Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

IMPLEMENTATION AND COMPARISON OF 64-BIT MULTIPLIERS OF UNSIGNED NUMBER THROUGH REDUCTION SCHEME

Ayesha Nasir[1], Usama Anwar[2], Muhammad Aqeel Aslam[*3]

[1,2,*3]*School of Engineering & Applied Sciences, Electrical Engineering Department, GIFT University, Gujranwala, Pakistan*

[1]ayeshanasir2019@gmail.com,[2]kusama296@gmail.com, [*3]aqeel.aslam@gift.edu.pk

DOI: https://doi.org/10.5281/zenodo.20813553

Keywords
FPGA, Logic Cells, Multiplier, Reduction Methods

Article History
Received: 05 January 2026
Accepted: 18 February 2026
Published: 16 March 2026

Corresponding Author: *
Muhammad Aqeel Aslam

Abstract
Multiplication algorithms good multiplication algorithms are critical towards improving the overall performance of computer systems because multiplication is one of the basic procedures in digital computing. This study presents our analysis and comparison of three 64-bit multiplication algorithm Booth multiplier, Montgomery multiplier using Carry Look-Ahead (CLA) adders and Wallace Tree multiplier using both full and half adders. The analysis investigates the execution time, parameter of area utilization of both multiplier systems and exhaustively evaluates their pros and cons in practice. In our results, Wallace Tree Multiplier is a better option because of its speed and area utilization hence it is preferable to use in the high-performance computing systems where minimal time was saved in its completion. This study offers much information regarding the best selection of multiplication approach to apply to a particular scenario, which forms the foundation of enhancing mathematical functions in the digital world.

INTRODUCTION

Most physical systems applications require real-time operations to interface the high speed. limitations. One of the most is programmable hardware structure such as FPGA (Field Programmable Gate Array). powerful tools on preparing systems, which require real-time functions [1]. The FPGA technology is ever-changing. it is making strides and is rapidly becoming an essential element of the modern embedded system. FPGA devices are advantageous for prototyping because they provide high performance hardware at a lower price than application specific integrated circuits (ASIC) [2]. The best prototype tools are FPGAs. The transition from the prototype to the finished product is significantly smaller and simpler to manage when the FPGA is incorporated in the final design [3]. Since the same FPGA may be used in several designs, stocking costs are decreased. Basic programmable logic cells (LC), which make up the matrix's core and are surrounded by programmable I/O (input, output) cells, are the foundation of FPGA architecture. The routing channels contain the programmable interconnect as shown in Figure 1. Each of the three primary functionalities (logic cells, programmable interconnect, and programmable I/O) will have different design specifications depending on the vendor [4].

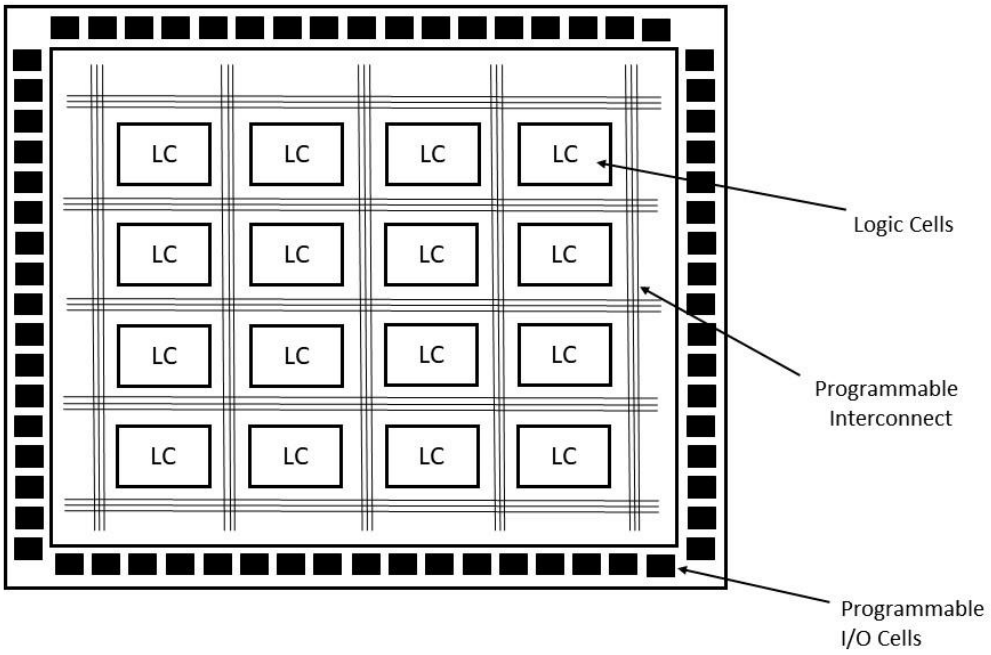

Figure 1. Generic Architecture for FPGA

Many methods used in scientific calculations involve MULTIPLICATION, a computation-intensive technique. Multiplication is still one of the most fundamental arithmetic operations having significant applications in a wide range of fields, from embedded systems to high-performance computers, in the constantly changing world of digital computation. The efficiency of multiplication is a complex task and it is of great importance to the effectiveness of the same. functionality Digital circuits and computers [5]. By offering an extensive comparative analysis of three popular methods -the Booth Multiplier, Montgomery Multiplier, and Wallace Multiplier--this research paper probes into in-depth details of multiplication algorithms. There are unique attributes in each of these number multiplication methods, Advantages, and disadvantages that influence its use on different purposes. Selecting the most appropriate method in a particular. situation, be it in the case of performance enhancement, power minimization or space constraint, needs to know about these differences, so as to provide a thorough analysis of different multiplication. algorithms, performance measures, hardware implementations and this paper shall consider them. operational concepts

Literature Review

Multiplication is an arithmetic operation in digital computing, and useful methods of multiplication. are necessary in order to enhance functionality of various applications. In the course of time, several multiplication. algorithms, which can meet different computing requirements [6], have been developed. To prepare the ground to our comparison. research, we discuss important literature on the Booth, Montgomery and Wallace multipliers in this section.

