JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Comparative Analysis of 32-Bit Complex Floating Point Multiplier

[1] Karthik Bandla, [2]Dhana Sri Darla, [3]Divya Teja Reddy Chekkara

[1 2 3]Student
[1]Department of Electronics and Communication Engineering,
[1]R.V.R. & J.C. College of Engineering, Guntur, India

Abstract : Many signal processing and communication applications rely on complex floating point numbers as their main component. Hence, optimizing algorithms and hardware for complex number multiplication is a key focus to achieve faster processing and lower power consumption in these domains. Since multiplication speed directly impacts a system's overall performance. This study implements complex floating point multiplication utilizing array multiplier, Vedic multiplier, and CIFM multiplier using carry look ahead adder in Verilog on VIVADO DESIGN SUITE. The performance of these three multipliers mainly varies in parameters like delay, and power consumption. Based on the obtained outcomes, the best multiplier can be used for fast multiplication operations.

IndexTerms – Vedic Multiplier, Urdhva Tiryagbhyam, Array Multiplier, Combined Integer Floating Point Multiplier

I.INTRODUCTION

The ever-growing demand for real-time signal processing and image processing applications necessitates efficient hardware implementations for complex number multiplication. Complex floating-point multiplication, a cornerstone of these domains, poses a significant challenge due to its computational complexity. This paper delves into a comparative analysis of three prominent techniques for realizing a 32-bit complex floating-point multiplier: Vedic multiplier, array multiplier, and Combined Integer and Floating-point Multiplier (CIFM).

1. Vedic Multiplier: Exploits ancient Vedic mathematical algorithms for potentially faster and more area-efficient designs.
2. Array Multiplier: Employs a straightforward approach using an array of AND gates and adders, offering a basic yet effective implementation.
3. Combined Integer and Floating-point Multiplier (CIFM): Leverages a hybrid approach to potentially achieve a balance between speed and resource utilization.

The following is how this paper is organized:
In Section II, complex floating point multiplication [1] is explained. A sophisticated multiplier explanation is given in Section III. The implementation of a 24-bit CIFM multiplier is provided in Section IV. The implementation of array multipliers is explained in Section V. The 24x24 bit Vedic multiplier module is explained in Section VI. The floating point addition/subtraction architecture is explained in Section VII. Results of the implementation are given in section VIII. The paper is concluded in Section IX..

II.COMPLEX FLOATING POINT MULTIPLIER

In [1], the topic of complex floating point multiplier is covered.An adder/subtractor and 32-bit multipliers make up the architecture. There are four inputs total—two real and two imaginary components of two complex numbers. Following 32-bit multiplication, the integers are fed into a floating point adder or subtractor.

1. Represent mathematical entities with both a real and an imaginary component.
2. Represented as a + bi, where 'a' is the real part, 'b' is the imaginary part (unit multiplier of the imaginary unit 'i', where i^2 = -1).
3. Multiply the real parts of the complex numbers together,Multiply the imaginary parts of the complex numbers together. Add the product of the real parts to the product of the imaginary parts, considering their signs.

