Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 6, No. 3, September 2018, pp. 333~342
ISSN: 2089-3272, DOI: 10.11591/ijeei.v6i3.532

Resource Efficient Single Precision Floating Point Multiplier Using Karatsuba Algorithm

K V Gowreesrinivas, P Samundiswary
Pondicherry University, India

Article Info

Article history:

Received Sep 3, 2018
Revised Sep 5, 2018
Accepted Sep 6, 2018

Keyword:

Compressor
Karatsuba algorithm
Multiplexer
Vedic multiplier
Xilinx

ABSTRACT

In floating point arithmetic opearations, multiplication is the most required operation for many signal processing and scientific applications. 24-bit length mantissa multiplication is involved to obtain the floating point multiplication final result for two given single precision floating point numbers. This mantissa multiplication plays the major role in the performance evaluation in respect of occupied area and propagation delay. This paper presents the design and analysis of single precision floating point multiplication using karatsuba algorithm with vedic multiplier with the considering of modified 2x1 multiplexers and modified 4:2 compressors in order to overcome the drawbacks in the existing techniques. Further, the performance analysis of single precision floating point multiplier is analyzed in terms of area and delay using Karatsuba Algorithm with different existing techniques such as 4x1 multiplexers and 3:2 compressors and modified techniques such as 2x1 multiplexers, 4:2 compressors. From the simulation results, it is observed that single precision floating point multiplication with karatsuba algorithm using modified 4:2 compressor with XOR-MUX logic provides better performance with efficient usage of resources such as area and delay than that of existing techniques. All the blocks involved for floating point multiplication are coded with Verilog and synthesized using Xilinx ISE Simulator.

Corresponding Author:

K V Gowreesrinivas,
Pondicherry University,
India.
Email: srinu43306@gmail.com

1. INTRODUCTION

In many applications like signal and numerical processing, floating point arithmetic operations are mostly used. The floating point number standard is defined by IEEE [3] for different formats like single-precision and double-precision. In floating-point operations, multiplication process is the complex one and performance deciding block. Hence, efficient implementation of floating-point multipliers is the major concern.

Over the last few decades, a lot of work has been done at both algorithmic level and implementation level done to improve the performance of floating point computations [13]-[15]. Several works have also focused on implementation in FPGA platforms [11],[20]. In spite of tremendous efforts, this arithmetic is still often the bottleneck in many computations.

The mantissa multiplication is the main part of the floating-point multiplication with respect to performance. For the single precision numbers 24-bit length of the mantissa, and generally multipliers need more hardware [1]-[2]. In this work, an attempt has been made to develop an approach for the mantissa multiplication of floating point numbers for single precision has been presented which permits to use a lesser amount of complexity and rich in performance.

The contribution of this paper can be summarized as follows:

- First, Single precision floating multiplier using Karatsuba algorithm is designed using Vedic multiplier

and analyzed with different techniques such as Multiplexers and Compressors for area-efficient implementation of Single Precision Floating-Point Multiplier.

- Further in multiplication process, full adder block is replaced with modified 2x1 multiplexers and modified 3:2 & 4:2 compressor techniques wherever addition is done with full adder to improve the performance of the multiplication block.
- Furthermore, performance analysis is summarized for proposed single precision floating point multiplication.

The respite of the paper is systematized as follows: Section 2 explains the implementation of the floating-point multiplication, design approach of karatsuba algorithm is discussed in Section 3, karatsuba algorithm using existing techniques are described in Section 4. In Section 5, karatsuba algorithm using proposed techniques are designated, simulation results are discussed in section 6.

2. FLOATING POINT MULTIPLICATION

The floating-point binary format are defined by IEEE-754 standard which is used for representing floating point numbers [12]. This standard specifies the format for single and double precision numbers i.e., 32-bit and 64-bit respectively. FPU arithmetic computations comprise addition, subtraction, multiplication, division and inverse etc. Generally, arithmetic operations of floating point numbers involves the mantissa, exponent and sign parts of the operands and then combining them after rounding and normalization. Brief overview of the computational flow of this arithmetic operation is given below.

Floating point multiplication is like normal multiplication with, excluding that it entails a complex mantissa multiplication which needs 24-bit large multiplier. which stances restriction on the performance of the hardware design. It can be figured in the subsequent steps [3]:

- XOR operation of the MSB bits to acquire the sign bit of the final product.
- Addition of the exponents parts of the given numbers.
- Multiplication of mantissas parts of the given numbers.
- Rounding operation is performed for final mantissa product.
- Final step is to normalize the acquired result to regulate exponent and mantissa parts.