2.1 Booth Multiplier:

Andrew Donald Booth contributed significantly to the theory of multiplication in 1950 with a concept known as the Booth. algorithm. It enhances efficiency by reducing the level of unfinished goods manufactured in the process of. multiplication.[7]. [Booth, 1950], One can multiply two signed binary values that are normally represented in two-s complement representation. another with a hardware method and implementation scheme is a

booth multiplier. In digital circuits, It is a good way of especially those associated with microprocessors and digital signal processors (DSPs). doing multiplication operations. [10]. The Booth multiplier was multiplied by searching and eliminating unnecessary additions. method decreases the number of additions of a partial product required in multiplying two binary numbers. As a Outcome, multiplication can be accomplished both faster and using less hardware than using the traditional shift and-add multipliers as in Figure 2. The Booth multiplier reduces the amount of partial products which. must be inserted, especially where the multiplier has prolonged 0s or 1s series. Quicker multiplication processes. are the outcome and this is particularly useful in hardware implementation such as microprocessors and DSPs where. speed and efficiency are important. In general, through the patterns of multiplier and minimization of the addition, it was possible to reduce the number of addition. computations required, the method used by Booth gives an ingenious means of optimizing binary multiplication [11].

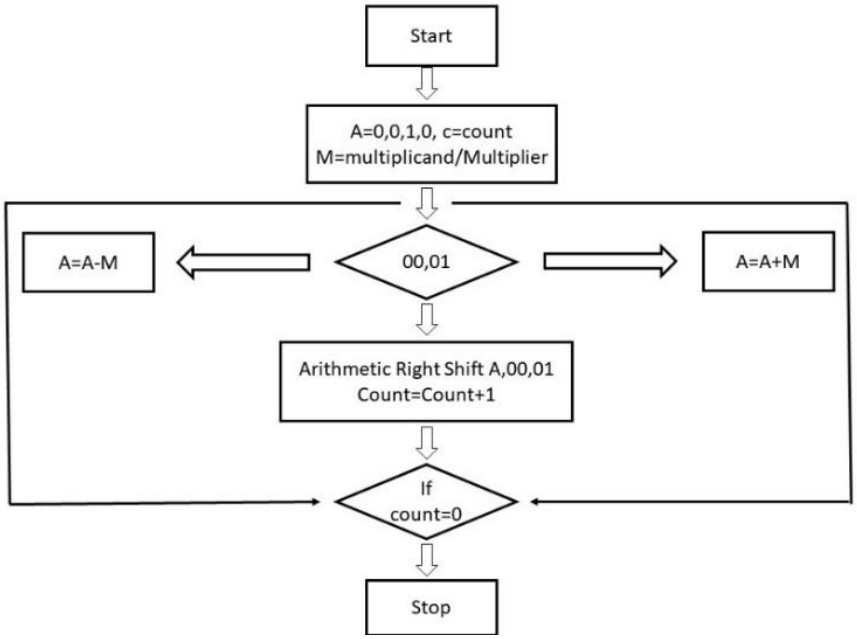

Figure 2. is the flow chart of the booth multiplier [9]

2.2 Wallace Multiplier:

A Wallace Tree, often referred to as a Wallace Tree Multiplier, is a rapid binary multiplication digital circuit. It is an effective way to multiply two binary values, especially when done in hardware. Andrew D. Wallace created the Wallace Tree Multiplier in the 1960s [13].

2.2.1 Full Adder:

The Wallace Tree Multiplier and other arithmetic and logical operations are implemented using a Full Adder whose logical circuit is shown in Figure 3, an essential digital logic circuit. Three binary inputs are added as its main task, the two binary numbers are A and B. The carry bit from the preceding addition is called Cin (Carry-in). Resulting in two outputs: a sum and a carry_out output its logical expressions are shown in equation (1) and (2).