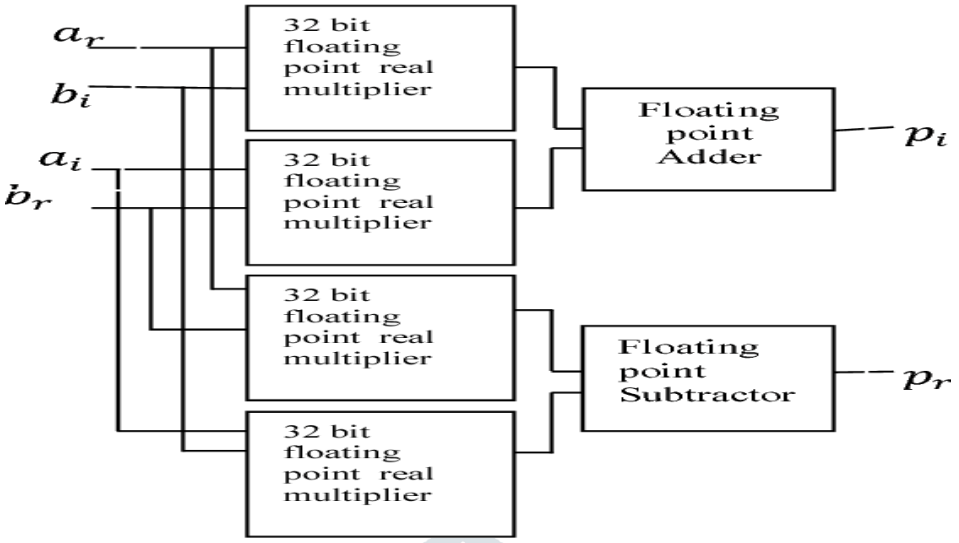

Fig.1. Complex Floating Point Architecture

III.FLOATING POINT MULTIPLIER

1. Sign bit: The sign bit in the 32-Bit format is the MSB, or the 31st bit, and it is set to 0 for positive values and 1 for negative ones.
2. Exponent portion: The exponent portion in the 32-bit format is the 8 bits that come after the sign bit, or [30:23] bits. The exponent is in the integer form of 8 bits, ranging from -127 to 128 and is recognized as the biased form.
3. Mantissa Part: If an exponent is not stored with all zeros, the Mantissa is approximately 23 bits long and has a leading bit with 1 at the MSB. Although a mantissa of 23 bits is visible, the total precision is 24 due to the concatenation of 1 bit and the use of various multiplier algorithms for multiplication.

As a result, three distinct multiplier algorithms and in addition to CIFM multiplier with carry look ahead adder are used for performance analysis, which also aids in cutting down on processing time and power usage.

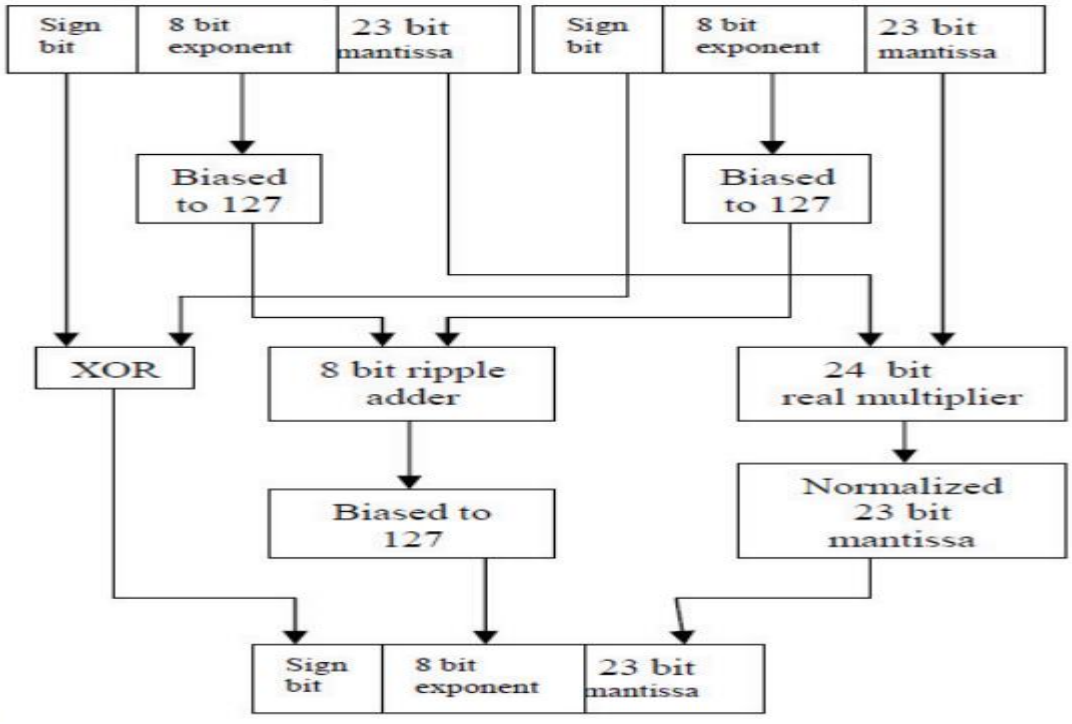

Fig.2. Floating Point Multiplier Architecture

IV.CIFM MULTIPLIER

In [4], the CIFM multiplier is covered. Four 12-bit parallel-operating multiplication modules make up the 24-bit multiplication block. For 12x12 multiplication, the four 12 bits are AH, AL, BH, and BL, correspondingly. Four-bit optimized multipliers are used to further divide the twelve-bit multiplication modules. The 4x4 multiplier blocks are used to categorize the entire 24x24 bit multiplication. Checkers are used by a 24-bit multiplication module as control signals.