Figure 1 illustrates the floating-point multiplication process through flow chart. It requires the calculation of exponent, mantissa and sign individually. Each of these are discussed below [5],[9]:

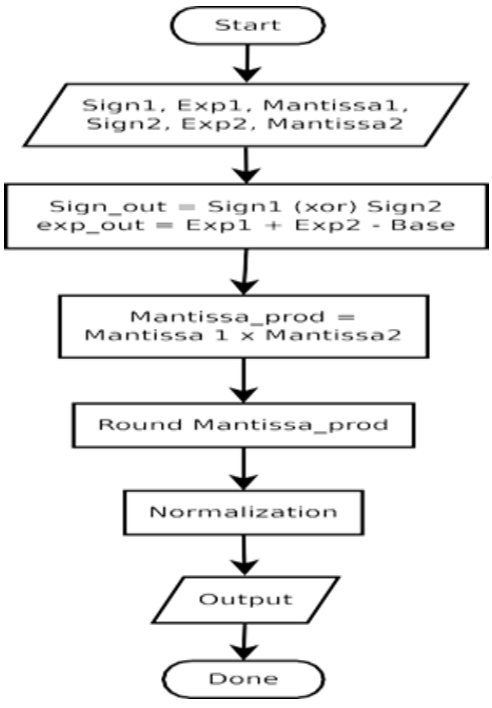

Figure 1. Floating point multiplication procedure using flow chart [3]

2.1. Floating Point Multiplication

- The logical XOR of the sign-bit of both operands gives the output sign: Out = in1 ⊕ in2
- The addition of both input exponents is given by final output exponent and then modify it by Base.

  Out1 = Exp_in1 + Exp_in2 – 127 (SPFPM)
  Out2 = Exp_in1 + Exp_in2 – 1023 (DPFPM) (1)

- Generally for any given floating-point number the base is drawn using this expression: $(2^{\text{exponent bits}-1} - 1)$.
- The output of the mantissa multiplication is Z = (-1S) *2(E-Bias) *(1.M)
- If there is an additional carry generated after multiplication, the product result is right shifted by 1-bit and the exponent result is incremented by one to make the result normalized.
- Rounding is required in order to trim back the 106-bit mantissa multiplication result to 53-bit only. This can be done as per IEEE standard [16]-[17].

3. DESIGN APPROACH OF KARATSUBA ALGORITHM FOR MULTIPLICATION

The Karatsuba algorithm for floating-point multiplication follows divide and conquer method. The basic steps for this algorithm are discussed below [10]. Let us consider two floating point numbers W and X : the below steps will explain how the algorithm follows to perform multiplication:

- Firstly, Divide each mantissa part into three parts such that each part should have equal number of bits i.e., 8-bits namely A_0, A_1, A_2 and B_0, B_1, B_2 for the inputs A and B commonly.
- The procedure for Karatsuba multiplication is interpreted using equations which are drafted below. This perception utilizes appropriate amount of adder blocks in places of few multipliers to improve the hardware efficiency.

$$A = A_2\, 2^{2n} + A_1\, 2n + A_0 \quad (2)$$

$$B = B_2 2^{2n} + B_1 2n + B_0 \quad (3)$$

$$AB = A_2\, B_2\, 2^{4n} + (A_2\, B_1 + A_1\, B_2)2^{3n} + [(A_2\, B_0 + A_0\, B_2) + A_1\, B_1]2^{2n} + (A_0\, B_1 + A_1\, B_0)2^{n} + A_0\, B_0 \quad (4)$$

- From the equation (4), it is observed that Karatsuba algorithm needs 9 multipliers and 8 adders to get result.
- In this, to perform mantissa multiplication vedic multiplier is used with different techniques like multiplexers and compressors [6], [4].

Figure 2 illustrates, multiplication of two 24-bit mantissa numbers using Vedic multiplier with Ripple Carry addition.

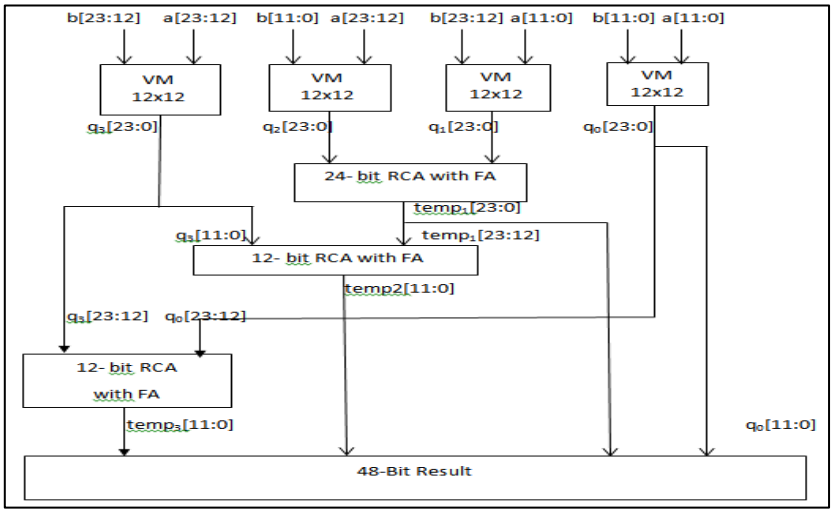

Figure 2. 24-bit mantissa multiplication using vedic multiplier

3.1. Vedic Multiplier

Vedic multiplier is developed based on Urdhava Triyakbhayam sutras. Partial products creation can be done using vertical and crosswise manner and then parallel addition of these partial product is done by using different available adders. In this paper, mantissa bits of both the numbers are multiplied using Vedic multiplier which works on the principle of UT. The partial products creation and their summation are produced in one step and which minimize the carry feeding from LSB to MSB. Because of this speed of the multiplier is improved as compared to the available multipliers [7]-[9].