Logical equation for output of full adder:

$$\text{Carry}_{\text{out}} = \text{A.B} + \text{carry\_in.}(\text{A} \oplus \text{B}) \quad (1)$$

$$\text{sum} = (\text{A} \oplus \text{B}) \oplus \text{Carry\_in} \quad (2)$$

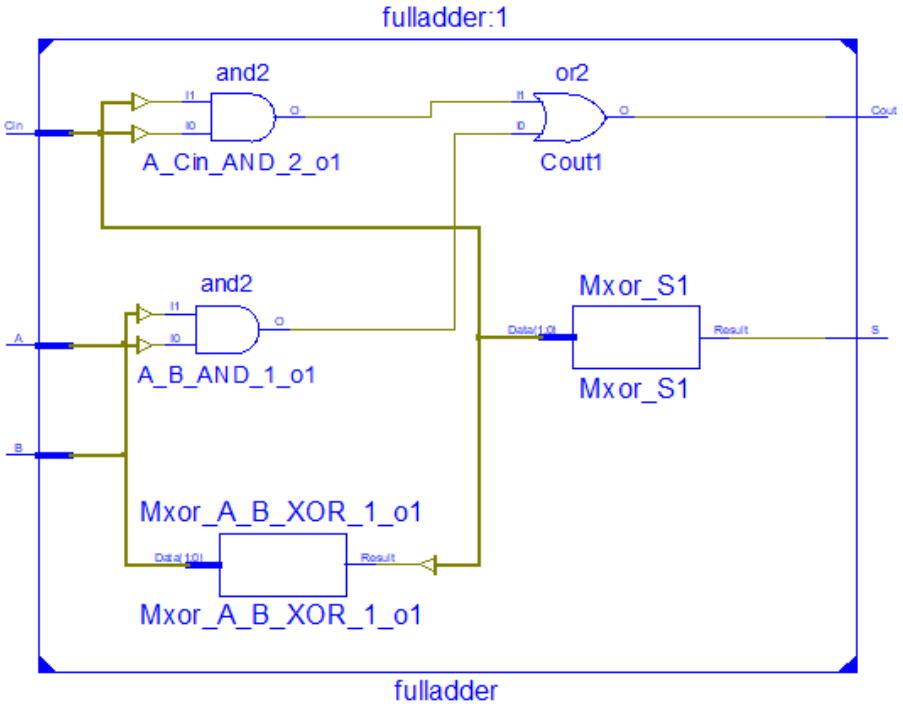

Figure 3. Logical Circuit of Full Adder

Based on the operations of a Full Adder, the Table 1. displays every conceivable combination of the inputs (A, B, and Carry_in) and the associated outputs (Sum and Carry_out) of Full Adder.

Table 1. Truth Table of Full Adder

A	B	Carry-In	Sum	Carry_Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1

2.2.2 Half Adder

The Wallace Tree Multiplier and other arithmetic operations are implemented using a basic digital circuit called a half adder, whose logical circuit is shown in Figure 4. It is mostly used to add two binary digits (bits) A and B and outputs the sum and carry that are produced during addition, its logical expressions are shown in equation (3) and (4).

Logical expression [14]. for sum and carry for half adder is:

$$\text{sum} = A \oplus B \quad (3)$$

$$\text{Carry} = A.B \quad (4)$$

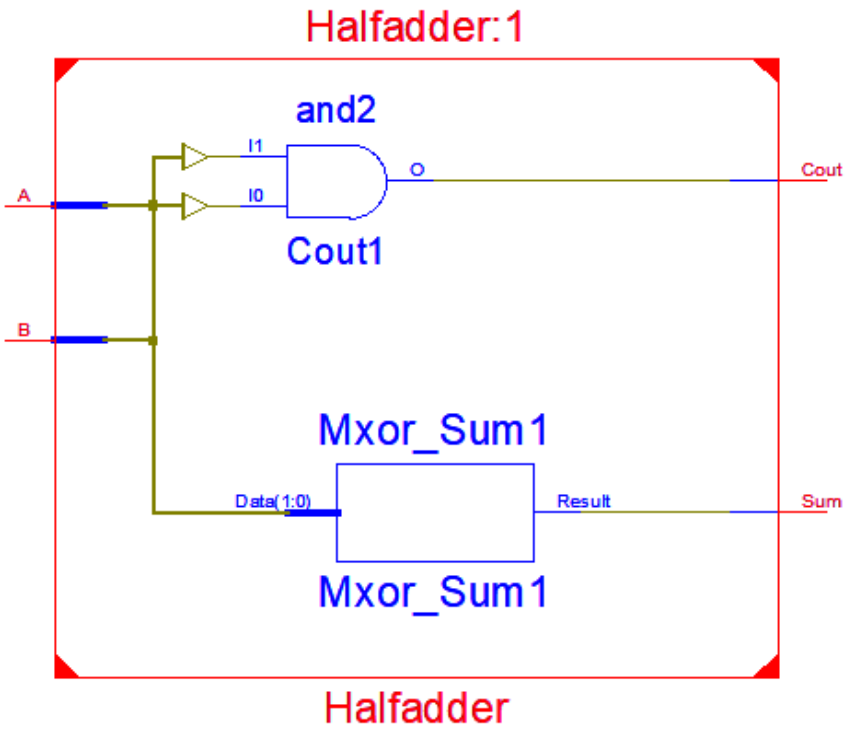

Figure 4. Logical Circuit of Half Adder

Based on the two input bits (A and B), Table 2. displays the half-adder's outputs (Sum and Carry).

Table 2. Truth Table of Half Adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The Wallace Tree Multiplier effectively produces partial products by using full adders and half adders, and then reduces these partial products into the final result using a tree-like structure as shown in Figure 5. This method is applicable to the design of digital circuits since it facilitates the use of hardware to do binary multiplication quickly. performances across minimizing propagation delay. This is especially suitable for applications where quick multiplication is essential, like digital signal processing and microprocessors [12].