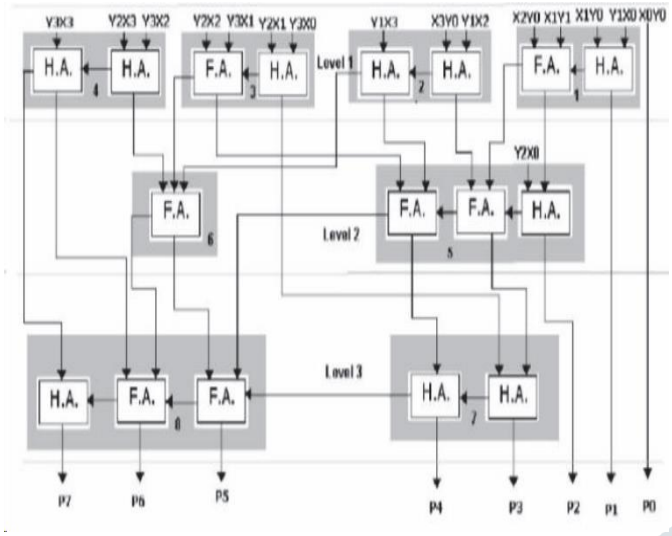

Fig.3. 4X4 CIFM Multiplier Architecture

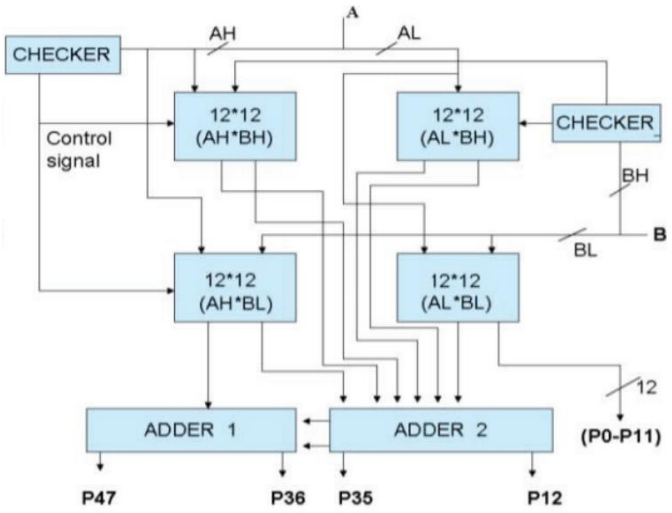

Fig.4. 24x24 CIFM Multiplier Architecture

It is possible to perform 24-bit mantissa multiplication using CIFM, and this yields the desired result using a 4x4 bit optimized multiplier. Four partial products are produced by a 4-bit multiplier, and these will be added simultaneously. Adjacency partial products are organized into 2-bit blocks, and the level 1 operation, as seen in fig. 10[4], is generated by a parallel adder picking the appropriate combination of adders to make a total of 2-bit. Blocks 5 and 6 add previously generated partial sums by selecting the appropriate set of adders, completing a level 2 operation. Level 3 operations use partial sums of the second level. Because the entire process has been split into multiple layers, power consumption can be greatly decreased. Therefore, operating in parallel can speed up the multiplier. Furthermore, CIFM multiplier is integrated with modified carry look ahead adder.

V. ARRAY MULTIPLIER

An array multiplier is a fundamental digital circuit design used to perform multiplication of binary numbers. In the context of your research, it's specifically employed for multiplying 32-bit complex floating-point numbers.

Basic Principle:

1. Partial Product Generation**:** The array multiplier breaks down the multiplication process into simpler steps. It multiplies each bit of one number (the multiplier) with each bit of the other number (the multiplicand). This generates individual product terms, each with a specific weight based on the bit positions being multiplied.
2. Shifting and Addition**:** These product terms are then shifted left according to their weight (number of zeros appended) and subsequently added together. The array structure facilitates performing these additions efficiently using adders arranged in a grid-like fashion.