Figure 3 illustrates the single precision floating point multiplication of two 24-bit numbers using Vedic multiplier with Ripple Carry Adder using multiplexers.

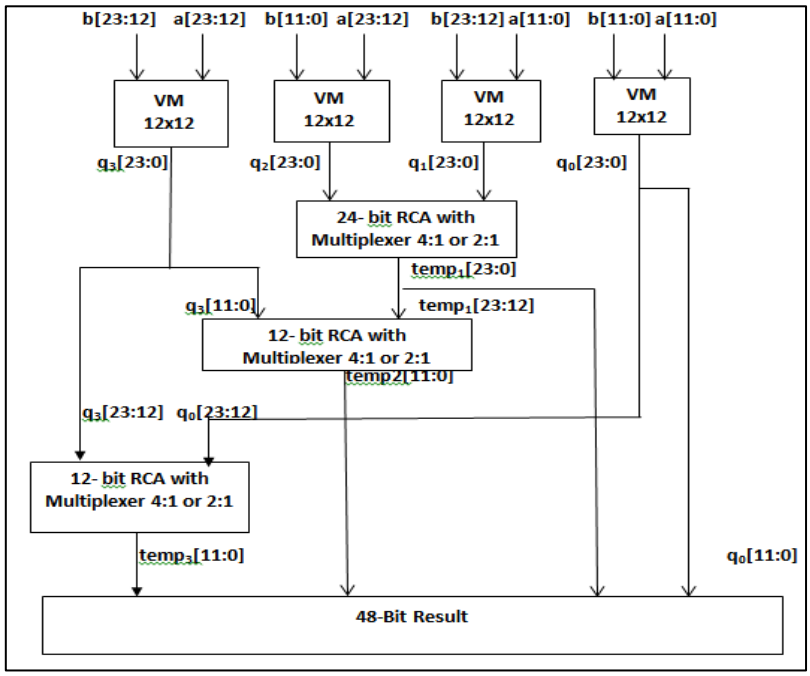

Figure 3. 24-bit mantissa multiplication using vedic multiplier with multiplexers

4. EXISTING WORK

The various existing techniques are listed below:

- Ripple Carry Adder is used for partial products addition in Vedic multiplier which is used in Karatsuba algorithm.
- Next, in place of Ripple Carry Adder multiplexers are used for partial products addition in Vedic multiplier which is used in Karatsuba algorithm.
- Finally, different 3:2 techniques with XOR-Mux and XOR-XNOR-Mux logics are used for partial products addition in Vedic multiplier which is used in Karatsuba algorithm.
- Finally, performance constraints of all existing techniques are discussed in this paper.

4.1. Karatsuba Algorithm with RCA using Full Adder based Multiplication

The Ripple Carry Adder with full adder is used to perform partial product addition. In Karatsuba algorithm, multiplication is used RCA for addition.

4.2. 4x1 Multiplexer based SPFP Multiplication using Karatsuba Algorithm

Full adder block is planned using two 4x1 multiplexers to complete partial product addition which is shown in Figure 4. From the Figure 4, A, B,C are considered as three inputs, among these three B,C inputs are acts as selection lines for both multiplexers and I0,I1,I2,I3 are input values to multiplexer. Out of these four inputs two input lines in first multiplexer I0, I3 is tied to A value and I1, I2 are tied to $\bar{A}$ for generating sum output. For, second multiplexer I1, I2 are tied to A input value and I0 is connected to logic 0 value and I3 connected to logic 1 value.

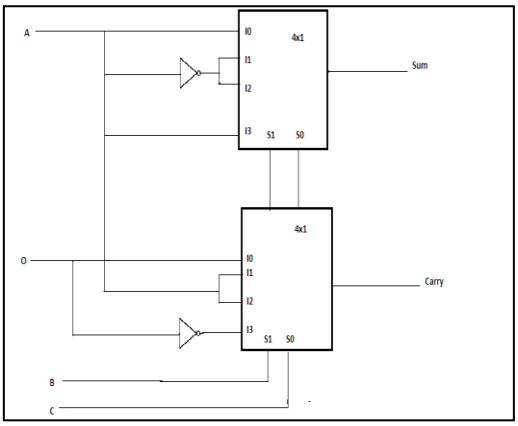

Figure 4. Full-adder using 4x1 multiplexer

4.3. Karatsuba Algorithm with 3:2 Compressor based multiplication

In multiplication, different types of compressors are used to add the partial product addition to improve the performance capability [15]. Compressors are usually involved to reduce the critical path which is important to improve the performance at the stage of reduction of the partial products. The symbolic representations of 3:2 compressors using XOR-Mux and XOR-XNOR-Mux logics are shown in Figure 5 and Figure 6.

