International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Design and Implementation of Fpga Based Complex Floating Point Multiplier Using Cifm

Dr. M. RAKESH[1], P V M Vijaya Bhaskar[2], Mannam Madhu Babu[3], K V Gautham[4]

[1]Associate Professor, Dept. of ECE, RISE Krishna Sai Prakasam Group of Institutions,Ongole, A.P.,
[2,3,4]Assistant Professor, Dept. of ECE, RISE Krishna Sai Prakasam Group of Institutions,Ongole, A.P.

ABSTRACT:
Floating point multiplication is one of the crucial operations in many application domains such as image processing, signal processing etc. But every application requires different working features. Some need high precision, some need low power consumption, low latency etc. The multiplication process requires more hardware resources and processing time when compared with addition and subtraction. This paper presents, Design and Implementation of FPGA based Complex Floating Point Multiplier using Combined Integer and Floating point Multiplier (CIFM). The Processing speed of the multipliers decides the execution time of the system as it consumes most of the time. The design is implemented in VHDL and design is synthesized on FPGA to know the performance. The architectures for the three multiplier solutions of complex multiplier for 32 x 32 bit complex numbers multiplication are coded in VHDL and implemented through Xilinx ISE 13.4 navigator and Modelsim 5.6 and their performance is compared. The complex floating point multiplication with single precision using CIFM multiplier has comparatively less amount of delay and power consumption with respect to Vedic and Array multiplier.

KEYWORDS: Vedic Real Multiplier, FPGA, CIFM, Array Multiplier.

I. INTRODUCTION

Multiplication involving complex numbers is of great importance in Image Processing (IP) and Digital Signal Processing (DSP). To implement the hardware module of Discrete Sine Transformation (DST), Discrete Cosine Transformation (DCT), Discrete Fourier Transformation (DFT), FIR filters, and modem broadband communications; requires large numbers of complex multipliers [1]. Four real number multiplications and two additions/ subtractions are used to perform complex number multiplication. Carry must be propagated from the least significant bit (LSB) to the most significant bit (MSB) when binary partial products are added. Therefore, the overall speed is limited by addition and subtraction after binary multiplications.

Floating point (FP) multiplication is an essential component required in a large set of FPGA based hardware acceleration application [2]. So, efficient implementation of FPGA based floating point multipliers are highly desirable. Floating point multiplication units are essential Intellectual Properties (IP) for modern multimedia and high performance computing such as graphics acceleration, signal processing, image processing etc [3]. There are lot of effort is made over the past few decades to

improve performance of floating point computations. Floating point units are not only complex, but also require more area and hence more power consuming as compared to fixed point multipliers. And the complexity of the floating point unit increases as accuracy becomes a major issue [4]. Even a minute error in accuracy can cause major consequences. These errors are possible in floating point units mainly because of the discrete behavior of the floating point representation.

Due to the high computational requirements of scientific applications such as computational geometry, climate modeling, computational physics, etc., it is necessary to have extreme precision in floating point calculations [5]. And these increased precision may not be provided with single precision or double precision format. That further increases the complexity of the unit. But some applications do not require high precision. Even an approximate value will be sufficient for the correct operation. For applications which require lower precision, the use of double precision or quadruple precision floating point units will be a luxury. It wastes area, power and also increases latency [6].

The speed of the processor is majorly determined by the processing speed of multipliers [7]. Hence, parallel and reconfigurable Field Programmable Gate Array (FPGA) based hardware architectures are needed to be designed. Moreover, due to the complex nature of wireless channel, complex multiplication process recently has received a significant importance in the area of broad band wireless communication techniques [8].

A collection of the ancient Indian mathematical tools and techniques called Vedic Mathematics , comprises of 16 Sutras (Formulae) [9]. "Urdhva-tiryakbyham" is a Sanskrit word which means "vertical and crosswise" formula, which is used as general case of multiplication [10]. "Nikhilam Navatascaramam Dasatah" also a Sanskrit term indicating "all from 9 and last from 10", formula is used for large number multiplication which are near to the base (i.e. 10, 100, 1000 etc.). The proposed multiplier is designed using "Urdhva-tiryakbyham" adopted from 16 sutras of ancient Indian Vedic Mathematics.

Using Vedic mathematics, high speed ASIC design of a complex multiplier is proposed [11] and implemented using the four real multipliers solution. However, FPGA implementation of a complex multiplier has not been discussed. Further, path delay analysis of Vedic real multiplier architectures, which will enable to choose architecture with minimum delay [12].

The organization of this paper is arranged as follows: Section II explains the literature survey, Section III explains the Complex Floating Point Multiplier, Section IV explains implementation details and finally paper is concluded with Section V.

II. LITERATURE SURVEY

Z. Gu and S. Li, et. al. [13] proposes a method of division-free Toom-Cook multiplication based Montgomery modular multiplication, which makes it possible for Toom-Cook multiplication to be applied in practical and efficient hardware implementations. We also provide a hardware implementation of modular multipliers of 256 bits and 1024 bits with advantages on area-time-product over previous researches.

P. Wang et al., [14] propose using the 3-D vertical channel NAND array architecture to implement the vector-matrix multiplication (VMM) with for the first time. Based on the array-level SPICE simulation, the bias condition including the selector layer and the unselected layers is optimized to achieve high computation accuracy of VMM. Since the VMM can be performed layer by layer in a 3-D NAND array, the read-out latency is largely improved compared to the conventional single-cell read-out operation. The impact of device-to-device variation on the computation accuracy is also analyzed.

R. Salarifard, S. Bayat-Sarmadi and H. Mosanaei-Boorani, et. al. [15] two low-complexity (LC) and low-latency (LL) architectures for the regular point multiplication using fixedbase comb method have been proposed. In this paper, a fixed-base comb point multiplication method has been used to perform regular point multiplication. The point multiplication architectures have been implemented using field-programmable gate array and application-specific integrated circuit (ASIC). Moreover, ASIC results show 100% energy improvement for the LC architecture implementation results over GF(2^{163}). In addition, the LL architecture has 99% reduction in point multiplication required time, respectively, using a pentanomial.

H. Saadat, H. Bokhari and S. Parameswaran, et. al. [16] proposes a novel error-configurable minimally biased approximate integer multiplier (MBM) design. The proposed MBM design is devised by coupling a unique error-reduction mechanism with an approximate log based integer multiplier. Then, we propose a set of new approximate FP multipliers and we show that these FP multipliers lie on the Pareto front on the design spaces of area versus error and power versus error. We synthesize the designs using the TSMC 45-nm standard-cell library. We also perform application-level evaluations of the proposed approximate integer and FP multipliers, showing that our proposed multipliers enable significant power and area reduction with minimal degradation in applications' output quality.

M. Kumm, M. Hardieck and P. Zipf, et. al. [17] Constant matrix multiplication (CMM), i.e., the multiplication of a constant matrix with a vector, is a common operation in digital signal processing. Like multiple constant multiplication (MCM), CMM can be reduced to additions/subtractions and bit shifts. Finding a circuit with minimal number of add/subtract operations is known as the CMM problem. While this leads to a reduction in circuit area it may be less efficient for power consumption or throughput. This paper addresses the optimization of CMM circuits which considers both adder depth and pipelining for the first time. For that, a heuristic is proposed which evaluates the most attractive graph topologies. It is shown that the proposed method requires 12.5% less adders with min. AD and 38.5% less pipelined operations. Synthesis results for recent FPGAs show that these reductions also translate to superior results in terms of delay and power consumption compared to the state-of-the-art.

S. -R. Kuang, C. -Y. Liang and C. -C. Chen, et. al. [18] presents a simple compression scheme and circuit to remove the data dependence in the accumulation process and accomplish one-cycle latency without quotient pipeline. To achieve low latency, existing radix-4 scalable architectures for word-based Montgomery modular multiplication usually suffer from high design and hardware complexities. The complex computation and encoding of quotient digits are thus avoided, leading to up to 10.6% and 17.7% reductions in area and power than previous work while maintaining very high performance.

Consequently, the proposed radix-4 scalable architecture appears to be very suited for low-complexity and low-power cryptographic applications.

S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park and N. S. Kim, et. al. [19] propose multiplier architectures that can tradeoff computational accuracy with energy consumption at design time. The need to support various digital signal processing (DSP) and classification applications on energy-constrained devices has steadily grown. Such applications often extensively perform matrix multiplications using fixed-point arithmetic while exhibiting tolerance for some computational errors. Compared with a precise multiplier, the proposed multiplier can consume 58% less energy/op with average computational error of ∼1 %. Finally, we demonstrate that such a small computational error does not notably impact the quality of DSP and the accuracy of classification applications.

X. Peng et al., [20] developing a large-scale reconfigurable data path (LSRDP) based on single-flux-quantum (SFQ) circuit technology for high-performance computing systems. In the SFQ LSRDP, a large number of SFQ floating-point adders (FPAs) and floating-point multipliers (FPMs) are directly connected to each other through routing networks to reduce a memory access rate. We show our recent results about the SFQ FPAs and FPMs. Utilization of the National Institute of Advanced Industrial Science and Technology's 10-kA/cm ^{2} Nb process makes it possible to accelerate the clock frequency to more than 50 GHz. We estimate the performance and energy efficiency of SFQ FPAs and FPMs based on the designed circuits.

III. COMPLEX FLOATING POINT MULTIPLIER

The architecture of the proposed multiplier based on four multipliers solution to multiply two complex floating point numbers a and b is shown in Fig. 1. In Fig. 1, p_r and p_i are the real and imaginary parts, respectively of the floating point result of multiplication of a and b.

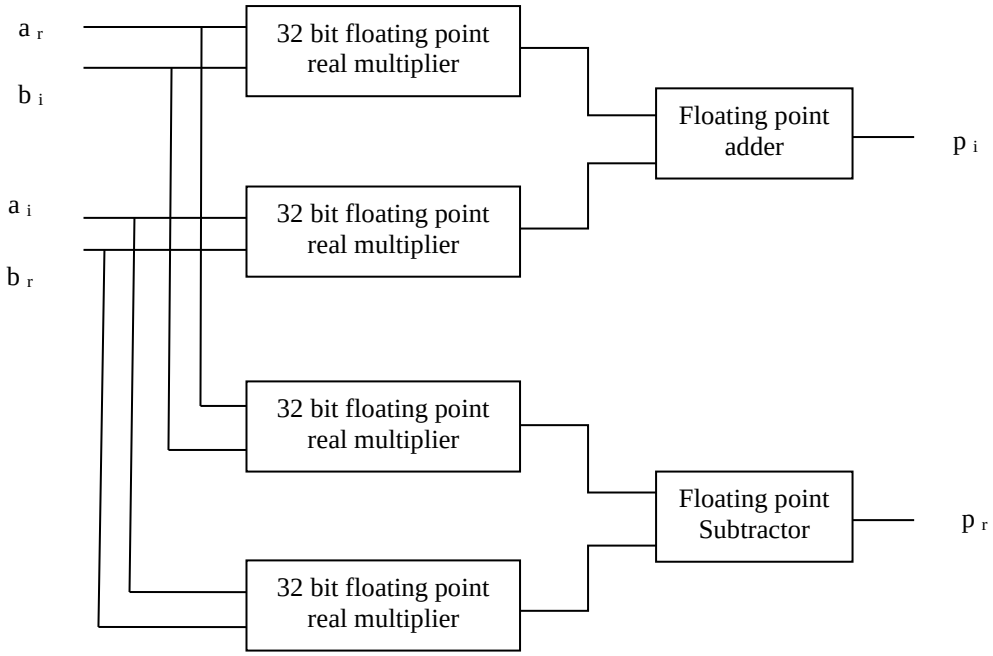

Fig. 1: ARCHITECTURE OF THE PROPOSED COMPLEX FLOATING POINT MULTIPLIER

The architecture of 32 bit floating point real multiplier is shown in Fig. 2. The architecture of floating point adder and subtractor is shown in Fig. 3. The floating point multiplier has three parts namely i) Sign bit ii) Exponent bit iii) Mantissa bit.

1. Sign bit: In the 32-Bit format, the MSB i.e the 31st bit is the sign bit, which is 0 for positive numbers and 1 for negative numbers.
2. Exponent Part: In the 32-Bit format the next 8 bits after the sign bit, i.e [30:23] bits are the exponent part, the exponent is from −127 to 128, which is in integer form of 8 bits and accepted as the biased form.
3. Mantissa Part: The Mantissa is about 23 bits and a leading bit with 1 at MSB, unless an exponent is stored with all zeros. Mantissa of 23 bits appears, but the total precision is 24 by concatenating 1 bit and multiplication is done by different multiplier algorithms.

