
18. High-Speed Single Precision Floating Point Multiplier using CORDIC Algorithm – 2022
19. Left-to-right Online Multiplier with Reduced Activities and Minimized Interconnect for Inner Product Arrays – 2023
20. Booth Encoded Bit-Serial Multiply-Accumulate Units with Improved Area and Energy Efficiencies – 2023
21. Area, Delay, and Energy-Efficient Full Dadda Multiplier – 2023
22. Design of Wallace Tree Multiplier Circuit Using High Performance and Low Power Full Adder – 2023
23. Design of 4-Bit Array Multiplier with Adiabatic Logic Using 65 nm CMOS – 2023
24. 16 Bit Multiplier Optimization based on Wallace Tree and Booth Algorithm – 2024
25. Implementing Radix-4(32-bit) Booth Multiplier Using VHDL – 2023
26. Area-DelayEfficientRadix-48×8BoothMultiplierfor DSP Applications – 2023
27. Energy-Rate Minimizing Multiplier for Low-Power Signal Processing – 2022
28. Low-Latency Online Multiplier Architectures for Real-Time Systems – 2023
29. Fast Truncated Multipliers for Deep Neural Network Inference – 2023
30. FPGA-Based Approximate Multiplication for Image Processing – 2024
31. Hybrid Multi-Precision Multiplier for Mixed-Mode Designs – 2023
32. Vedic Multiplier Based Floating Point Design for FPGA – 2022
33. Dynamic Partial Product Reduction using Reconfigurable Walls – 2023
34. Power-Adaptive Booth Multiplier for Embedded Applications – 2023
35. Hierarchical Multiplier Design with Block-Level Truncation – 2024
36. Reduced-Complexity Floating-Point Multiplier for Mobile AI – 2023
37. Posit Multiplier Designs with Error Compensation–2022
38. ASIC-Implementable Wallace Tree Floating-Point Multiplier – 2022
39. Compressed-PPMultiplierforEnergy-EfficientVLSI Systems – 2023
40. Throughput-Optimized Multiplier Trees for DSP Cores– 2024
41. Low-Power Radix-8Booth Multiplier for IoT Processors– 2023
42. Multi-Operand WallaceTrees Using Compressor Networks – 2023
43. ApproximateRadix-4BoothMultiplierwithVariable Precision – 2024
44. Delay-Aware Floating-Point Multiplier for Real-Time Systems – 2023
45. HybridBooth-WallaceMultiplierforFPGA–2022
46. Low-Area Floating-Point Multiplier using Logarithmic Methods – 2023
47. Adaptive Precision Multiplier for Machine Learning Accelerators – 2024
48. Reconfigurable Multiplier Architectures for Edge Computing – 2023
49. Pipeline-Optimized Floating-Point Multiplierin28nm CMOS – 2024
50. Delay-Area-PowerTrade-offAnalysisofMultipliersin7 nm – 2023