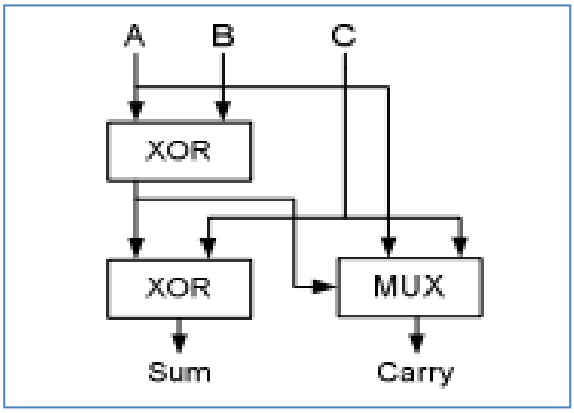

Figure 5. 3:2 Compressor using XOR-Mux logic

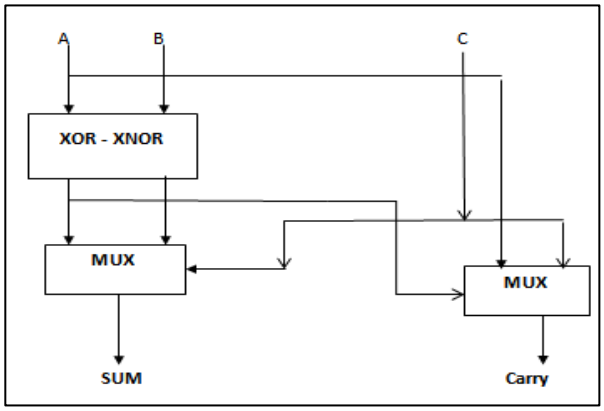

Figure 6. 3:2 Compressors with XOR-XNOR-Mux Logic

5. PROPOSED WORK

The contribution of this paper is involved in improving the performance of the single precision floating point multiplication. In this, Karatsuba algorithm is analyzed and to further improvement of this algorithm, different modified techniques are used in multiplier part. Brief outline of the modifications in mantissa multiplication part are discussed below:

- Firstly, mantissa multiplication for single precision numbers is completed with modified multiplexers 1 and 2 models using two 2x1 Multiplexers. Adder in the above-mentioned multiplication is replaced with 4x1 multiplexer and 2x1 multiplexer to improve the area and speed.
- Mantissa multiplication for single precision numbers is performed using modified 4:2 compressor techniques with XOR-MUX and modified 2x1 multiplexers. And also, it is done with XOR-XNOR-MUX and 2x1 multiplexers to improve the area and speed
- Performance Constraints of proposed techniques are summarized and conclusion is drawn.

In next section, detailed explanation is given for each proposed model designed with multiplexer and 4:2 compressor logics.

5.1. Modified 2x1 Multiplexer based Multiplication

In this, multiplexers are replaced in place of full adder to add partial products. Full adder block is designed using two 2x1 multiplexers which are explained using Figure 7 and Figure 8. From Figure 7, outputs of the XNOR, XOR with A, B are the inputs to the first multiplexer and it gives sum output. Outputs of the AND and OR of A, B are the inputs to the second multiplexer and it bounces final carry. Here, common select line for both multiplexers is cin.

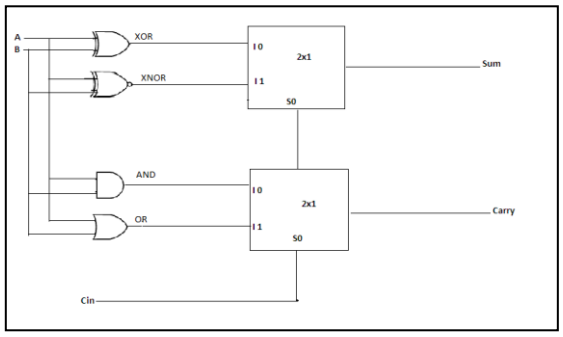

Figure 7. Full adder using modified 1 2x1 multiplexer

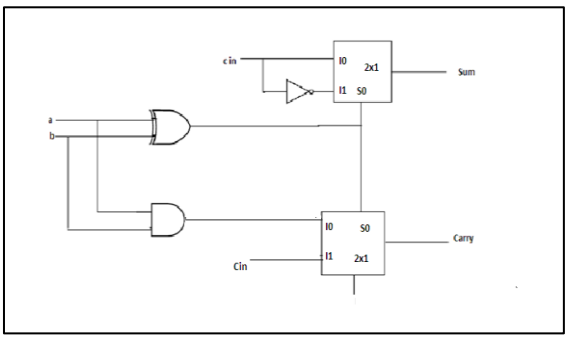

Figure 8. Full adder using modified 2 2x1 multiplexer

The performance comparison is done for modified models of multiplexer based single precision floating point multiplication with respect to area and delay.

5.2. Modified Compressor based Multiplication

In this, compressors are replaced in place of full adder to add partial products to achieve better results in terms of delay and power. To design 4:2 compressor two types of modules are involved: XOR-XNOR-Mux block and XNOR-MUX. The representation of 4:2 compressors using both the logics are shown in Figure 9 and Figure 10.