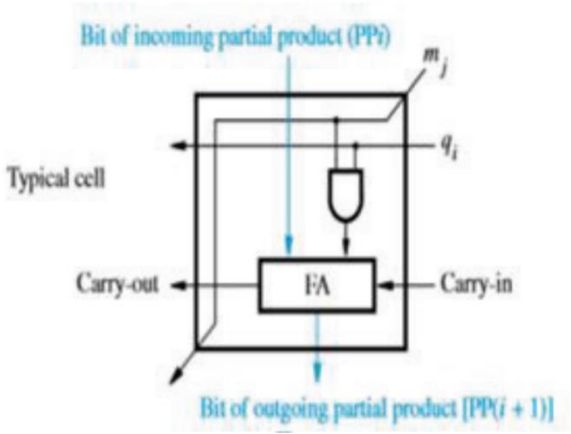

Fig.5. Array multiplier basic cell

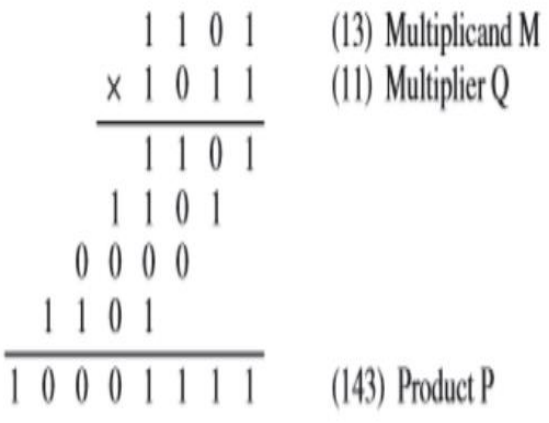

Fig.6. Array multiplier multiplication

VI. VEDIC MULTIPLIER

In [1], the Vedic multiplier is covered.sixteen sutras in all are included in the ancient Vedic multiplication. This study uses UrdhvaTiryakbhyam (U-T) sutra to perform 24x24 bit Vedic multiplication. A thorough discussion of each of them is outside the purview of this paper. Nine 8-bit VM (Vedic Multiplier) devices, which are also made up of ripple carry adders, are used to buildthe 24-bit VM.

