Implementation of High speed and Area efficient Karatsuba Multiplier Using Sklansky adder

Sai Rama Rao T[1], Tulasi Ratnam K[2], Navya Sri K[3], Bhagya Sree E[4], Nalini Devi G[5], Nagaraju P[6]

[12345]Graduate Student,[6]Assistant Professor, Department of Electronics and Communication Engineering, Sri Vasavi Engineering College, Tadepalligudem, West Godavari, Andhra Pradesh, India.

---***---

Abstract -*Multiplication serves as the center of many computer processes, hence the creation of efficient multiplication algorithms is a crucial area of research. The Karatsuba method is widely recognized for its ability to speed multiplication by dividing big operands into smaller, simpler to handle sub problems. However, delivering high-speed performance while preserving area is still a difficulty in real applications. This paper proposes a novel approach to enhance both speed and area efficiency of Karatsuba multiplication through the integration of Sklansky adders. Sklansky adders are a type of parallel prefix adder, known for their inherent parallelism and compact layout, offer a promising solution to address the performance and resource utilization concerns*

associated with traditional multiplication architectures. The proposed design leverages the recursive nature of Karatsuba multiplication to exploit parallelism at various stages of the algorithm. By incorporating Sklansky adders into critical sections of the Karatsuba multiplier, the proposed architecture achieves significant improvements in both speed and area efficiency compared to Vedic multiplier implementation. Furthermore, the design exhibits scalability across different operand sizes, making it suitable for a wide range of applications demanding high-performance multiplication capabilities.

Key Words: Karatsuba Multiplier, Sklansky Adder, Vedic multiplier.

1. INTRODUCTION

In the realm of Very Large-Scale Integration (VLSI) design, optimizing arithmetic operations for floating-point numbers is crucial for achieving high-performance computing systems. Floating-point multiplication, a fundamental arithmetic operation in many computational tasks, demands efficient algorithms and hardware implementations to meet the increasing demands of modern applications.

The realm of floating-point multiplication underlies a wide range of scientific calculations, from simulating complicated phenomena in nature to processing enormous datasets in machine learning. However, the typical, elementary-school approach of multiplying big floating-point numbers rapidly becomes computationally costly. This is where the brilliant Karatsuba algorithm shines.

Traditional methods for floating-point multiplication often face challenges in terms of speed and area efficiency. One promising approach to address these challenges is leveraging the Karatsuba algorithm, a fast multiplication technique known for its divide-and-conquer strategy, which reduces massive number multiplication to smaller, more manageable subproblems. It accomplishes this by intelligently breaking down the two floating-point values (A and B) into high and low-order bits. This decomposition facilitates recursive calls to the Karatsuba algorithm on the smaller parts. The real magic, is its ability to estimate the middle product (P2) - the product of A and B's middle-order bits - using just high and low-order data. This estimation reduces the requirement for a full multiplication, resulting in a considerable reduction in the number of multiplication steps necessary.

The benefits of the Karatsuba algorithm are numerous. Firstly, it has a time complexity of around O(n^1.58), which is significantly better than the O(n^2) difficulty of the grade-school technique, especially for large numbers. This leads to considerably quicker computations, which is beneficial for applications that rely heavily on floating-point calculations. Second, its divide-and-conquer nature makes it ideal for hardware implementation in processors and other computing devices. This enables effective use of hardware resources for floating-point multiplication applications.

This paper explores the design and implementation of floating-point multiplication using the Karatsuba multiplier in VLSI. We begin by providing a brief overview of floating-point representation and the challenges associated with its multiplication. Subsequently, we delve into the principles of the Karatsuba algorithm and its adaptation for floating-point arithmetic.

2. Literature Survey

Haripriya A et al. present a special Vedic multiplier architecture that utilizes the fundamental components of digital signal processors, the RCA and CSLA architectures. The study encourages CSLA integration, emphasizing power, speed, and area efficiency in order to increase multiplication. The Urdhva-Tiryakbhyam approach is used to divide the input operands into smaller blocks. The intermediate products are then obtained, and the CSLA is used to sum the results to achieve the final result. Despite problems with area efficiency brought on by CSLA's twin RCA design, synthesis experiments using Xilinx ISE 14.7 demonstrate a 12% reduction in latency and an increase in area of 3% when compared to the RCA-based Vedic multiplier[1].

Sumit Vaidya et al. explore the computational burden of multiplication operations in processors, aiming to enhance performance and minimize power consumption. Leveraging insights from Ancient Indian Vedic Mathematics, specifically the "Urdhva Tiryakbhyam" algorithm, they seek to improve multipliers speed and area efficiency. Additionally, they introduce the "Nikhilam Sutra" formula to expedite multiplication, thereby enhancing overall multiplier performance[2].