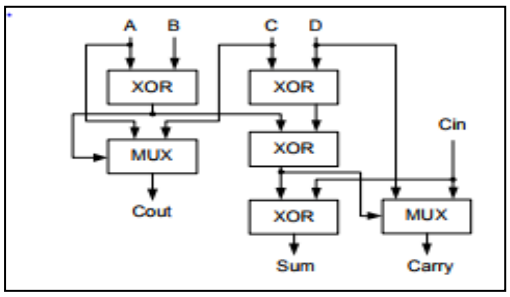

Figure 9. 4:2 Compressor with XOR-MUX logic

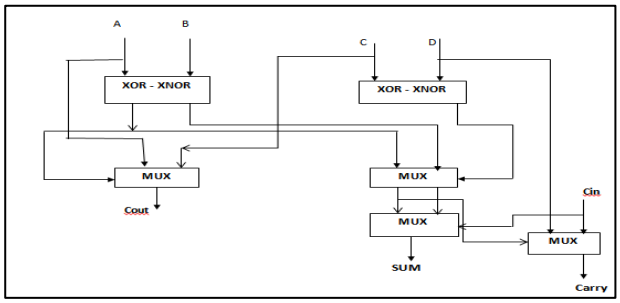

Figure 10. 4:2 Compressor with XOR-XNOR-Mux logic

6. SIMULATION RESULTS

The modules involved in floating point multiplication are implemented by using Verilog-Hardware Description Language. All blocks are simulated and synthesized on FPGA targets with Xilinx ISE. In this, karatsuba algorithm with vedic multiplier using different existing and proposed techniques are considered for simulation and synthesis. Further, the performance comparison is done for those different existing and proposed techniques in view of delay and area.

6.1. Simulation Results of SPFPM using Existing and Proposed Techniques

Figure 11-14 depicts, that the device utilization summary in terms of number of slices and LUTs, delay and performance comparison analysis for different existing and proposed techniques of Single Precision Floating Point multiplication. From the graphs it is concluded that 4:2 compressor with XOR-Mux provides improved results in the point of both parameters i.e. number of slices and 4-input LUTs.

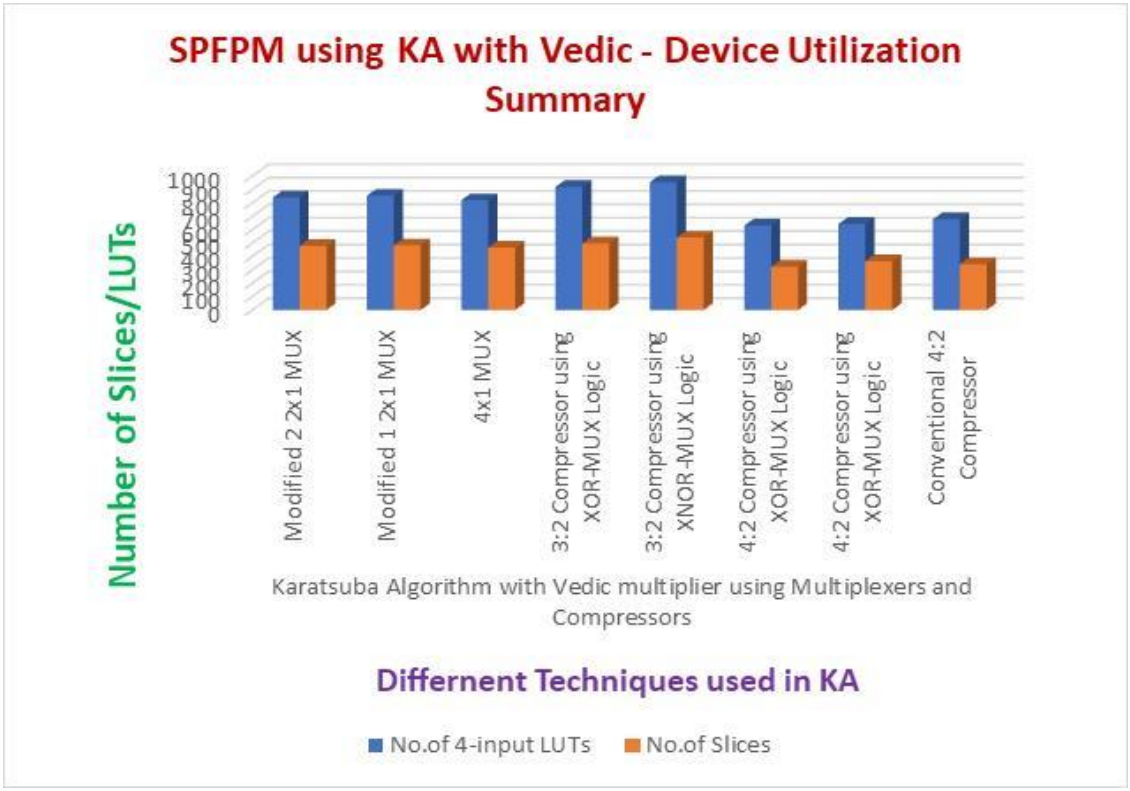

Figure 11. Device utilization summary of KA with vedic using different existing and proposed techniques