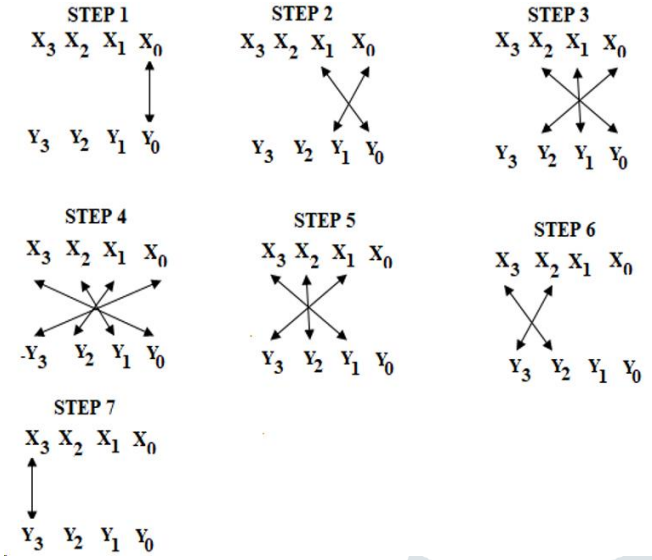

Fig.7. U-T sutra of 4-bit multiplication

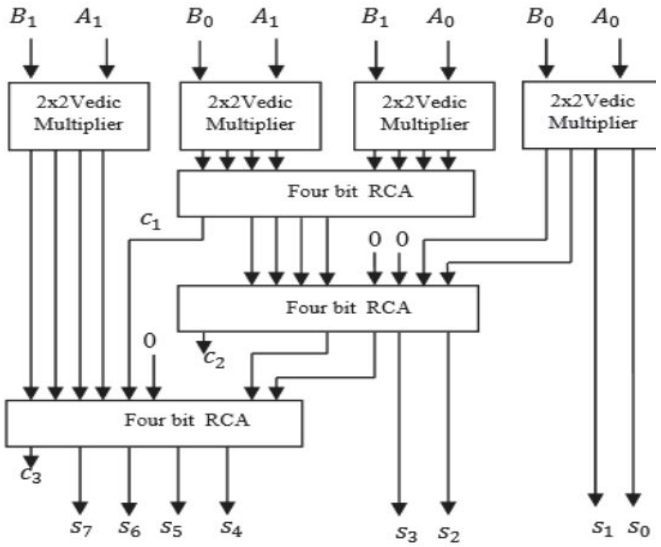

Fig.8. 4x4 Vedic multiplier module

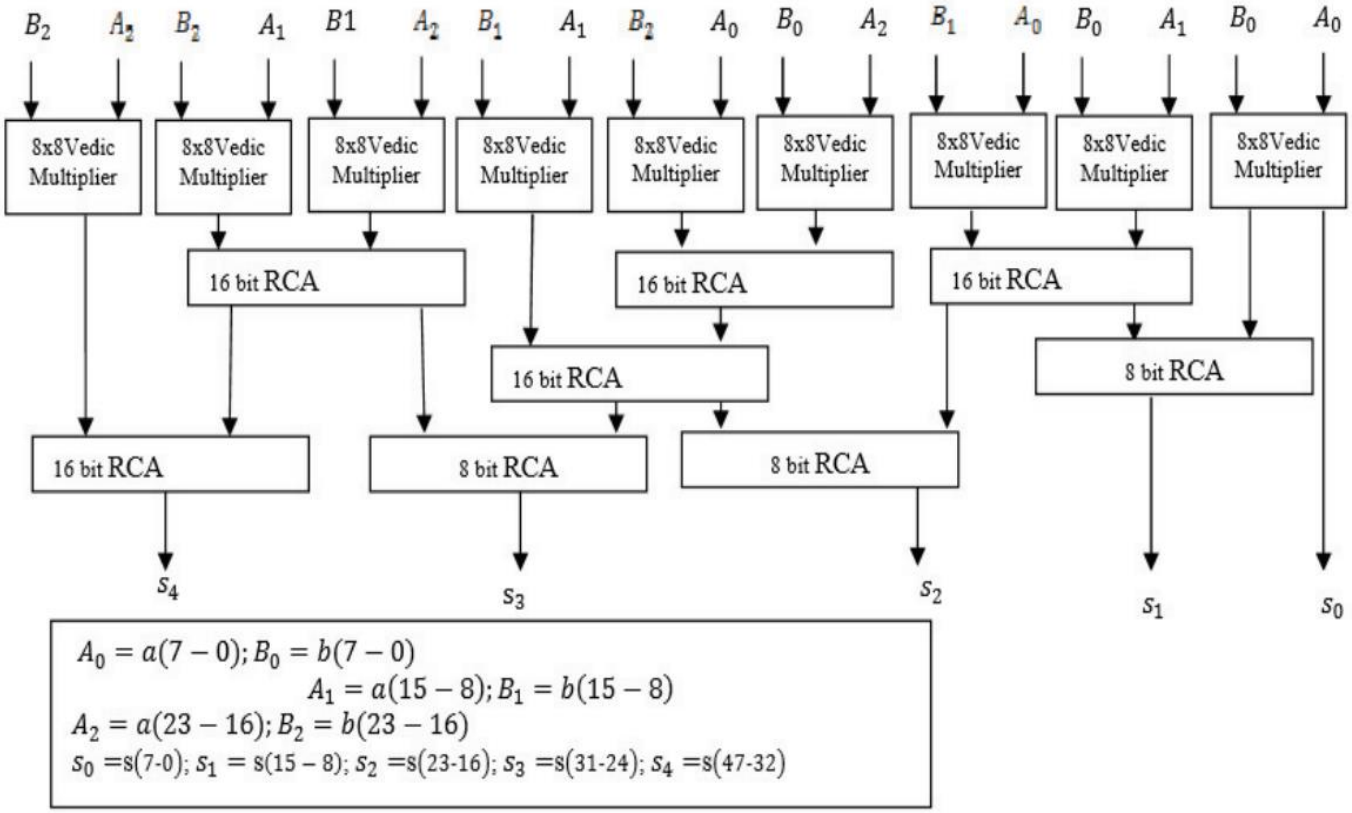

Fig.9. 24x24 Vedic Mutiplier module

VII.FLOATING POINT ADDER/SUBTRACTOR

In [5], the topic of floating point adder/subtractor is covered. A floating point adder and subtractor's primary components are as follows:

1. Mantissa alignment to achieve equality in the exponent (exponents compared by subtracting one another).
2. The aligned mantissas' addition and subtraction according to the sign bit.
3. If necessary, normalize the outcome.

Prior to normalization, the value of the exponent that is greater than both is used to determine the final result.
Once leading zeroes are found, they are moved until the MSB turns into a leading.