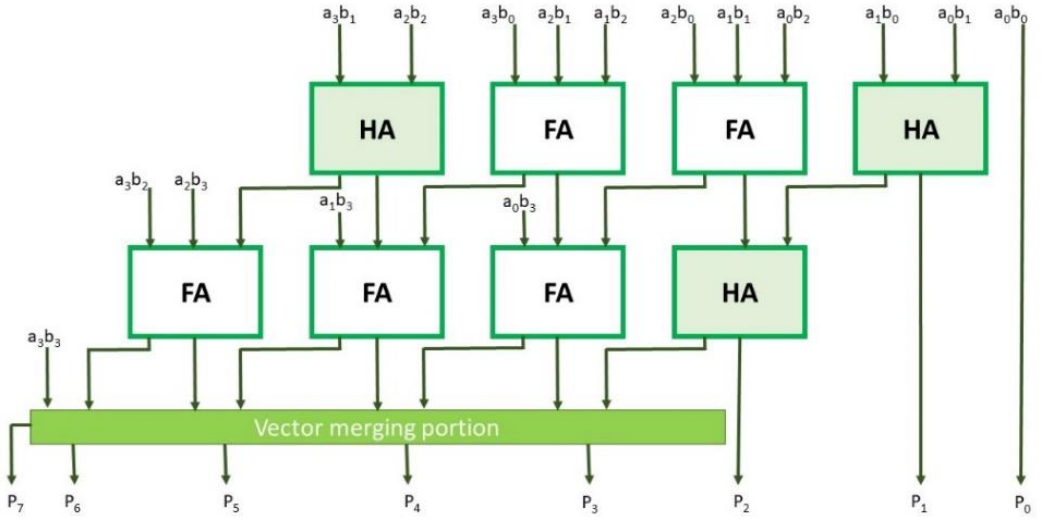

Figure 5. Wallace Multiplier

2.3 Montgomery Multiplier

Montgomery reduction, another name for the Montgomery multiplication algorithm, is a technique for modular multiplication that has benefits in some cryptographic applications, especially when working with big numbers. In the 1980s, Peter L. Montgomery created it.[8], Accelerating modular multiplications, which are often used in public-key cryptography techniques like RSA and ECC (Elliptic Curve Cryptography), is the main goal of the Montgomery multiplication method. It substitutes a more effective operation that minimizes the number of multiplications for conventional modular multiplication [2]

Modular arithmetic procedures need fewer modular multiplications when using the Montgomery multiplier. To compute the answer in conventional modular multiplication, several modular multiplications and modular reductions (division operations) techniques as shown in Figure 6. are applied. The Montgomery multiplier, on the other hand, just requires a single modular multiplication at the end, greatly reducing the computational work.

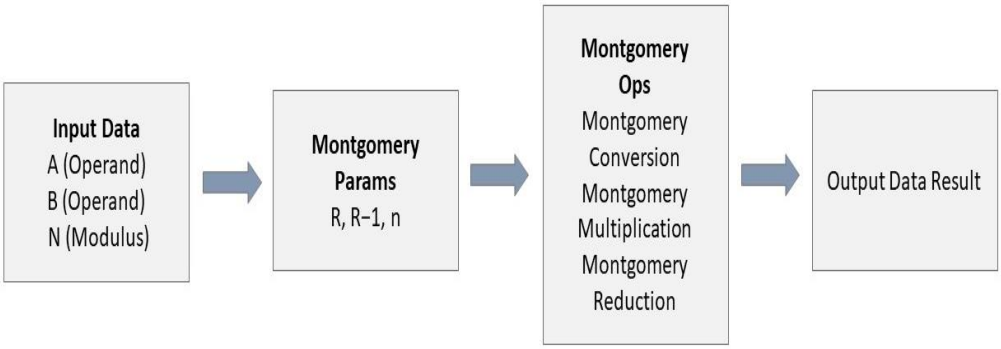

Figure 6. Montgomery Multiplier Block Diagram

2.3.1 Carry Look-Ahead Adder:

A digital circuit called a carry look-ahead adder (CLA adder) as shown in Figure 7. It is used in computer math to quickly add binary numbers. It is intended to speed up the computation of the carry-out (carry), which is the process of moving from one bit location to the next, during addition operations. A CLA adder predicts carries in parallel, leading to noticeably quicker addition than ripple-carry adders, which calculate carries sequentially from the least significant bit (LSB) to the most significant bit (MSB).

Logical expression [15] for sum and carry for carry Look-Ahead Adder is:

$$sum = A \oplus B \quad (5)$$

$$G = A \,\&\, B \text{ (bitwise AND)} \quad (6)$$

$$P = A \mid B \text{ (bitwise OR)} \quad (7)$$

$$Carry - out = G + (P \,\&\, Cin) \quad (8)$$

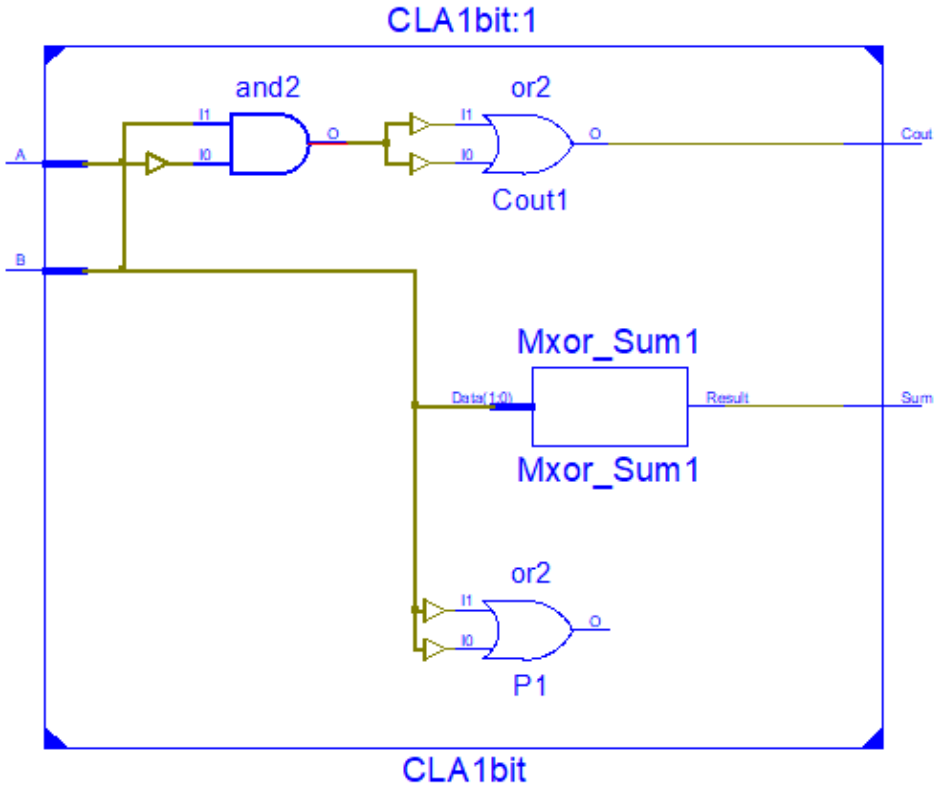

Figure 7. Logic Circuit of Carry Look-Ahead Adder

Table 3. illustrates the behavior of a 1-bit CLA adder for all possible input combinations, showing the corresponding values for the Generate (**G**), Propagate (**P**), sum (**Sum**), and carry-out (**Cout**) signals.

Table 3. Truth Table for CLA Adder

A	B	G	P	Sum	Cout

0	0	0	0	0	0
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	1

A computational technique called the Montgomery Multiplier, which makes use of the Carry Look-Ahead (CLA) adder, is an important number theoretic tool. cryptography, is improved. In doing so, it is possible to accomplish modular multiplication, which plays a very important role in cryptographic methods. similar to RSA and elliptic curve cryptography-increases its performance. To reduce propagation time, CLA adders employ a network of logic gates to forecast carry bits in parallel. This parallel carry prediction is a crucial tool for cryptographic applications since it coincides with the modular multiplication process and speeds things up significantly [16]

Simulation Results:

3.1 Results of 64-bit Booth Multiplier

Figure 8. shows the RTL (Register-Transfer Level) view of 64-bit Booth multiplier that uses Booth encoding to effectively perform the multiplication of two 32-bit operands (M and R). The circuit is made up of clock-synchronized input and output registers, Booth encoding logic to determine the necessary arithmetic operations, a partial product generator to calculate partial products based on encoding, and an accumulator to add up these partial products to produce the final result. When the outcome is reliable and suitable for use, a valid signal is created, and a Reset signal enables the internal state to be reset.

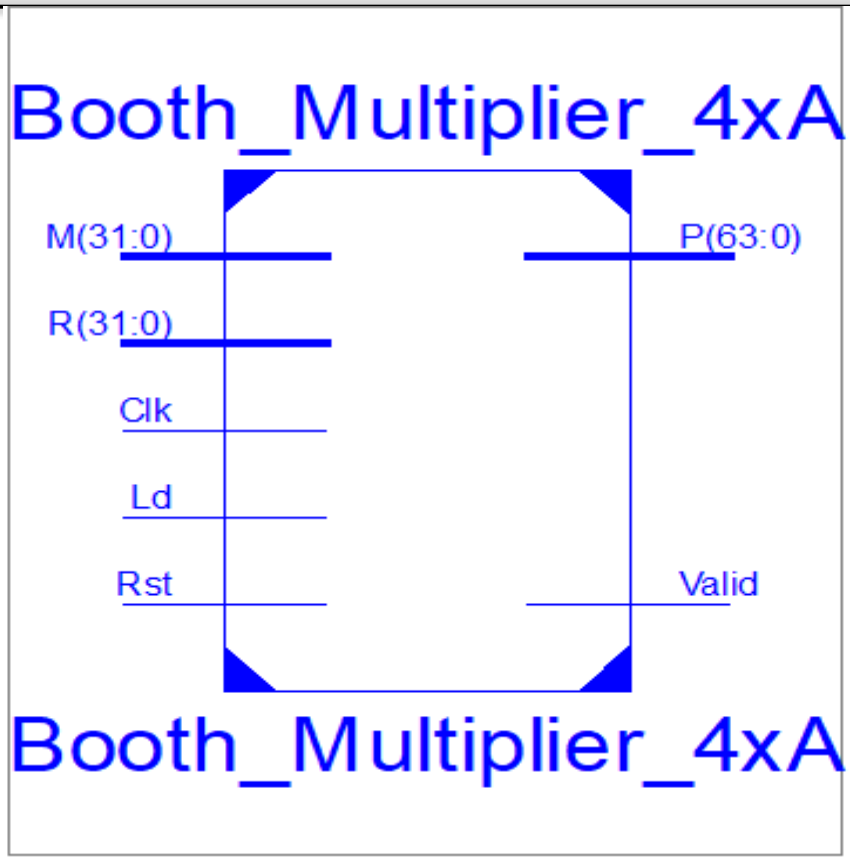

Figure 8. RTL view of 64 Bit Booth Multiplier

Figure 9. shows the output result of the 64-Bit booth multiplier

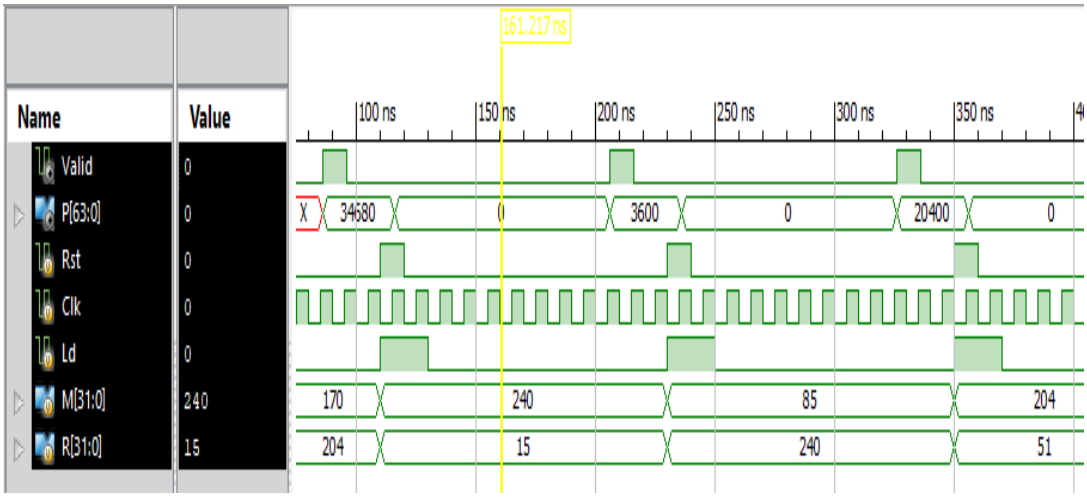

Figure 9. Output waveform of 64 Bit Booth Multiplier.

3.2 Results of 64-bit Wallace Multiplier

Figure 10. shows the RTL view of 64-Bit Wallace multiplier, The RTL view of a Wallace multiplier uses 32-Bit registers to store input values in a and b, bit-wise multiplication to create partial products in parallel, efficient reduction of these partial products using a tree structure, and computing the final 64-bit output result, C.

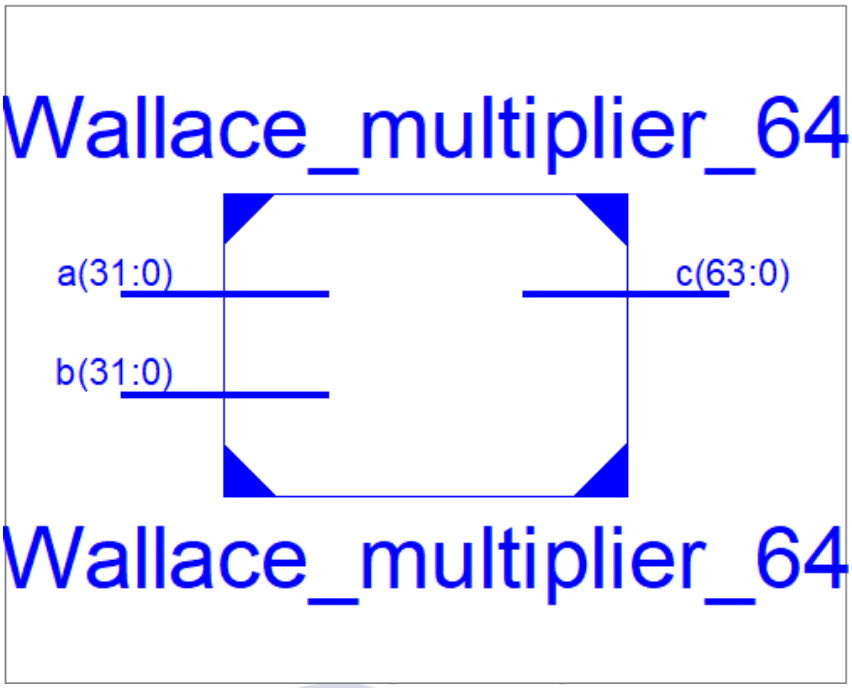

Figure 10. RTL View of 64 Bit Wallace Multiplier

Figure 11. shows the output Waveform of 64-Bit Wallace Multiplier

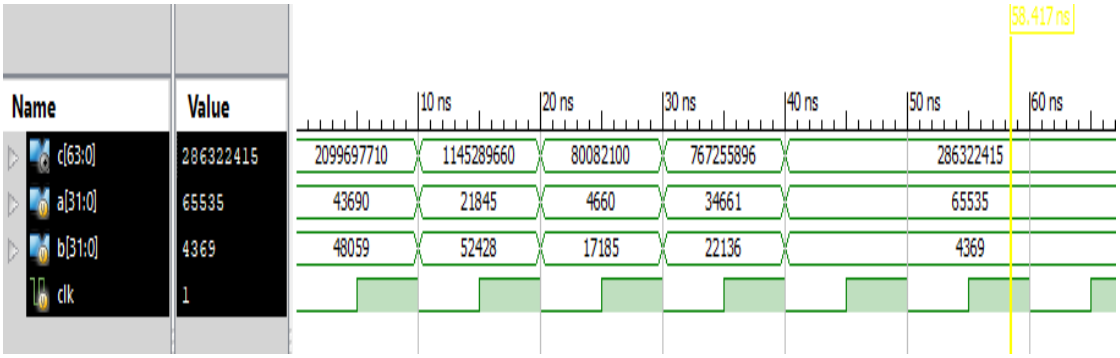

Figure 11. Output waveform of 64 Bit Wallace Multiplier

3.3 Results of 64-bit Montgomery Multiplier

Figure 12. shows the Register-Transfer Level (RTL) view of a Montgomery multiplier with 32-bit inputs in1 and in2, an 11-bit modulus n, clock signal (clk), enable, reset, and 36-bit output mult_result along with a mult_done flag. The operands in1 and in2 undergo Montgomery multiplication while the enable signal is active, with the outcome confined to the modulo field specified by n. When necessary, reset initializes the multiplier while mult_result saves the 36-bit result and mult_done indicates completion. These operations are synchronized by

clk. A crucial part in cryptographic and modular arithmetic processes is played by Montgomery multipliers.

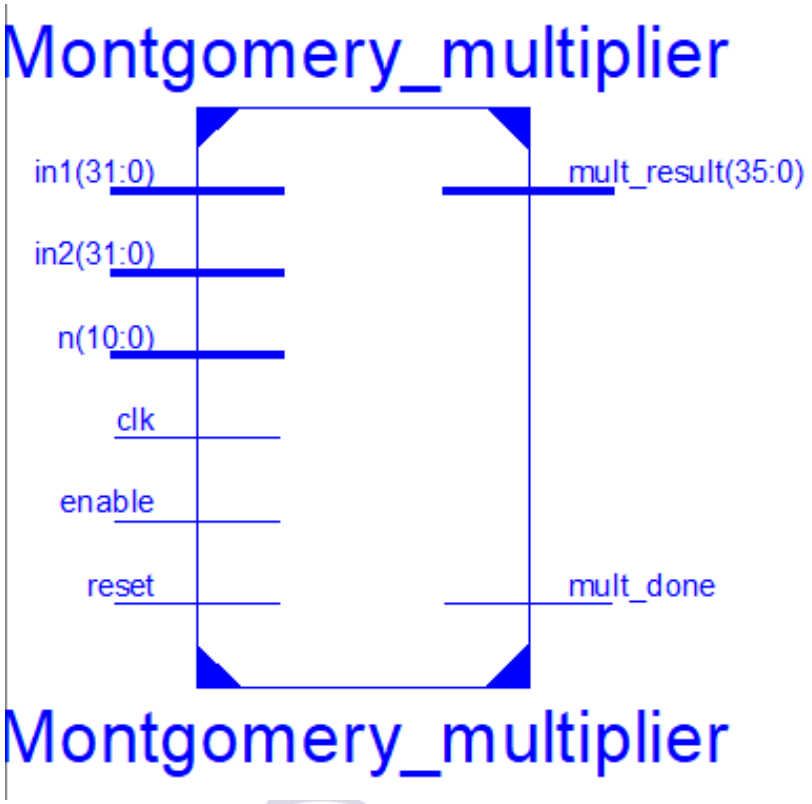

Figure 12. RTL View of 64 Bit Montgomery Multiplier

Figure 10. shows the output waveform of Montgomery multiplier.

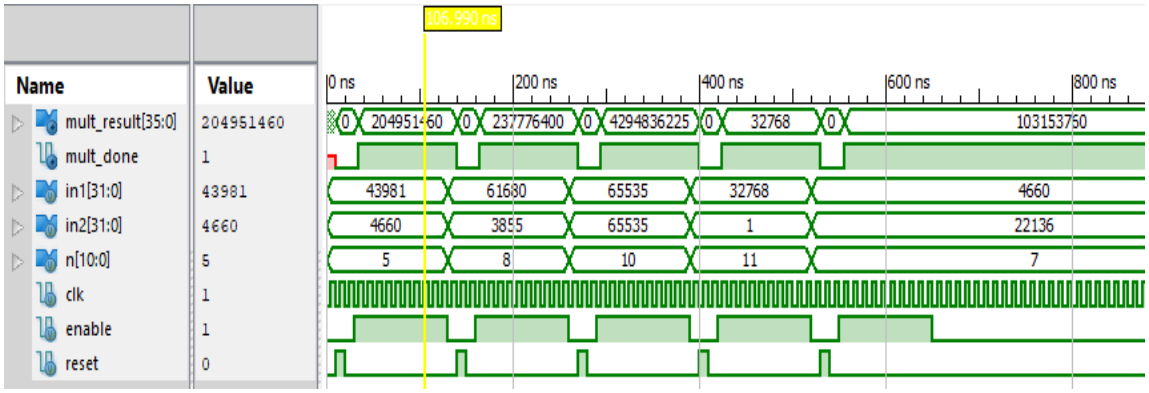

Figure 10. Output waveform of 64 Bit Montgomery Multiplier

Performance Analysis and Comparison

In this Table 4, three distinct multipliers Montgomery, Wallace Tree, and Booth Multipliers are given different metrics. These measures include execution time, peak memory utilization, the number of full adders and half adders, and the number of Look-Up Tables (LUTs) used as a percentage of the available LUTs in the design. Based on particular project needs, these metrics can assist determine which multiplier to

utilize by assessing the performance, resource usage, and efficiency of each multiplier in various circumstances.

Table 4. Comparison of Montgomery Multiplier, Wallace Multiplier and Booth Multiplier

	Montgomery Multiplier	Wallace Tree Multiplier	Booth Multiplier
Number of Full Adder	672	1697	N/A
Number of Half Adder	0	271	N/A
Peak Memory Usage	4610 MB	4620 MB	4602 MB
Time	10.594 ns	6.936 ns	8.677 ns
Number of LUTs	661 (2%)	1784 (6%)	169 (1%)

Conclusion:

The Wallace Tree Multiplier looks to have certain benefits over the Montgomery Multiplier and the Booth Multiplier, The Wallace Tree Multiplier is the one that is quickest among the three, with the lowest time (6.936 ns). This is frequently an important consideration in many applications, particularly high-performance computing. Due to its shorter propagation delay (6.936 ns), the Wallace Tree Multiplier surpasses the other two multipliers in terms of speed. As the number of full and half adders (1697 full adders and 271 half adders) was dramatically increased. adders) and LUTs (6%), it will cost in the consumption of more resources. The Wallace Tree Multiplier is the best choice in case you want the minimum propagation delay or you want to reduce the number of calculationstop priority. This decision, nevertheless, results in higher resource use.

REFERENCE:

[1]. Sulaiman, N., Obaid, Z., Marhaban, M. H., & Hamidon, M. N. (2009). Design and Implementation of FPGA-Based Systems - A Review. Australian Journal of Basic and Applied Sciences, 3.

[2]. Iana, G. V., Anghelescu, P., & Serban, G. (2011, September). RSA encryption algorithm implemented on FPGA. In 2011 International Conference On Applied Electronics (pp. 1-4). IEEE.

[3]. Wolf, W. (2004). FPGA-based System Design (Illustrated ed., pp. 24-25). Prentice Hall PTR. ISBN: 0131424610, 9780131424616.

[4]. Grout, I. (2008). Digital Systems Design with FPGAs and CPLDs (pp. 28-29). Newnes. https://doi.org/10.1016/B978-0-7506-8397-5.00011-8

[5]. Todman, T. J., Constantinides, G. A., Wilton, S. J. E., Mencer, O., Luk, W., & Cheung, P. Y. K. (2005). Reconfigurable computing: Architectures and design methods. IEE Proceedings - Computers and Digital Techniques, 152(2), 193-207.

[6]. Dhillon, H., & Mitra, A. (2008). A Reduced-Bit Multiplication Algorithm for Digital Arithmetic. International Journal of Computational and Mathematical Sciences, 2.

[7]. Chandel, D., Kumawat, G., Lahoty, P., Vart, V., & Sharma, S. (2013). Booth Multiplier: Ease of Multiplication.

[8]. Bajard, J.-C., Laurent-Stéphane, Didier, & Kornerup, Peter. (1998). An RNS Montgomery Modular Multiplication Algorithm. IEEE Transactions on Computers, 47(8), 766-776.

[9]. Surabhi, G. S., Bhavana, J., Madhu, S., & Arunarao, B. P. (2020). Design and Implementation of 256*256 Booth Multiplier and Its Applications. International Journal of Engineering Research & Technology (IJERT), 8(11).

[10]. Kureshi, A.K. (2016). Hardware Implementation of Configurable Booth Multiplier on FPGA. International Journal of VLSI System Design and Communication Systems, 4, 99-103.

[11]. Usha, S.M., & Mahesh, H.B. (2020). Minimization of power and area of digital modulator for cellular communication using Cadence in 180nm, 90nm, and 45nm CMOS technology. Indian Journal of Science and Technology, 13(25), 2537-2546.

[12]. Khan, S., Kakde, S., & Suryawanshi, Y. (2013, December). VLSI implementation of reduced complexity wallace multiplier using energy efficient CMOS full adder. In 2013 IEEE international conference on computational intelligence and computing research (pp. 1-4). IEEE.

[13]. Shalini K, & Balaji B.S. (2022). Efficient Wallace Tree Multiplier Using Parallel Prefix Adder. Journal of Emerging Technologies and Innovative Research (JETIR), 9(7), f206.

[14]. Aditya, M., Kumar, Y. N., & Vasantha, M. H. (2016, January). Reversible full/half adder with optimum power dissipation. In 2016 10th International Conference on Intelligent Systems and Control (ISCO) (pp. 1-4). IEEE.

[15]. Pai, Y. T., & Chen, Y. K. (2004, January). The fastest carry lookahead adder. In Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications (pp. 434-436). IEEE.

[16]. Narmadha, G., & Balasubadra, K. (2016). An optimized montgomery multiplier for public key cryptography. Int J Adv Engg Tech/Vol. VII/Issue I/Jan.-March, 5, 07.