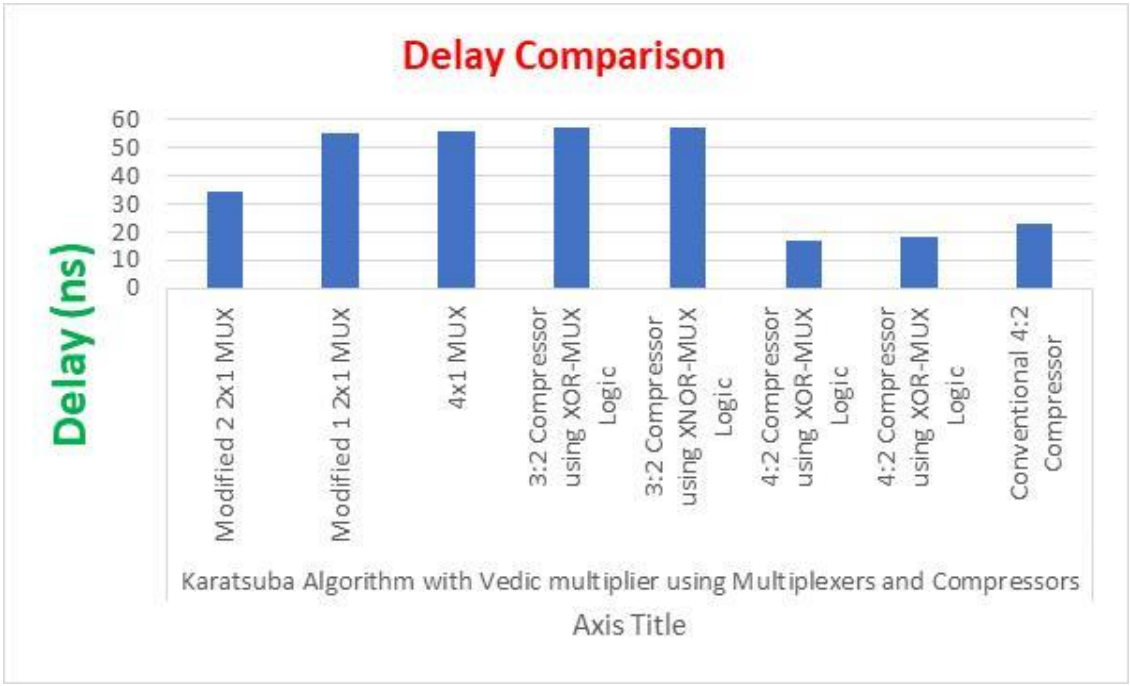

Figure 12. Delay analysis of KA with vedic using different existing and proposed techniques

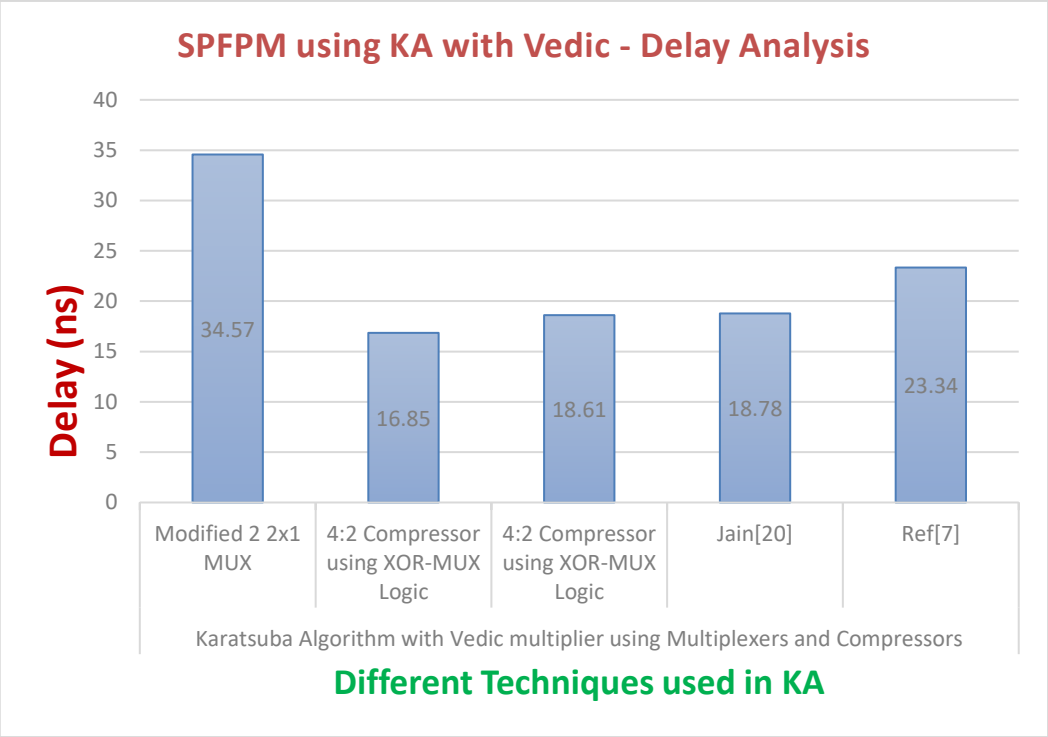

Figure 13. Delay summary of KA with vedic using different existing and proposed techniques

6.2. Performance Comparison of Efficient Techniques

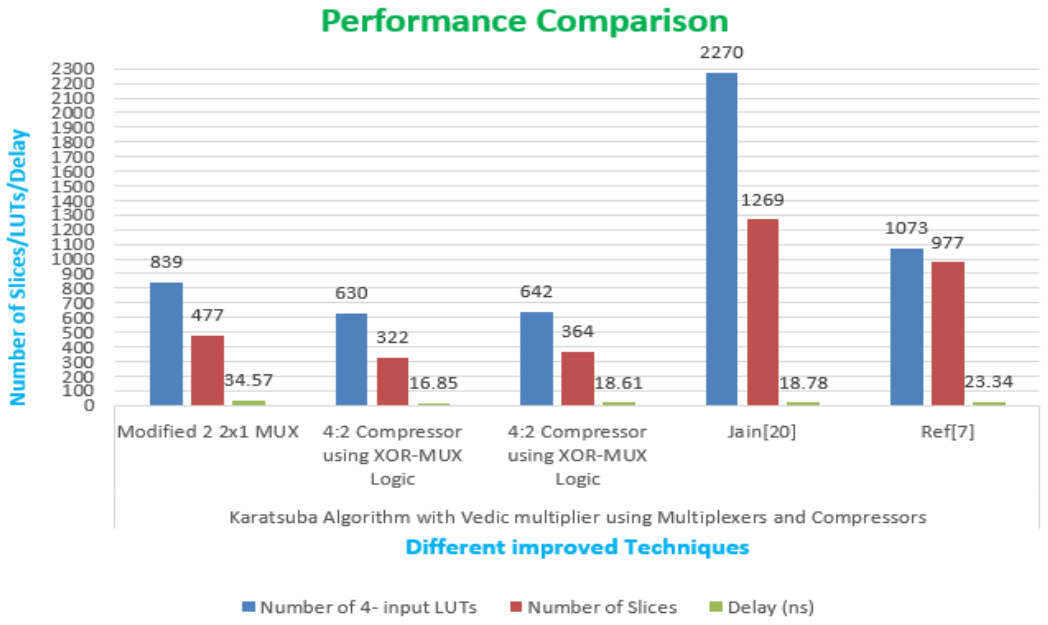

Figure 14. Performance comparison of KA with vedic using different improvised techniques

6.3. Comparison Analysis

From Table 1, it is noticed that SPFPM using karatsuba algorithm with vedic multiplier with the consideration of 4x1 MUX achieves better results in terms of area. But karatsuba algorithm with vedic multiplier with consideration of modified 2 2x1 MUX provides better results in terms of delay than that of Full adder and Multiplexer based Single Precision Floating Point Multipliers.

Table 1. Area and Delay of SPFPM KA with Vedic using FA and Multiplexers (existing and proposed)

Single Precision Floating Point Multiplication		No.of 4-input LUTs	No.of Slices	Delay (ns)
Karatsuba Algorithm With Vedic using Mux/ FA	Modified 2 2x1 MUX	839	477	34.57
	Modified 1 2x1 MUX	854	483	55.32
	4x1 MUX (existing)	819	465	55.86
	Full-Adder (existing)	840	476	54.27

From Table 2, it is noticed that SPFPM using karatsuba algorithm with vedic multiplier with consideration of 4:2 compressor using XOR-MUX provides better results in terms of delay and area.

Table 2. Area and Delay of SPFPM using Compressors

Single Precision Floating Point Multiplication		No.of 4-input LUTs	No.of Slices	Delay (ns)
Karatsuba Algorithm using Vedic with 3:2 and 4:2 compressors	3:2 compressor using XOR-MUX (existing)	919	497	56.99
	3:2 compressor using XOR-XNOR-MUX (existing)	955	540	56.98
	4:2 compressor using XOR-MUX (proposed)	630	322	16.85
	4:2 compressor using XOR-XNOR-MUX (proposed)	642	364	18.61
	Conventional 4:2 (existing)	677	342	23.34

From Table 3, it is illustrated that SPFPM using karatsuba algorithm with vedic multiplier with consideration of 4:2 compressor with XOR-Mux logic provides better results in view of area and delay.

Table 3. Area and Delay Comparison of SP-Floating Point Multiplier with KA using Proposed Models

Single Precision Floating Point Multiplication	No. of 4-input LUTs	No. of Slices	Delay (ns)
KA with Modified 2 2x1 Mux	839	477	34.57
KA with 4:2 compressor with XOR-MUX logic	630	322	16.85
KA with 4:2 Compressor with XNOR-MUX logic	642	364	18.61
Ref [7]	1073	977	16.18
Jain [20]	2270	1269	18.78

7. CONCLUSION

In this paper, floating point multiplication for single precision numbers is developed by using Karatsuba algorithm with vedic multiplier with different existing like full adder, using multiplexers and 4:2, 3:2 compressors and proposed techniques such as modified 2x1 multiplexers and modified 3:2, 4:2 compressors. Further, the performance comparison analysis is done among these techniques in terms of area and delay. From the simulated results, it is inferred that single precision multiplication using karatsuba algorithm with 4:2 Compressor with XOR-MUX logic combination provides better results in terms of area and delay than that of other techniques.

REFERENCES

[1] Banescu, S.; de Dinechin, F.; Pasca, B.; Tudoran, R. Multipliers for floating-point double precision and beyond on FPGAs. *Comput. Archit. News*, 2011, *Vol.* 38, pp.73–79, doi:10.1145/1926367.1926380

[2] Shoba, M.; Nakkeeren, R. Energy and area efficient hierarchy multiplier architecture based on Vedic mathematics and GDI logic, *international journal of Engineering Science and Technology*, 2016, pp. 1-11.

[3] Manish Kumar, J.; Ray, C.C. Cheung. VLSI Implementation of Double-Precision Floating-Point Multiplier Using Karatsuba Technique. *Circuits, Systems and Signal Processing*. 2013, Vol.32, pp. 15–27. doi:10.1007/s00034-012-9457-3.

[4] Hemmert, K.S.; Underwood, K.D. Fast, efficient floating-point adders and multipliers for FPGAs. *ACM Transaction on Reconfigurable Technology and Systems*, 2011, Vol.3, issue 3.

[5] Manish Kumar, J.; Ray, C.C. Cheung. High Performance FPGA Implementation of Double Precision Floating Point Adder/Subtractor. *International Journal of Hybrid Information Technology*, 2011, Vol. 4, No. 4.

[6] Sushma, S. Mahakalkar; Sanjay L. Haridas. Design of High-Performance IEEE 754 Floating Point Multiplier using Vedic mathematics, *Proceedings of International conference on computational Intelligence and Communication Networks*", Brazil, pp.985-988,2014

[7] Arish, S.; Sharma, R.K. A highly efficient floating-point multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm. *Proceedings of international conference on Signal Processing and Communication*, Noida, pp.303-308, 2015.
[8] Irine Padma, B.T.; Suchitra, K. Pipelined Floating Point Multiplier Based on Vedic Multiplication Technique, *International Journal of Innovative Research in Science, Engineering and Technology* (IJIRSET), Vol-3, pp.130-137, July 2014
[9] Josmin, Thomas.; Pushpangadan, R.; Jinesh, S. Comparative Study of Performance Vedic Multiplier on The Basis of Adders Used, *Proceedings of International Conference on Electrical and Computer Engineering*, Bangladesh, vol.2,pp.325-328, Dec 2015.
[10] Karatsuba, A.; Ofman, Y. *Multiplication of many-digital numbers by automatic computers*, in Proceedings of the USSR Academy of Sciences, 1962, vol. 145, pp. 293–294.
[11] Kim, C.H.; Kwon, S.; Hong, C.P. FPGA implementation of high performance elliptic curve cryptographic processor over GF(2^{163}). *Journal of System Architecture*, 2008, 54(10), 893–900. doi:10.1016/j.sysarc. 2008.03.005
[12] Koohi, A.; Bagherzadeh, N.; Pan, C. A fast parallel Reed–Solomon decoder on a reconfigurable architecture, *in First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2003, pp. 59–64.
[13] Lienhart, G.; Kugel, A.; Manner, R. Using floating-point arithmetic on FPGAs to accelerate scientific n-body simulations, *in 10th Annual IEEE Symposium on Field-Programable Custom Computing Machines, 2002.*
[14] Paschalakis, S.; Lee, P. Double precision floating-point arithmetic on FPGAs, *in 2nd IEEE International Conference on Field Programmable Technology*, 2003, pp. 352–358.
[15] Smith, M., Vetter, J., Liang, X. Accelerating scientific applications with the SRC-6 reconfigurable computer: methodologies and analysis. *in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005, p. 157b
[16] Venishetti, S.; Akoglu, A. A highly parallel FPGA based IEEE-754 compliant double-precision binary floating-point multiplication algorithm, *in International Conference on Field-Programmable Technology*, 2007, pp. 145–152. doi:10.1109/FPT.2007.4439243
[17] Wang, X., Leeser; M., Float V. a variable precision fixed and floating-point library for reconfigurable hardware. *ACM Trans. Reconfigurable Technol. Syst*. 2010,3, 16. doi:http://doi.acm.org/10.1145/1839480.1839486
[18] Xilinx, Xilinx floating-point IP core. http://www.xilinx.com
[19] Priyamka, K.; Sreenivasu, T.; Purna Ramesh, A. Asynchronous advanced single precision floating point multiplier using Verilog HDL., *international journal of Advanced Engineering Research in Electronics and Communication Engineering*, pp.885-887, 2014.
[20] Jain, A.; Dash, B.; Panda, A.K. FPGA design of a 32-bit multipliers unit, *Proceedings of international conference on Devices, Circuits and Systems*, pp.545-547, 2012.