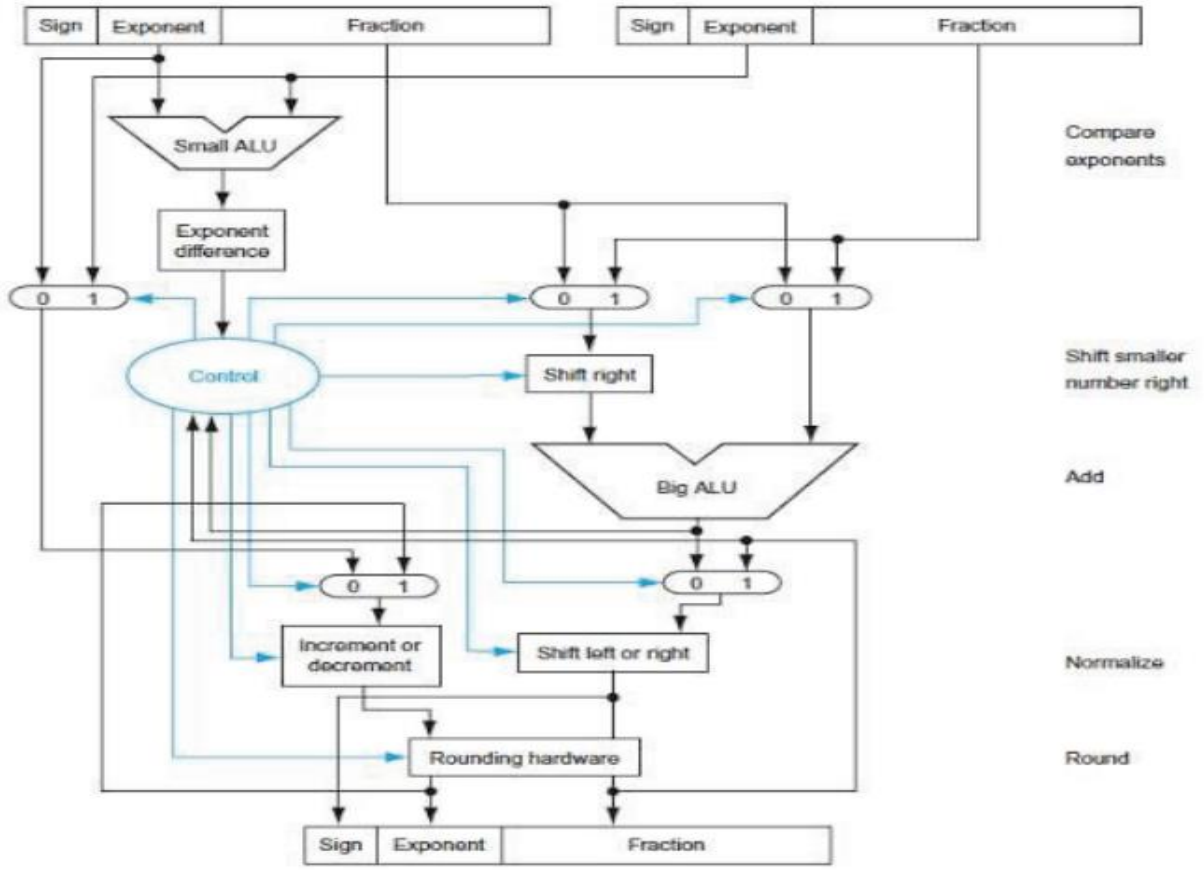

Fig.10. Floating point adder/subtractor Architecture

VIII. RESULTS AND DISCUSSION

The 32-bit complex multiplication using Vedic, Array, CIFM and CIFM Multiplier integrating Carry look ahead are implemented in Verilog; simulation is shown in Figure 11. The results are tabulated in Table 1. In summary, the results show that the floating point CIFM complex multiplier is much faster (34.852ns) than the Vedic multiplier (40.172ns) and array multiplier.

Parameters	Vedic Multiplier	Array Multiplier	CIFM Multiplier	CIFM Multiplier using Carry Look Ahead adder
Delay(ns)	40.172	61.380	34.852	31.145
Power(W)	1.417	1.417	1.297	1.322