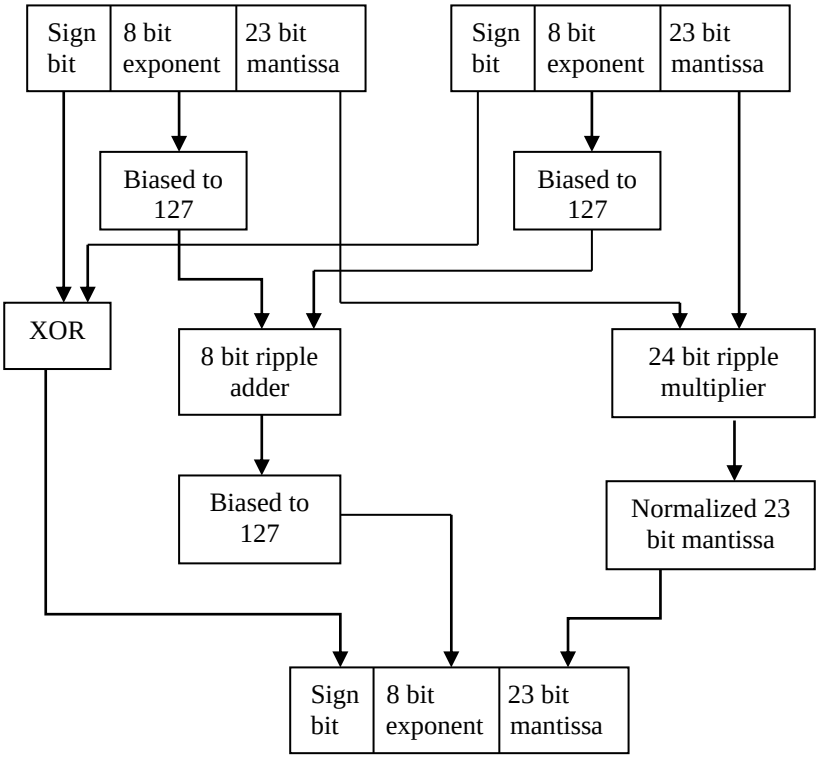

Fig. 2: 32 BIT FLOATING POINT REAL MULTIPLIER ARCHITECTURE

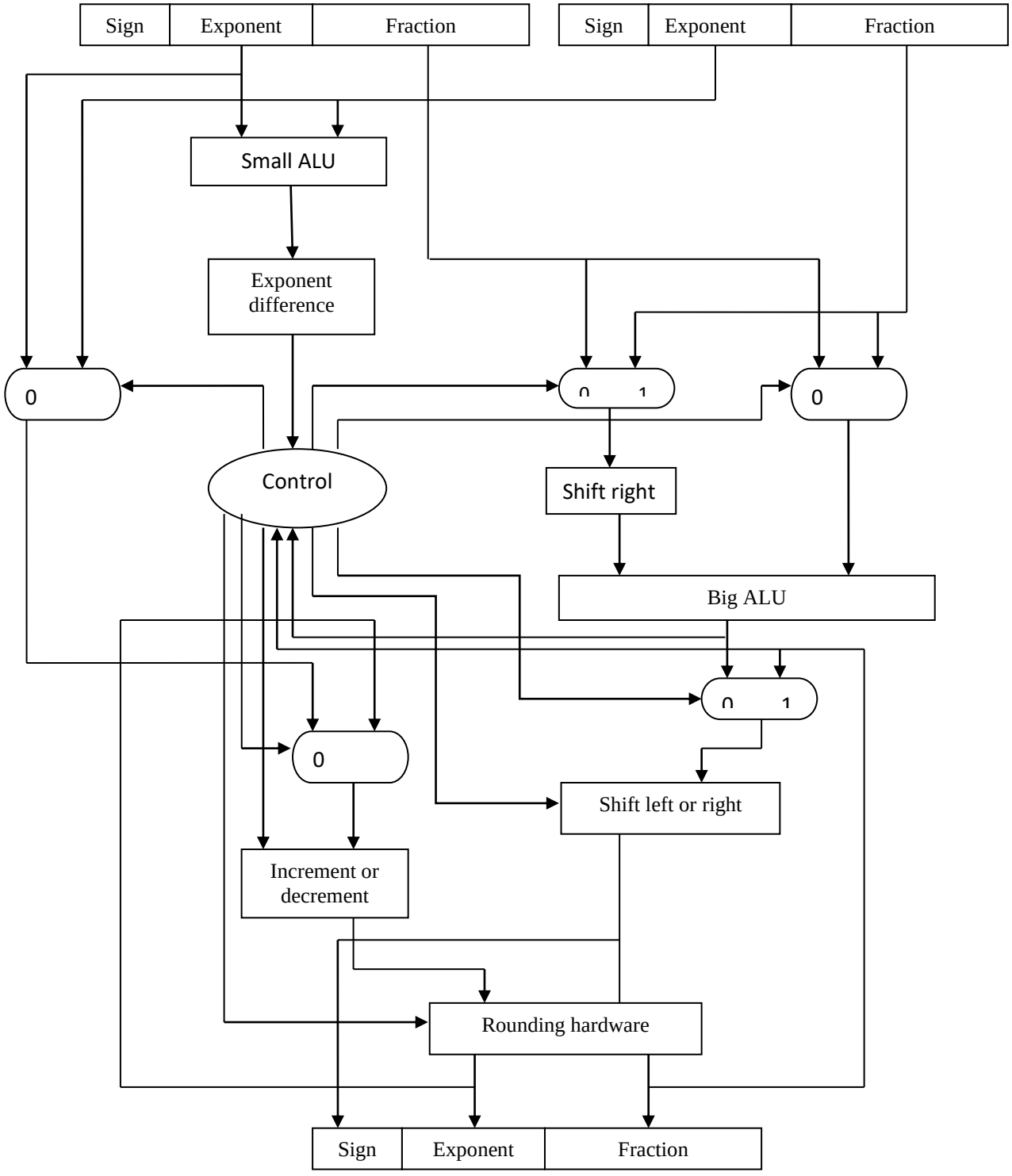

Fig. 3: FLOATING POINT ADD/SUB

The main parts of an floating point adder and subtractor are as follows:

1. Alignment of the mantissas to make the exponent equal (exponents compared from subtracting each other).
2. Addition/Subtraction of the aligned mantissas based on the sign bit.
3. Normalizing the result if required.

The value of the exponents which is larger among the two is taken for final output before the normalization. Leading zeroes are detected and shifted till the MSB becomes leading 1.

Complex floating point multiplication architecture is designed using Vedic multiplier, Array multiplier, CIFM multiplier are implemented with single precision floating point.

CIFM multiplier:

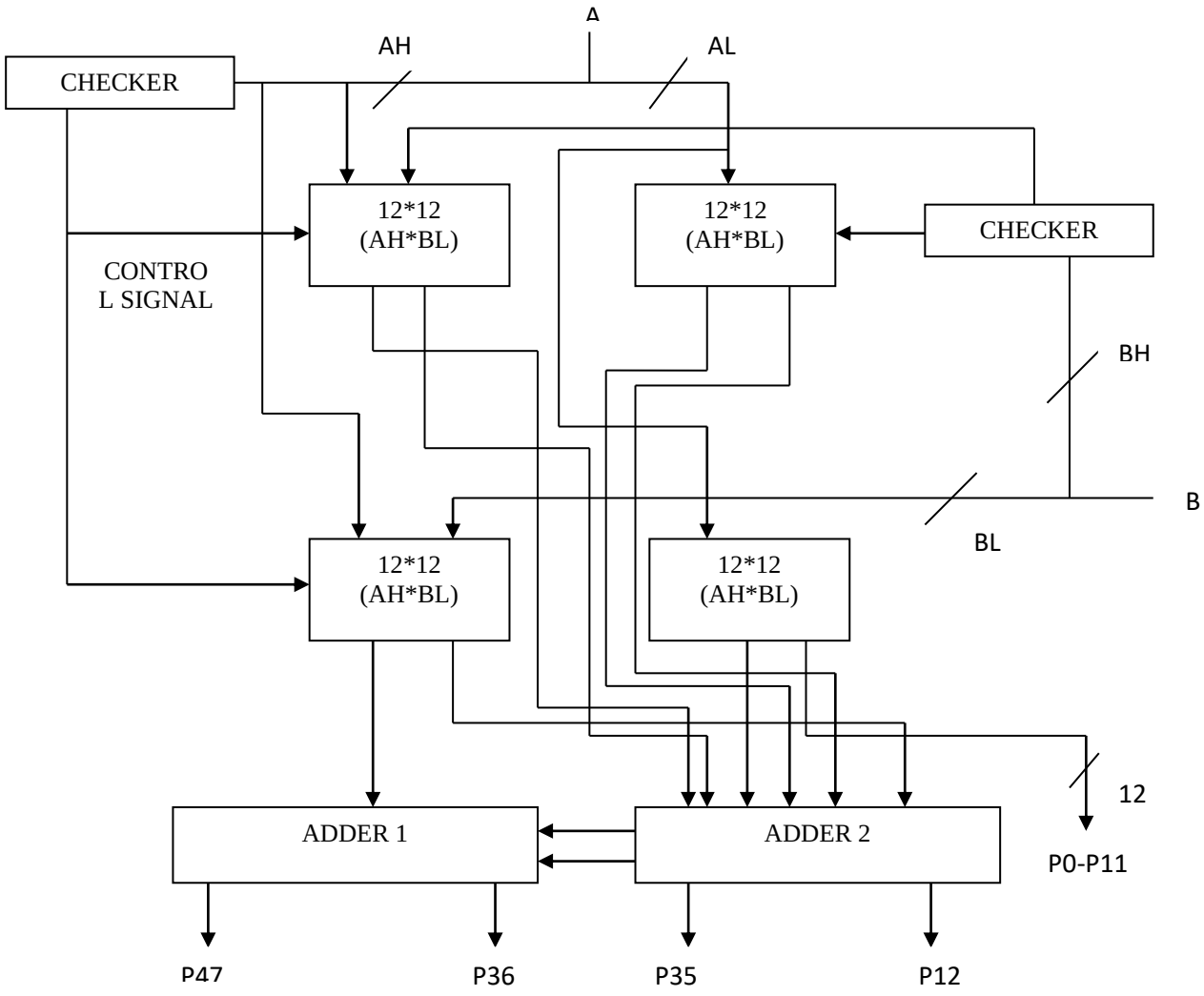

Fig. 4: 24X24 BIT CIFM MULTIPLIER

The 24-bit multiplication block is classified into four 12-bit multiplication modules which are working in parallel. The four 12 bits are AH, AL, BH and BL respectively used for 12x12 multiplication. The 12-bit multiplication modules are further divided using 4-bit optimized multipliers. The complete 24x24 bit multiplication is classified into 4x4 multiplier blocks. Checkers working as control signals for 24-bit multiplication module.

Using CIFM, 24-bit multiplication of mantissa can be done and it produces the required product using 4x4 bit optimized multiplier. 4-bit multiplier generates 4 partial products which will be added in parallel. Adjoining partial products are grouped into 2-bit blocks and sum of 2-bit is produced by a parallel adder by selecting the correct combination of adders, which forms the level 1 operation. Previously generated partial sums are added in block 5&6 by choosing the correct combination of adders, which forms the level 2 operation. Second level partial sums are used in level 3 operation.

Array multiplier:

Array multiplier is best known and simplest multiplier which uses add and shift method. By examining the multiplier bits and generating the partial products which take sequential operation and requires add and shift method. Addition is implemented row by row. An adder is required to generate the sum of partial products and carry combinations. (a*b) AND gates and (a-1) b bit adders are requires to produce the product of (a + b) bits for a multiplier bits and b multiplicand bits. The combinational circuit with 24-bit has been designed using a basic cell of an array which contains 23 cells in each row and shifted accordingly.

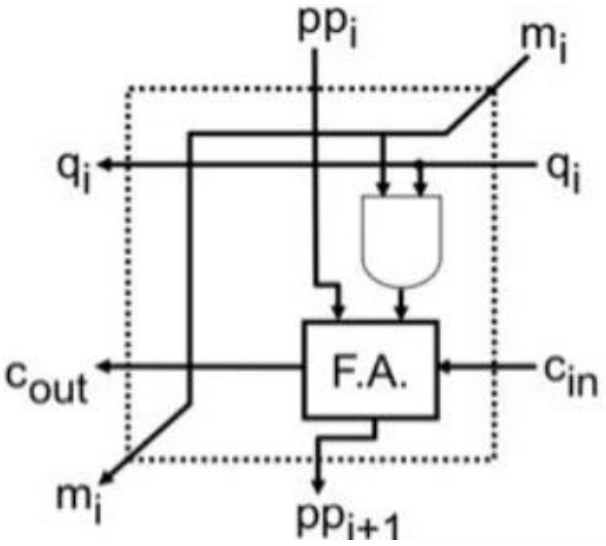

Fig. 5: BASIC CELL IN AN ARRAY MULTIPLIER

Vedic multiplier:

There are total sixteen sutras in ancient Vedic multiplication. In this paper 24x24bit Vedic multiplication is implemented using UrdhvaTiryakbhyam (U-T) sutra. The 24-bit VM (Vedic Multiplier) is implemented using nine 8-bit VM which also consists of Ripple carry adders.

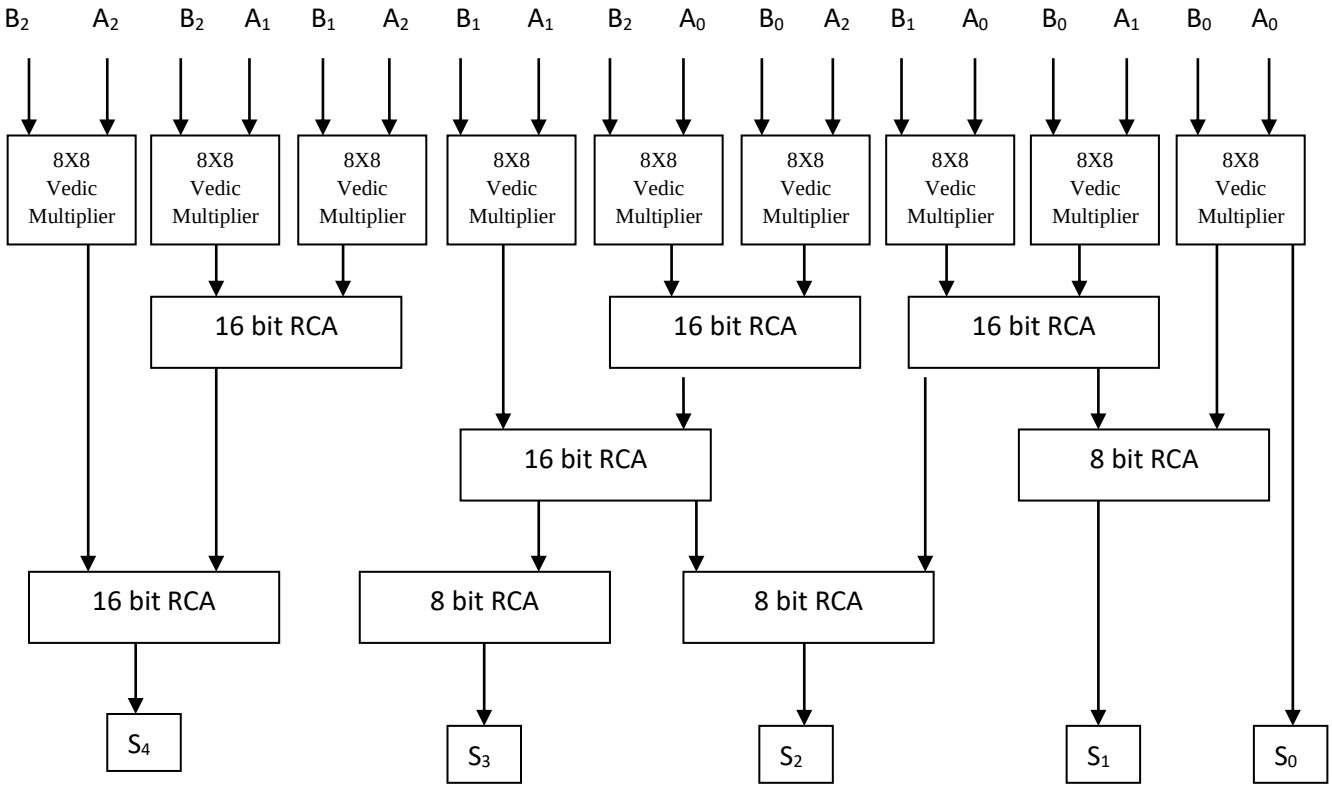

Fig. 6: ARCHITECTURE OF THE 24 X 24 BIT VEDIC MULTIPLIER USING 8X8 BIT VEDIC MULTIPLIERS

$$A_0 = a(7-0); B_0 = b(7-0)$$

$$A_1 = a(15-8); B_1 = b(15-8)$$

$$A_2 = a(23-16); B_2 = b(23-16)$$

$$S_0 = s(7-0);\ S_1 = s(15-8)$$

$$S_2 = s(23-16);\ S_3 = s(31-24)$$

$$S_4 = s(47-32)$$

IV. IMPLEMENTATION DETAILS

The 32x32 bit complex multiplier using three multipliers is implemented using VHDL and functionally verified using Xilinx ISE 13.4 and Modelsim 5.6 simulators. The 32-bit complex multiplication using Vedic, Array and CIFM multiplier is implemented in Verilog and analyzed based on performance factors such as delay and power. Simulation is shown in Fig. 7. The device Utilization of these three multipliers is summarized in Table 1.

Table 1: COMPARATIVE PERFORMANCE ANALYSIS

Parameter	CIFM multiplier	Array multiplier	Vedic multiplier
Path Delay (ns)	19.45	40.12	38.13
Number of Slice LUTs	10416	8954	11124
Logic Power (mW)	187.06	209.93	191.5
Number of bonded IOBs	228	102	320

Fig. 7: SIMULATION WAVE OF COMPLEX FLOATING POINT MULTIPLIER

Fig. 8 and Fig. 9 are shows three multipliers delay analysis and logic power analysis respectively.

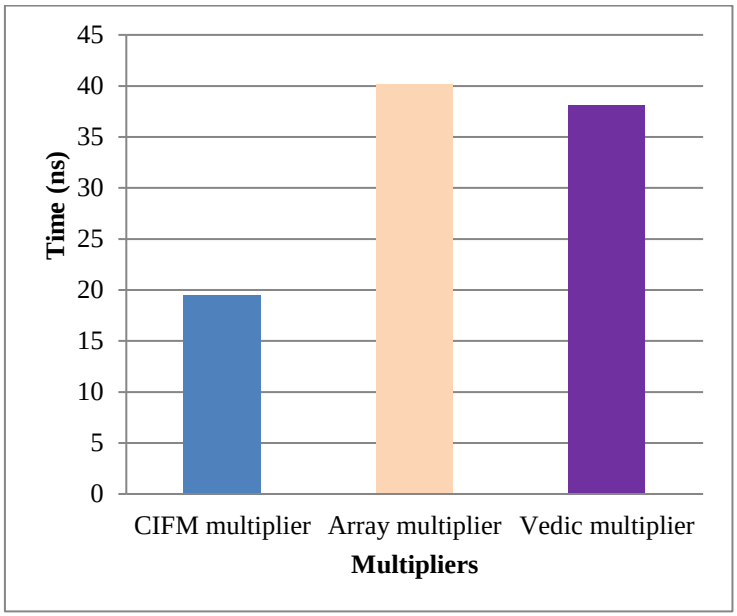

Fig. 8: DELAY ANALYSIS

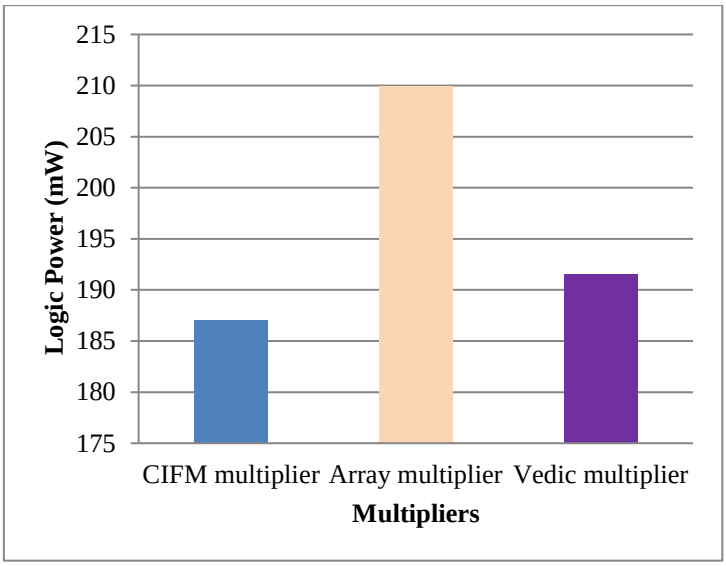

Fig. 9: LOGIC POWER ANALYSIS

It is summarized from the results that, the floating point CIFM complex multiplier is much faster (19.45ns) as compared to the Vedic multiplier (38.13ns) and array multiplier (40.12ns). The floating point CIFM complex multiplier has low power (187.07mW) than other two multipliers.

From these three designs we can draw the conclusion that, the complex floating point multiplication with single precision using CIFM multiplier has comparatively less amount of delay and power consumption with respect to Vedic and Array multiplier which proves to be advantageous for fast multiplication operation which can be used for DSP applications.

V. CONCLUSION

In this paper, Design and Implementation of FPGA based Complex Floating Point Multiplier using Combined Integer and Floating point Multiplier (CIFM). The architecture of Complex Floating Point Multiplier consists two main blocks as floating point real multiplier and floating point adder and subtractor. A design of Vedic real multiplier based on Urdhva Tiryakbhyam sutra of ancient Indian Vedic Mathematics. The 32x32 bit complex multiplier using three multipliers is implemented using VHDL and functionally verified using Xilinx ISE 13.4 and Modelsim 5.6 simulators. The performance of three multipliers is differentiated in parameters as tabulated. In Single Precision floating point multiplier for mantissa multiplication, different multipliers have been used for implementation in which CIFM consumes low power and delay to execute compared to Vedic and array multiplier. From these three designs we can draw the conclusion that, the complex floating point multiplication with single precision using CIFM multiplier has comparatively less amount of delay and power consumption with respect to Vedic and Array multiplier which proves to be advantageous for fast multiplication operation which can be used for DSP applications.

VI. REFERENCES

1. C. Eleftheriadis and G. Karakonstantis, "Optimal Adder-Multiplexer Co-Optimization for Time-Multiplexed Multiplierless Architectures," in IEEE Transactions on Circuits and Systems I: Regular Papers, doi: 10.1109/TCSI.2023.3289229.

2. A. Towhidy, R. Omidi and K. Mohammadi, "On the Design of Iterative Approximate Floating-Point Multipliers," in IEEE Transactions on Computers, vol. 72, no. 6, pp. 1623-1635, 1 June 2023, doi: 10.1109/TC.2022.3216465.
3. W. Li, Z. Chen, Z. Wang, S. A. Jafar and H. Jafarkhani, "Flexible Distributed Matrix Multiplication," in IEEE Transactions on Information Theory, vol. 68, no. 11, pp. 7500-7514, Nov. 2022, doi: 10.1109/TIT.2022.3204488.
4. B. Xiong, S. Fan, X. He, T. Xu and Y. Chang, "Small Logarithmic Floating-Point Multiplier Based on FPGA and Its Application on MobileNet," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 12, pp. 5119-5123, Dec. 2022, doi: 10.1109/TCSII.2022.3205945.
5. H. Zhang and S. -B. Ko, "Variable-Precision Approximate Floating-Point Multiplier for Efficient Deep Learning Computation," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 5, pp. 2503-2507, May 2022, doi: 10.1109/TCSII.2022.3161005.
6. S. Mach, F. Schuiki, F. Zaruba and L. Benini, "FPnew: An Open-Source Multiformat Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 4, pp. 774-787, April 2021, doi: 10.1109/TVLSI.2020.3044752.
7. W. Tan, B. M. Case, A. Wang, S. Gao and Y. Lao, "High-Speed Modular Multiplier for Lattice-Based Cryptosystems," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 8, pp. 2927-2931, Aug. 2021, doi: 10.1109/TCSII.2021.3064232.
8. H. A. S. Kamimura "Real-Time Passive Acoustic Mapping Using Sparse Matrix Multiplication," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 68, no. 1, pp. 164-177, Jan. 2021, doi: 10.1109/TUFFC.2020.3001848.
9. D. Peroni, M. Imani and T. S. Rosing, "Runtime Efficiency-Accuracy Tradeoff Using Configurable Floating Point Multiplier," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 2, pp. 346-358, Feb. 2020, doi: 10.1109/TCAD.2018.2885317.
10. A. Deepa and C. N. Marimuthu, "VLSI Design of a Squaring Architecture based on Yavadunam Sutra of Vedic Mathematics," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 1162-1167, doi: 10.1109/ICESC48915.2020.9155768.
11. N. Gadda and U. Eranna, "64-bit ALU Design using Vedic Mathematics," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-4, doi: 10.1109/ic-ETITE47903.2020.122.
12. S. Lad and V. S. Bendre, "Design and Comparison of Multiplier using Vedic Sutras," 2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA), Pune, India, 2019, pp. 1-5, doi: 10.1109/ICCUBEA47591.2019.9128517.
13. Z. Gu and S. Li, "A Division-Free Toom–Cook Multiplication-Based Montgomery Modular Multiplication," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 66, no. 8, pp. 1401-1405, Aug. 2019, doi: 10.1109/TCSII.2018.2886962.
14. P. Wang "Three-Dimensional nand Flash for Vector–Matrix Multiplication," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 4, pp. 988-991, April 2019, doi: 10.1109/TVLSI.2018.2882194.

15. R. Salarifard, S. Bayat-Sarmadi and H. Mosanaei-Boorani, "A Low-Latency and Low-Complexity Point-Multiplication in ECC," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 9, pp. 2869-2877, Sept. 2018, doi: 10.1109/TCSI.2018.2801118.
16. H. Saadat, H. Bokhari and S. Parameswaran, "Minimally Biased Multipliers for Approximate Integer and Floating-Point Multiplication," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2623-2635, Nov. 2018, doi: 10.1109/TCAD.2018.2857262.
17. M. Kumm, M. Hardieck and P. Zipf, "Optimization of Constant Matrix Multiplication with Low Power and High Throughput," in IEEE Transactions on Computers, vol. 66, no. 12, pp. 2072-2080, 1 Dec. 2017, doi: 10.1109/TC.2017.2701365.
18. S. -R. Kuang, C. -Y. Liang and C. -C. Chen, "An Efficient Radix-4 Scalable Architecture for Montgomery Modular Multiplication," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 63, no. 6, pp. 568-572, June 2016, doi: 10.1109/TCSII.2016.2530801.
19. S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park and N. S. Kim, "Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 6, pp. 1180-1184, June 2015, doi: 10.1109/TVLSI.2014.2333366.
20. X. Peng "High-Speed Demonstration of Bit-Serial Floating-Point Adders and Multipliers Using Single-Flux-Quantum Circuits," in IEEE Transactions on Applied Superconductivity, vol. 25, no. 3, pp. 1-6, June 2015, Art no. 1301106, doi: 10.1109/TASC.2014.2382973.