K.N. Vijeyakumar et al. introduce a novel approach to enhance the speed and area efficiency of Arithmetic Units (AUs) crucial for signal and image processing architectures. By integrating the "vertical and crosswise sutra" from Vedic mathematics, the research focuses on improving multipliers. This method effectively reduces critical delay by simplifying Partial Product (PP) generating units. Implementation results demonstrate significant improvements, with a 19.2% decrease in the Area-latency Product (ADP) and a 13.7% reduction in latency, particularly utilizing the TSMC 180 nm CMOS process with CADENCE Encounter Digital Implementation[3].

S. Alwyn Rajiv et al. stress the importance of floating-point multipliers in DSP, delving into IEEE 754 standard representations and advocating the Vedic method for 24-bit multiplication. They scrutinize adder designs—9-bit to 36-bit variants—including Carry Look Ahead, Carry Save, Ripple Carry, and Kogge Stone types, aiming to enhance VLSI efficiency for DSP. The research seeks to optimize adder performance for DSP via Verilog HDL implementation and Xilinx ISE simulation, focusing on area and latency evaluations[4].

Ranjeeta Yadav et al. conduct a comparative analysis of various combinations of Finite Impulse Response (FIR) filters, multipliers, and adders sourced from literature. Emphasizing the significance of FIR filters in digital signal processing applications, they assess different types of adders and multipliers, including Vedic Multipliers, Han Carlson, Dadda, Booth, Ripple Carry, and Carry Look Ahead. Leveraging insights from Vedic mathematics, they evaluate key criteria such as area, latency, power consumption, and Look-Up Table (LUT) utilization to determine effective FIR Filter designs. Notably, the Vedic Multiplier emerges as a high-speed and low-power option among the compared multipliers[5].

S. Jayakumar et al. investigate signal processing applications, with a focus on the crucial role of Finite Impulse Response (FIR) filters in enhancing signal quality. They advocate for the adoption of the Anurupy multiplier to address critical path challenges, leading to enhanced efficiency and speed in FIR filter designs, as demonstrated through comprehensive simulation and synthesis using VHDL, Model Sim, and Xilinx tools[6].

Arish S and R. K. Sharma underscore the pivotal role of binary multiplication in floating-point multiplier designs and high-power computing, emphasizing its impact on time, area, and power consumption. To optimize implementation efficiency, they propose a novel technique for unsigned binary number multiplication, blending the Urdhva-Triyakbhyam algorithm from Vedic Mathematics for lower bit operations with the Karatsuba algorithm for higher bit operations. Implementing this approach in Verilog HDL, they develop an unsigned binary multiplier tailored for Spartan-3E and Virtex-4 FPGA devices[7].

Y. Srinivasa Rao et al. highlight the widespread use of floating-point arithmetic, especially in scientific and digital signal processing applications requiring double-precision arithmetic. They emphasize the complexity of multiplication, proposing the Urdhva Tiryagbhyam method to optimize mantissa multiplication for improved speed and area efficiency. Implementation of the double-precision floating multiplier is done using Verilog HDL and Xilinx ISE tools, targeting Virtex-5 FPGA platforms[8].

Anand Mehta et al. focus on efficient floating-point multiplication for resource-intensive digital signal processing and media applications. They implement the Karatsuba algorithm in Verilog HDL, adhering to the IEEE 754 standard, to enhance power and time efficiency. The multiplier employs a three-stage pipelining mechanism with an 8 clock cycle delay, and mantissa bits are divided into three halves to utilize common multipliers in the FPGA Cyclone II device family[9].

Arish S et al. acknowledge the significant time and power consumption associated with efficient floating-point multiplication, particularly in high-power computing applications. They propose a unique method for implementing an unsigned binary multiplier for mantissa multiplication in IEEE 754 floating-point format by combining the Urdhva-Triyakbhyam and Karatsuba algorithms from Vedic mathematics.

The implementation, conducted in Verilog HDL, targets FPGA platforms such as Virtex-4 and Spartan-3E. The study aims to enhance the latency and power efficiency of floating-point multiplication operations, crucial for signal processing, image processing, and other computing tasks[10].

Ravi Kishore Kodali et al. underscore the computational challenges posed by IEEE-754 standards, emphasizing the importance of floating-point arithmetic, particularly multiplication, in scientific and signal processing domains. They address the constraints imposed by large mantissa multiplications on space and performance. Through implementation on a standardized reconfigurable FPGA architecture, the study evaluates the performance of two widely used multiplication algorithms—Booth and Karatsuba (Normal and Recursive). The recursive Karatsuba method emerges as the most efficient solution, offering insights into effective floating-point multiplication techniques through comparisons of resource consumption and execution speed[11].

Rahul Rathod et al. focus on the computational demands of floating-point multiplication in signal processing and multimedia applications. Acknowledging its resource-intensive nature, the study evaluates complex floating-point multiplication performance using Vedic, array, and CIFM multipliers. Through Verilog implementation using VIVADO DESIGN SUITE 2018.3, the research aims to optimize floating-point multiplication for efficient signal processing and multimedia computations by comparing speed and resource consumption[12].

Vijeta Iyer et al. introduced a Vedic Mathematics-based method for efficient multiplication, particularly suited for binary strings of one, with potential application to decimal values. This algorithm's effectiveness lies in its adeptness at processing decimal numbers within specific formats, enhancing computational efficiency and offering versatility in solving arithmetic problems[13].

N. Mohana Priya et.al The study presents a brand-new strategy for improving FIR digital filters' effectiveness in DSP applications. For high-speed and low-power multiplication, it suggests a signed vedic multiplier that makes use of parallel operations and the Urdhva Triyakbhyam sutra. The suggested method exhibits a notable 52% reduction in power usage and a 15% speed boost over traditional radix-4 Booth multipliers. The crucial demand for increased performance and energy economy in signal data processing is addressed by this breakthrough. It provides flexibility for a range of DSP applications by combining signed and unsigned multiplications. The importance of designing multiplier topologies for improved performance in digital signal processing is emphasized in the study. This work provides important new understandings for improving FIR filter architectures for practical uses[14].

3. Fixed Point Multiplication

a. Vedic multiplier

The Vedic multiplier is a mathematical method that is based on ancient Indian mathematical principles from the Vedas. It is an efficient and quick technique used for multiplying two numbers. The Vedic multiplier is especially beneficial when dealing with large numbers as it simplifies the multiplication process by reducing the number of steps required.

The 64-bit Vedic multiplier technique can be utilized to multiply two 64-bit numbers. By leveraging the inherent properties of the Vedic mathematics system, the 64-bit Vedic multiplier performs the multiplication in a streamlined manner. It achieves this by breaking down the multiplication process into smaller steps and employing various sutras or formulas from Vedic mathematics. Consequently, the 64-bit Vedic multiplier significantly decreases the number of computations needed.

The above Fig-1 illustrates the block diagram of a 64-bit Vedic multiplier. This innovative and efficient multiplier is designed to perform high-speed multiplication operations on 64-bit numbers. The block diagram showcases the various components and stages involved in the multiplication process. These components work together seamlessly to ensure accurate and fast multiplication results. The Vedic multiplier is known for its ability to perform complex multiplication operations with reduced latency and power consumption. With its advanced architecture and optimized design, the 64-bit Vedic multiplier is a valuable tool for applications that require high-performance multiplication capabilities.

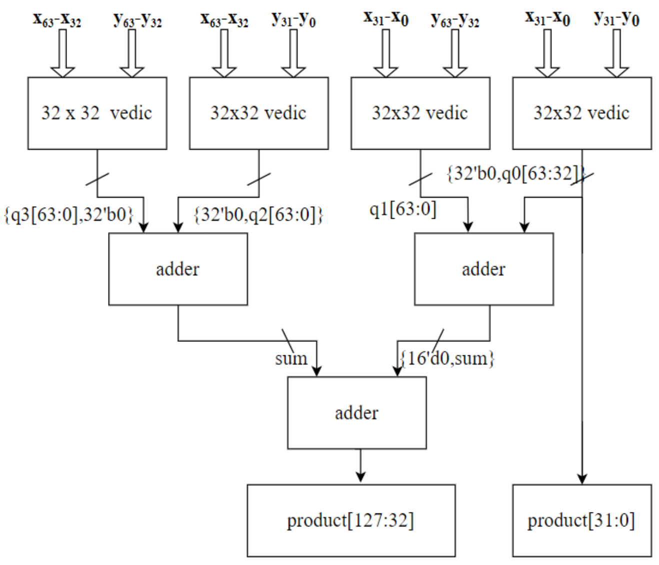


Fig- 1: Vedic multiplier

The use of a 64-bit Vedic multiplier can have practical applications in various fields, such as computer science and engineering. The use of a 64-bit Vedic multiplier can help improve the efficiency and speed of computations. Additionally, in signal processing and digital signal processing applications, where high-speed multiplication is required, the 64-bit Vedic multiplier can offer significant performance benefits.

b. Karatsuba multiplier

The Karatsuba multiplier is an algorithm used for multiplying large numbers efficiently. Specifically, it is designed to perform multiplication on numbers with a large number of digits, such as 64-bit numbers. The algorithm was developed by Anatolii Alexeevitch Karatsuba in 1960 and has since become widely used in computer science and cryptography.

The main advantage of the Karatsuba multiplier is its ability to reduce the number of multiplications required to perform the multiplication operation. Traditionally, the multiplication of two n-digit numbers would require n^2 individual multiplications. However, the Karatsuba algorithm reduces this number to roughly n^(log3/log2), resulting in a significant improvement in efficiency.

The key idea behind the Karatsuba algorithm is to break down the original multiplication problem into smaller sub-problems that can be solved recursively. Instead of performing individual multiplications for each digit pair, the algorithm splits the numbers into two halves and performs three separate multiplications: one for the high-order digits, one for the low-order digits, and one for the cross product.

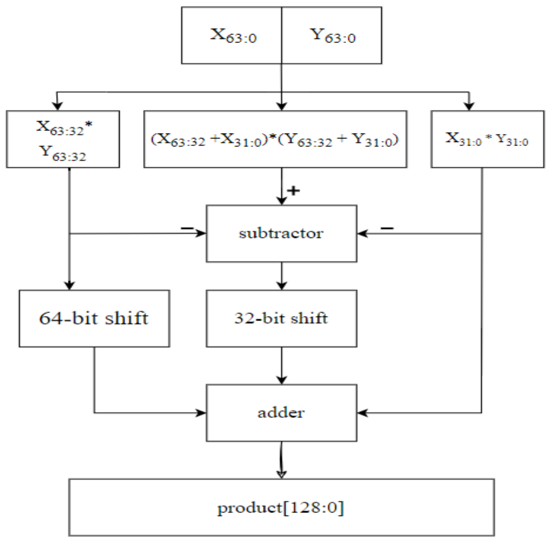


Fig- 2: Karatsuba multiplier 64-Bit

The figure labeled as Fig. 2, represents a 64-bit Karatsuba multiplier. The Karatsuba algorithm is a method used for fast multiplication of large numbers. This particular implementation is designed to handle 64-bit numbers efficiently. The figure provides a visual representation of the circuitry and components involved in the Karatsuba multiplier. By utilizing this algorithm, multiplication operations can be performed more quickly and efficiently, making it an ideal choice for applications that involve large numbers and require high-performance computing.

P1 = Xh * Yh (1)

P2 = Xl * Yl (2)

P3 = (Xh + Xl) * (Yh + Yl) (3)

final product

XY = P1 * 2^64 + (P3 - P1 - P2) * 2^32 + P2 (4)

Multiplication of two 64-bit numbers can perform by (1), (2), (3) and (4) as shown in above.

4. Double precision floating Point Multiplication

Double precision floating-point in the IEEE 754 standard refers to the format for representing floating-point numbers using 64 bits. This format is commonly used for representing real numbers in computer systems. The following three fields are present in format of double precision floating point: Sign bit (1 bit): It specifies the sign of the number. 0 means positive and 1 means negative.

Exponent (11 bits): This field represents the exponent of the number in a biased form. Bias is added to the actual exponent to allow both positive and negative exponents to be represented. For double precision, the bias is 1023, so an exponent of 0 in this fields represents -1023, and an exponent of 2047 represents +1024.Mantissa (52 bits): This field stores the significant digits of the number in binary. The leading bit is always 1 (except for denormalized numbers), and the remaining bits represent the mantissa part. The Fig 3 Shows the field format of double precision floating point

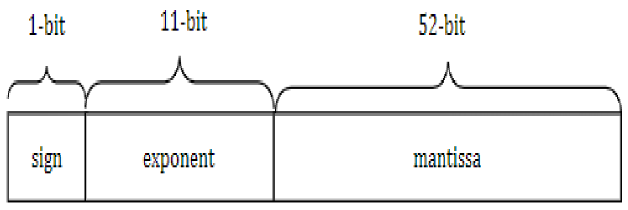


Fig- 3: Format of Double precision floating point number in IEEE 754 standard representation

Multiplication of two double precision floating-point numbers involves the following steps:

Sign Bit Multiplication: Multiply the sign bits of the two numbers to determine the sign of the result, simply performing the ex-or of two sign bits.

Exponent Addition: Add the exponents of the two numbers together. Since the exponents are biased (by 1023 in double precision), subtract 1023 from the sum to get the unbiased result.

Significand Multiplication: Multiply the significands of the two numbers together. This is a multiplication of two 53-bit numbers (including the implied leading 1 bit).

Normalization: Normalize the result of the significand multiplication by shifting the result and adjusting the exponent accordingly. Normalization involves shifting the binary point to the right until there is only one non-zero digit to the left of the binary point. Adjust the exponent accordingly to compensate for the shifts.

Rounding: Round the result according to the rounding mode specified in the IEEE 754 standard. The result may need to be rounded up, down, or to the nearest representable value.

Overflow and Underflow: Check for overflow (result too large to be represented) or underflow (result too small to be represented). If an overflow or underflow occurs, adjust the result and exponent accordingly.

a. Double precision Vedic multiplier

A floating-point Vedic multiplier is a hardware implementation that performs multiplication on floating-point numbers using the Vedic mathematics technique. This technique, derived from ancient Indian mathematics, offers a fast and efficient way of performing complex calculations. In the context of a 64-bit floating-point multiplier, this means that the hardware is designed to handle numbers with a precision of 64 bits, including both the integer and fractional parts. This type of multiplier is commonly used in applications that require high precision and accuracy, such as scientific simulations, financial modeling, and data analysis. By leveraging the Vedic mathematics principles, the floating-point Vedic multiplier can deliver faster and more accurate results compared to traditional multiplication algorithms.

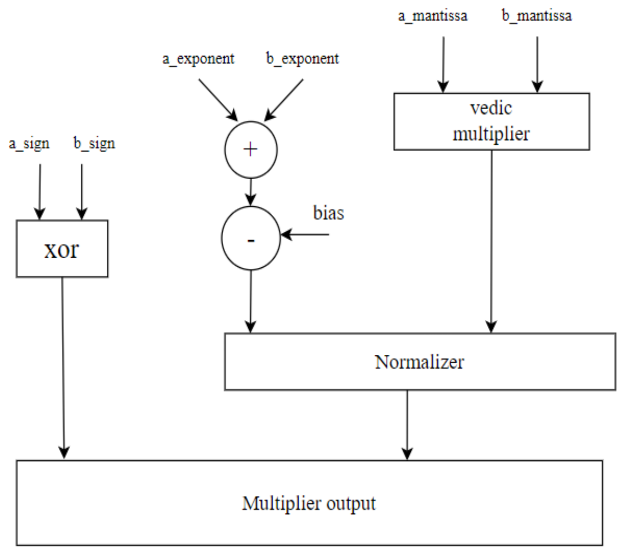


Fig- 4: Double precision floating point multiplier using Vedic multiplier

b. Double precision karatsuba multiplier

In a traditional floating-point multiplication algorithm, the two floating-point numbers are multiplied using the standard multiplication rules. This involves multiplying the significands (mantissas) of the numbers and adding their exponents. However, this approach can be computationally expensive, especially for larger floating-point numbers.

The floating-point Karatsuba multiplier addresses this issue by breaking down the multiplication process into smaller sub-problems. It recursively divides the original floating-point numbers into smaller parts and performs multiplications on these parts. These intermediate results are then combined to obtain the final product as shown in Figure 5.

The key idea behind the Karatsuba multiplier is to exploit the properties of polynomial multiplication. Floating-point numbers can be represented as polynomials, where the coefficients correspond to the bits of the significand and the exponents represent the powers of 2. By using polynomial multiplication techniques, the Karatsuba algorithm can efficiently compute the product of two floating-point numbers.

One of the main advantages of using the floating-point Karatsuba multiplier is its ability to reduce the number of multiplication operations required. This is achieved by breaking down the original problem into smaller sub-problems and reusing intermediate results. As a result, the overall efficiency of the multiplication process is improved.

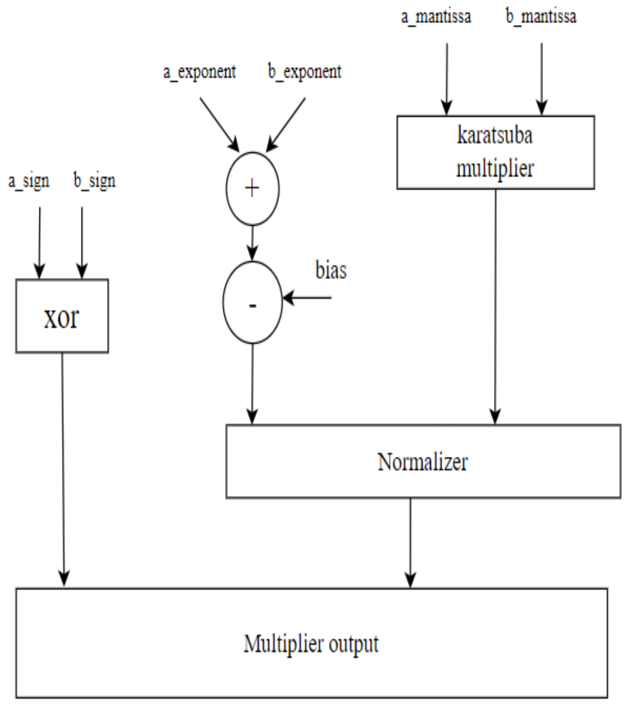


Fig-5: Double precision floating point multiplier multiplier using karatsuba multiplier

6. Results

The designed and existing multiplier is synthesized in cadence genus synthesizer tool and the simulations results are obtained from the vivado tool.

The schematic and simulation results of 64-bit Vedic multiplier are shown in Fig-6 and Fig-7 simultaneously

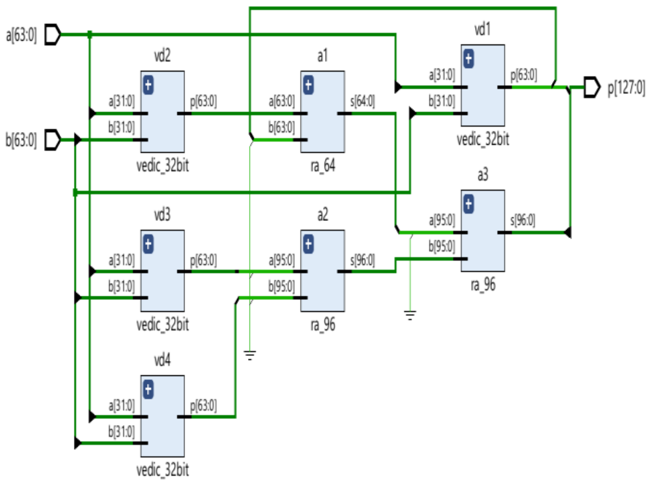
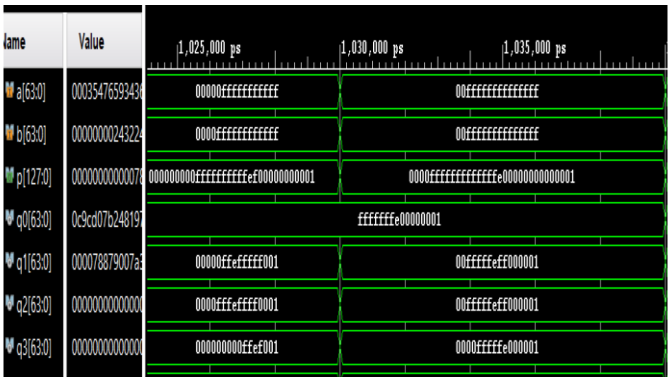


Fig -6: Schematic of 64-bit Vedic multiplier



```
Name     Value              1,025,000 ps                      1,030,000 ps                      1,035,000 ps
a[63:0]  00035476593436     00000ffffffffff                   00ffffffffffffff
b[63:0]  0000000243224      0000ffffffffffff                  00ffffffffffffff
p[127:0] 00000000000078     000000000ffffffffffef00000000001  0000ffffffffffffffe0000000000001
q0[63:0] 0c9cd07b248197                     fffffffe00000001
q1[63:0] 00078879007a3      00000ffeffffff001                 00fffffeff000001
q2[63:0] 0000000000000      0000fffeffff0001                  00fffffeff000001
q3[63:0] 0000000000000      000000000ffef001                  0000fffffe000001
```

Fig -7 Simulation results of 64-bit vedic multiplier

The schematic and simulation result of 64-bit multiplier is shown in Fig-8 and Fig-9 respectively.

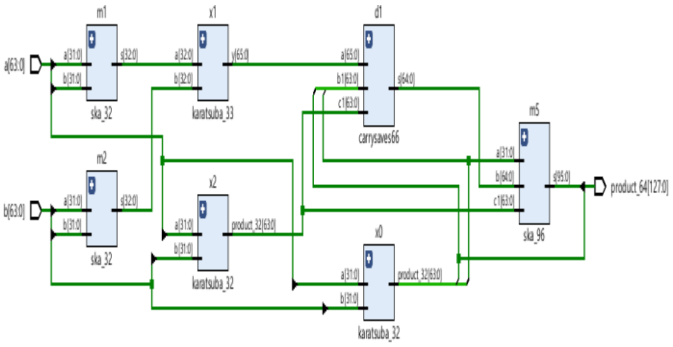
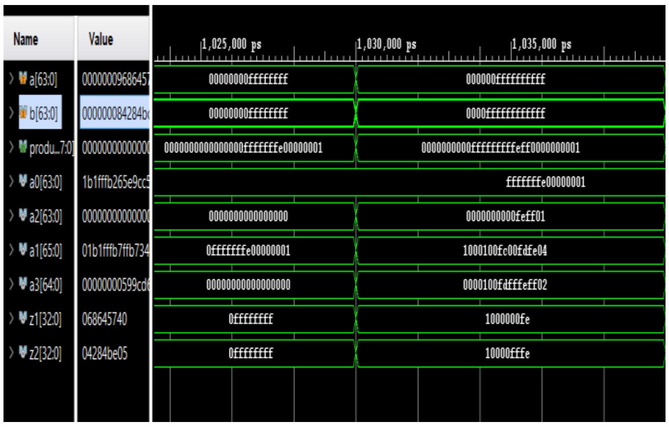


Fig-8: Schematic results of 64-bit karatsuba multiplier



```
Name         Value            1,025,000 ps                      1,030,000 ps                      1,035,000 ps
a[63:0]      0000000968645    00000000ffffffff                  000000ffffffffff
b[63:0]      000000084284bc   00000000ffffffff                  0000ffffffffffff
produ...7:0] 00000000000000   0000000000000000fffffffe00000001  0000000000fffffffffeff0000000001
a0[63:0]     1b1fffb265e9cc5                                    fffffffe00000001
a2[63:0]     00000000000000   0000000000000000                  0000000000feff01
a1[65:0]     01b1fffb7ffb734  0fffffffe00000001                 1000100fc00fdfe04
a3[64:0]     00000000599cd6   00000000000000000                 0000100fdfffeff02
z1[32:0]     068645740        0ffffffff                         1000000fe
z2[32:0]     04284be05        0ffffffff                         10000fffe
```

Fig -9: Simulation results of 64-bit karatsuba multiplier

The table -1 presents the performance parameters of 64bit multiplication.

Table-1: Performance parameters of 64-bit multipliers

Multiplier	Area(um^2)	Delay (ps)	Power(mw)
vedic	145917	31892	0.11
karatsuba	114586	28555	0.23

The schematic and simulation results of double precision floating multiplier using Vedic multiplier are shown in Fig-10 and Fig-11 respectively

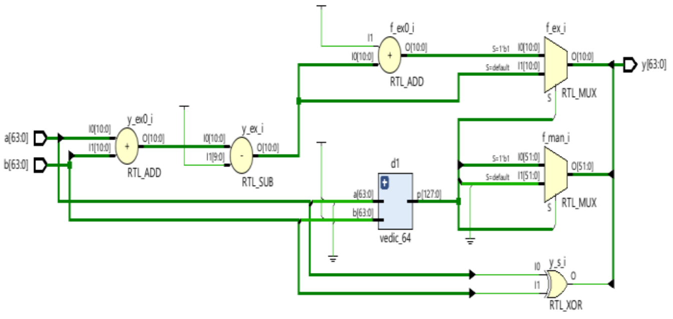
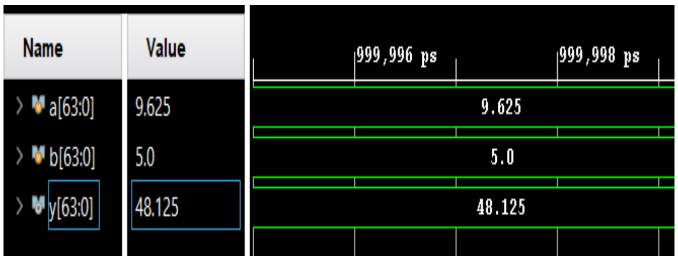


Fig -10: Schematic of Double precision floating point

Multiplier using Vedic multiplier



```
Name     Value    999,996 ps        999,998 ps
a[63:0]  9.625              9.625
b[63:0]  5.0                5.0
y[63:0]  48.125             48.125
```

Fig-11: Simulation of Double precision floating point multiplier using Vedic multiplier

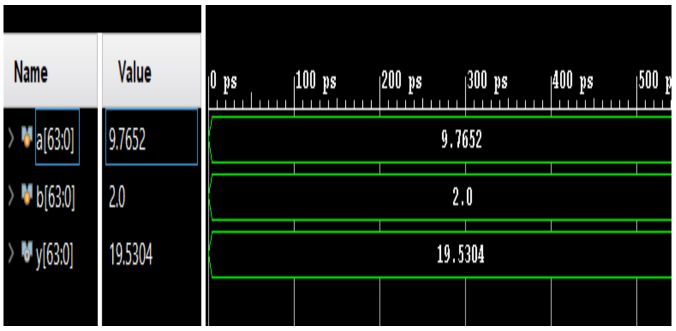


Fig-12: Simulation of Double precision floating point multiplication using karatsuba multiplier

The schematic and simulation results of double precision floating point multiplier using karatsuba multiplier are shown in Fig-12 and Fig-13 respectively.

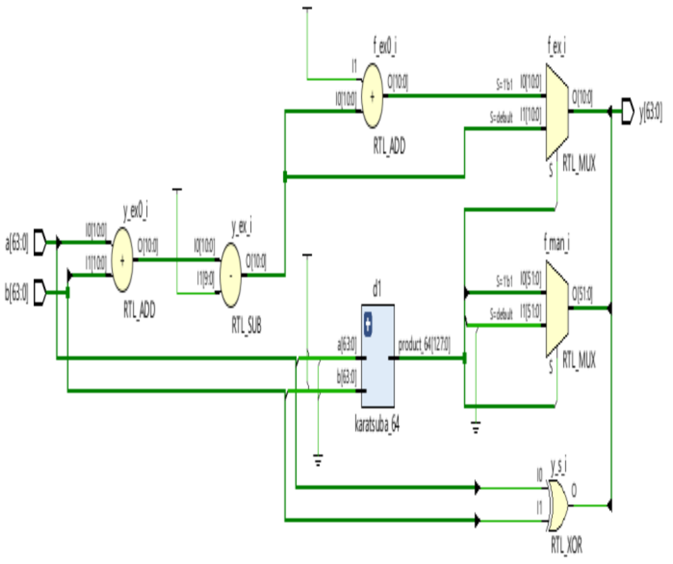


Fig 13: Schematic of Double precision floating point multiplication using karatsuba multiplier

The table-2 gives the performance parameters of floating-point multiplication (FPM)

Table-2: Performance parameters of floating-point multiplier

Multiplier	Area(um^2)	Delay(ps)	Power(mw)
FPM using vedic	98281	30747	8
FPM using karatsuba	91869	30165	18

7. CONCLUSION

The proposed karatsuba multiplier rather than other multipliers has advantage to reduce the area and delay. The area is calculated in terms of micro meters^2(um^2). Area is reduced in floatingpoint karatsuba multiplier compared to floating point vedic multiplier and also reduced delay in floating point karatsuba multiplier compared to floating point Vedic multiplier. Finally the area and delay are reduced for karatsuba multiplier than the vedic multiplier.

ACKNOWLEDGEMENT

This work carried out by Research and Development center, department of ECE, Sri Vasavi Engineering College(A), Tadepalligudem, West Godavari, Andhra Pradesh and India.

REFERENCES

[1] Haripriya A, "Design and Analysis of 16-bit Vedic Multiplier using RCA and CSLA", 2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)

[2] R. Sridevi, "Design of a High Speed Multiplier (Ancient Vedic Mathematics Approach) International Journal of Engineering Research (ISSN : 2319-6890) Volume No.2, Issue No.3, pp : 183-186

[3] K.N. Vijeyakumar, " Vlsi Implementation Of High Speed Area Efficient Arithmetic Unit Using Vedic Mathematics", Ictact Journal On Microelectronics, April 2016, Volume: 02, Issue: 01

[4] S. Alwyn Rajiv, " Comparative Study on Performance of various Adders used in Vedic Multiplier", © 2019 JETIR May 2019, Volume 6, Issue 5

[5] R. Yadav, "Review on FIR Filters Using Different Adders and Multipliers Based on Vedic Mathematics," 2021 International Conference on Simulation, Automation & Smart Manufacturing (SASM), Mathura, India, 2021, pp. 1-5

[6] Jayakumar, "High-Performance FIR Filter Implementation Using Anurupye Vedic Multiplier", Circuits and Systems, 7, 3723-3733 Circuits and Systems , Vol.7 No.11, September 2016.

[7] Arish S, "An efficient binary multiplier design for high speed applications using Karatsuba algorithm and UrdhvaTiryagbhyam algorithm", Proceedings of 2015 Global Conference on Communication Technologies (GCCT 2015)

[8] Y. Srinivasa Rao, M. Kamaraju," An FPGA Implementation of High Speed and Area Efficient Double-Precision

Floating Point Multiplier Using Urdhva Tiryagbhyam Technique", 2015 Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG) December11-12, 2015, Kurnool, Andhra Pradesh, India

[9] Anand Mehta," Implementation of Single Precision Floating Point Multiplier using Karatsuba Algorithm", 978-1-4673-6126-2/13/$31.00 © 2013 IEEE

[10] Arish S," An efficient floating point multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm", 978-1-4799-6761-2/15/$31.00 ©2015 IEEE

[11] Ravi Kishore Kodali," FPGA Implementation of IEEE-754 Floating Point Karatsuba Multiplier", 978-1-4799-4190-2/14/$31.00 ©2014 IEEE

[12] R. Rathod, P. Ramesh, P. S. Zele and A. K. Y, "Implementation of 32-Bit Complex Floating-Point Multiplier Using Vedic Multiplier, Array Multiplier and Combined integer and floating point Multiplier (CIFM)," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bangluru, India, 2020, pp. 1-5,

[13] Vijeta Iyer, "Generalised Algorithm for Multiplying Binary Numbers Via Vedic Mathematics", 021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)

[14] N. Mohana Priya," High performance fr flter based on vedic mathematics", Int J Syst Assur Eng Manag (June 2023) 14(3):829–835

BIOGRAPHIES

	Mr. Sai Rama Rao T Final Year B. Tech student in the Electronics & Communication Engineering Department, Sri Vasavi Engineering College, Tadepalligudem, West Godavari, Andhra Pradesh, India.
	Ms. Tulasi Ratnam K Final Year B. Tech student in the Electronics & Communication Engineering Department, Sri Vasavi Engineering College Tadepalligudem, West Godavari, Andhra Pradesh, India.
	Ms. Navya Sri L Final Year B. Tech student in the Electronics & Communication Engineering Department Sri Vasavi Engineering College Tadepalligudem, West Godavari, Andhra Pradesh, India.
	Ms. Bhagya Sree E Final Year B. Tech student in the Electronics & Communication Engineering Department, Sri Vasavi Engineering College Tadepalligudem, West Godavari, Andhra Pradesh, India.
	Ms. Nalini Devi G Final Year B. Tech student in the Electronics & Communication Engineering Department, Sri Vasavi Engineering College Tadepalligudem, West Godavari, Andhra Pradesh, India.
	Mr. Nagaraju P Completed his M. Tech in 2013 from JNTUK, Hyderabad. He has totally 11 Years of teaching experience. Presently he is working as Assistant Professor in the Department, Sri Vasavi Engineering College, Tadepalligudem, West Godavari, Andhra Pradesh, India. His interested Research area is Microwave and mm wave filters.