Table.1. Comparison of complex floating point multiplier

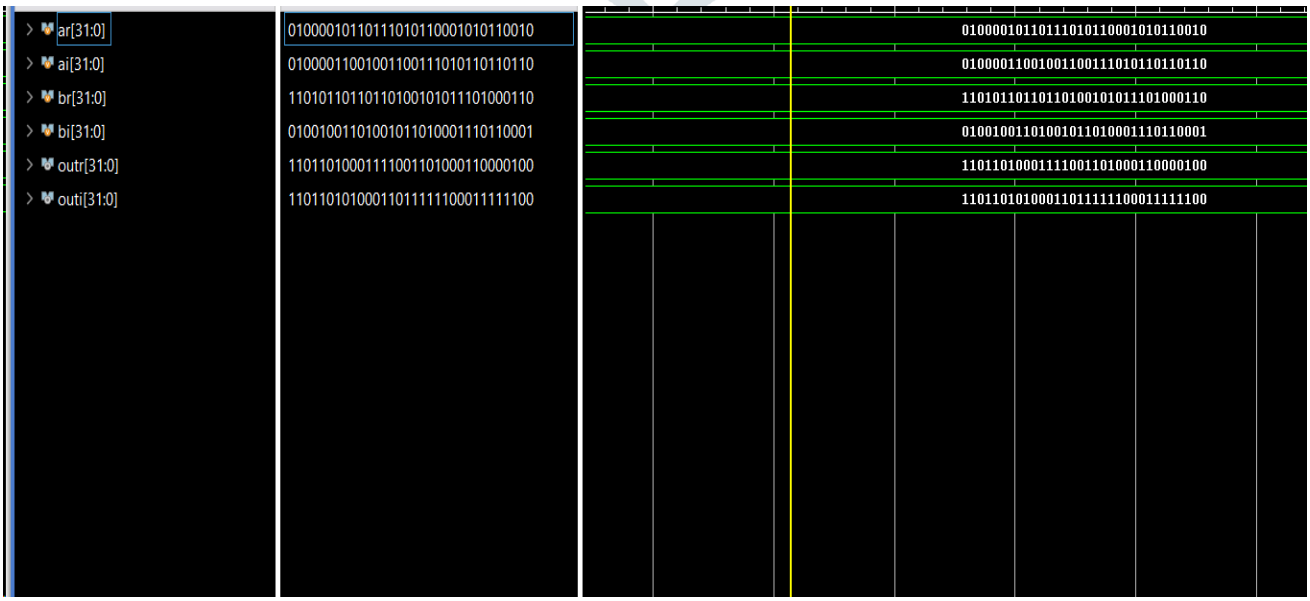

Fig.11. Simulation of 32-bit complex floating point multiplier

IX. CONCLUSION

This paper presented a comparative analysis of three multiplier architectures for implementing 32-bit complex floating-point multiplication: Vedic multiplier, array multiplier, and Combined Integer and Floating-point Multiplier (CIFM). We implemented each design and evaluated their performance based on delay and power consumption.

The findings of our research demonstrate that the CIFM multiplier achieved the lowest delay among the compared architectures and CIFM multiplier using carry look ahead achieved significant decrease in delay compared to CIFM multiplier. This suggests that the CIFM design effectively balances dedicated logic for integer and floating-point operations, leading to faster multiplication when combined with the efficiency of a CLA adder. While the CIFM with CLA resulted in a slight increase in power consumption compared to the other multipliers, the trade-off for achieving significantly lower delay might be advantageous for applications prioritizing high-speed complex number processing.

Explore alternative adder architectures beyond CLAs for the CIFM multiplier to potentially achieve a balance between delay, power, and area consumption. Evaluate the performance of these multipliers in real-world applications involving complex number processing to assess their practical impact.

REFERENCES

[1] 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering, FPGA Implementation of 32-Bit Complex Floating Point Multiplier Using Vedic Real Multipliers with Minimum Path Delay.

[2] H.D. Tiwari, G. Gankhuyag, C.M. Kim,, and Y.B. Cho, "Multiplier design based on ancient Indian Vedic Mathematics, "Proc. IEEE ISOCC'08, Vol.2, pp.65-68, 2008.

[3] "Computer Organization and Architecture", sixth edition by William Stallings.

[4] Himanshu Thapliyal, Hamid R Arabnia, A P Vinod, "Combined Integer and Floating Point Multiplication Architecture (CIFM) for FPGAs and its Reversible logic Implementation",2006 49thInernational Midwest Symposium on Circuits and Systems.

[5] Organization of Computer Systems:https://www.cise.ufl.edu/~mssz/CompOrg/CDA-arith.html.

[6] Anand Mehta, C. B. Bidhul, Sajeevan Joseph, Jayakrishnan. P " Implementation of Single Precision Floating Point Multiplier using Karatsuba Algorithm", 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE).