Volume 4, Issue 7, July 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Design of Double Precision Floating Point Multiplier by Using Vedic Multiplication

Pranal D. Kale
Dept. of Electronics & Telecomm.
B.D.C.O.E, Wardha, Sevagram, India

Prof.M. N. Thakre
Dept. of Electronics & Telecomm.
B.D.C.O.E, Wardha, Sevagram, India

Prof. Mrs. R. N. Mandavgane
Dept. of Electronics & Telecomm.
B.D.C.O.E, Wardha, Sevagram, India

Abstract*—Mainly computers use binary numbers. Area of graph theory, multidimensional graphics, and digital signal processing, high performance computing etc essentially need Floating point number's multiplication and it would like more precision however, it is found that binary numbers should be precise enough for most scientific and engineering calculations. Floating point multiplication is also applicable in many commercial applications like financial analysis, banking, tax calculation, currency conversion, insurance, and accounting. High speed floating point (FP) multipliers are essential for high speed calculation systems because increasingly large numbers of FP multiplications must be carried out in various applications. In this paper design of 64 bit floating point multiplier by using Vedic multiplication is presented and compared it with the design of 64 bit floating point multiplier by using conventional method of multiplication. Among the various methods of multiplication in Vedic mathematics, Urdhava Tiryakbhyam is used and discussed in detail. Urdhava Tiryakbhyam is the multiplication technique which is used to all types of multiplication. The coding is done in VHDL (very high speed integrated circuit hardware description language) and synthesis is done using Xilinx ISE series.

Keywords*— Floating point multiplier, IEEE 754 standard, Urdhava Tiryakbhyam, convectional method, Vedic Mathematics.

I. INTRODUCTION

Modern Applications Such As 3D Graphics Accelerators, Digital Signal Processors (Dsps), High Performance Computing Etc Usually Involve Floating Point Calculations With Double Precision Format. The Growing Computational Demands Of Scientific Applications Shows That In Many Cases There Is A Need For Increased Precision In Floating Point Calculations. Examples Are The Fields Of Computational Physics, Computational Geometry, Climate Modeling Etc., Which Require High Precision Calculations And Great Accuracy.

The IEEE 754 standard provides the format for representation of binary floating point numbers [7]. The Binary Floating point numbers are represented in Single and Double formats. The Single consist of 32 bits and the Double consist of 64 bits. The Fig1 shows the structure of Single and Fig2 shows the structure Double formats of IEEE 754 standard. The formats are composed of 3 fields; Sign, Exponent and Mantissa. For single precession format, the Mantissa is represented in 23 bits and 1bit is added to the MSB for normalization, Exponent is represented in 8 bits which is biased to 127 and MSB of Single is reserved for Sign bit. For Double precision format, the Mantissa is represented in 52 bits, the Exponent is represented in 11 bits which is biased to 1023 and the MSB of Double is reserved for sign bit. For both When the sign bit is 1 that means the number is negative and when the sign bit is 0 that means the number is positive.

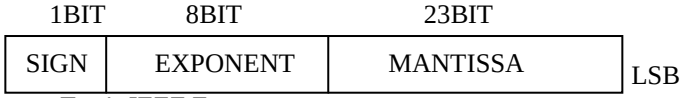

FIG1- IEEE FORMAT FOR SINGLE PRECISION

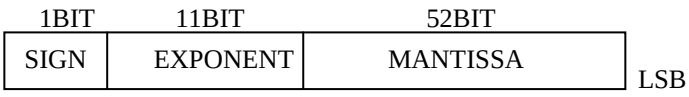

Fig2-IEEE Format for double precision

The architectures for multipliers are mainly Array, Vedic multipliers and booth's. Design of the mantissa multiplier by using proper technique can enhance the performance of the Floating Point Multiplier. This unit requires unsigned multiplier for multiplication of BITs. The Vedic Multiplication technique can be beneficial for the designing of this unit. The beauty of Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a very simple one. As, the Vedic multiplication technique are generally based on natural human mind working process. Vedic mathematics is mainly based on 16 Sutras. Out of these 16 Vedic Sutras the Urdhva-triyakbhyam sutra is used for multiplication purpose. Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and crosswise". It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products. The parallelism in generation of partial products and their summation is obtained using Urdhva Triyakbhyam .

The algorithm can be generalized for n x n bit number. Since the partial products and their sums are calculated in parallel, the multiplier is independent of the clock frequency of the processor. Thus the multiplier will require the same amount of time to calculate the product and hence is independent of the clock frequency.

II. LITERATURE REVIEW

In earlier work array multiplication technique was used where two binary numbers A and B, of 'm' and 'n' bits. There are 'mn' summands that are produced in parallel by a set of 'mn' AND gates and booth's multiplication techniques was used for multiplication where two signed binary numbers multiplied in two's complement notation. The algorithm was invented by Andrew Donald Booth. As compared to array multiplication booth's multiplication is faster. In array multiplier for n x n multiplier requires n(n-2) full adders, n half-adders and n2 AND gates. Also, in array multiplier worst case delay would be (2n+1) t_d. Booth's floating point multiplier is faster than the array multiplier, by calculating the delay value we can get that power dissipation is also less compare to array multiplier. In BOOTH multiplication algorithm to reduce the time needed for multiplication number of partial products to be added are reduced. BOOTH recording reduces the number of adder units needed and hence reduced the delay by reducing number of nonzero bits in the multiplier [13]. In BOOTH recoding, the long sequence of 1s is replaced by two no zero bits; for example, If digits *j* through (down to) *k* are 1s, then,

$$2j + 2j+1 + \ldots + 2k+1 + 2k = 2j+1 - 2k$$

This represents, the sequence of additions can be replaced by an addition of the multiplicand shifted by *j*+1 positions and a subtraction of the multiplicand shifted by *k* positions [12]. The drawback of BOOTH recording is the high power consumption and thus reduced efficiency [8]. The multiplier implementation in floating point multiplication is done by Modified Booth Encoding (MBE) multiplier to reduce the partial products by half. The multiplier takes care of overflow and underflow cases. Rounding is not to give more precision when using the multiplier implemented in a multiply and Accumulate (MAC) unit. By using MBE multiplier we increases the speed of multiplication, reduces the power dissipation and cost of a system. The proposed multiplier will be designed and verified using ModelSim with Verilog HDL. Xilinx is used for synthesis. This paper presents an implementation of a floating point multiplier that supports the IEEE 754 binary interchange format; one of the important aspects of the presented design method is that it can be applicable to all kinds of floating point multipliers. The present design is compared with an ordinary floating point array multiplier and modified Booth encoder multiplier via synthesis. It shows that Booth's floating point multiplier is faster than the array multiplier, by seeing the delay value we can known this factor and power [9].

Table1- comparison of multipliers

Design (single precision)	Number of slices	Adders/ Subtractor	4 input LUTs	Bonded IOBs	Delay
Array Multiplier	630	32	1147	96	96.08ns
Booth Multiplier	1582	26	2781	97	36.97ns

Implement multiplier for both conventional, as well as Vedic mathematical methods in VHDL language and highlight a comparative study of both approaches in terms of gate delays. The functional verification through simulation of the VHDL code was carried out using ModelSim SE 6.0 simulator. The synthesis is done using Xilinx Synthesis Tool (XST) available with Xilinx ISE 9.1i. The design is optimized for speed and area using Xilinx, device family Spartan3. In this paper, it is observed that 86.71% lesser slice and also around 88% lesser four input look-up are utilized for Vedic multiplier compared to other multipliers. It shows that 8 bit Vedic multiplier achieves higher speed by reducing gate delay by factor of 24% compared to array multiplier and around 18.2% compared to booth multiplier. Similarly, 16 bit Vedic multiplier achieves higher speed by reducing gate delay by factor of 39.9% compared to array multiplier and around 48.36% compared to booth multiplier[1].

Table2- Design Summary

Name of the Multiplier (16 bit)	Number of slices	No of IOs	4 input LUTs	Bonded IOBs	Delay
Array multiplier	290 out of 768	65	505 out of 1536	64 out of 124	70.928 ns
Booth multiplier	499 out of 768	65	923 out of 1536	65 out of 124	60.809 ns
Vedic multiplier	120 out of 768	90	240 out of 1536	90 out of 124	36.563 ns

Vedic Multiplication Technique is used to implement IEEE 754 single precision (32 bits) Floating point multiplier. The Urdhvatri- yakbhyam sutra is used for the multiplication of Mantissa. The underflow and over flow cases are handled. The inputs to the multiplier are provided in IEEE 754, 32 bit format. The multiplier is implemented in VHDL and Virtex-5 FPGA is used. A test bench is used to generate the stimulus and the multiplier operation is verified. The over flow and under flow flags are incorporated in the design in order to show the overflow and under flow cases The paper shows the efficient use of Vedic multiplication method in order to multiply two floating point numbers. The lesser number of LUTs verifies that the hardware requirement is reduced, thereby reducing the power consumption. The power is reduced affectively still not compromising delay so much [3].

III. DESIGIN METHOD FOR DOUBLE PRECISION FLOTING POINT MULTIPLIER

IEEE-754 standards give the floating point number representation as:

V= (-1) sign * 2exponent-bias*1.fraction [1]

Implicit bit is used before fraction or mantissa, which is „1 for normalized number and, 0 for un-normalized number. Exponent bias is (2e-1) , which comes out to be 127 for single precision and 1023 for double precision exponent. Floating point multiplication is not as simple as integer multiplication. Designing of a floating point multiplier of floating point numbers represented in IEEE 754 format can be divided in different units:

Mantissa Calculation Unit

Exponent Calculation Unit

Sign Calculation Unit

Initially the inputs with IEEE754 format will be unpacked and will be assigned to the check sign, add exponent and multiply mantissa.

The product is positive when the two operands have the same sign; otherwise it is negative. Sign of the result is calculated by XORing sign bits of both the operands A and B.

Exponents of two multiplying numbers will be added to get the resultant exponent. Addition of exponent will be using 16 bits adder. Exponents will be expressed in done 1023 bit.

The Mantissa Calculation Unit requires a 52 bit multiple. This unit requires unsigned multiplier for multiplication of 54*54 BITs. There are number of techniques that can be used to performance of mantissa calculation Unit .Mantissa calculation Unit dominates overall performance of the Floating Point Multiplier. This unit requires unsigned multiplier for multiplication of BITs.

So for 54*54 multiplier we first create 3*3 multiplier by using this 3*3, we create 9*9 multiplier by using this 9*9, we create 27*27 multiplier and at last by using this 27*27, we create 54* 54 multiplier.

Below Figure4. Shows the line diagram for 3 bit numbers multiplication taking place by using the Urdhva-triyakbhyam sutra with detailed execution process of sutra.

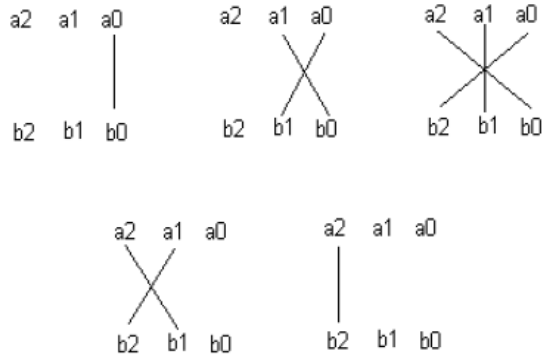

Fig-4: Line diagram for multiplication of two 3 – bit Numbers

Consider the numbers A and B where A = a2a1a0 and B = b2b1b0. The LSB of A is multiplied with the LSB of B:

s0=a0b0;

Then a0 is multiplied with b1, and b0 is multiplied with a1 and the result are added together as:

c1s1=a1b0+a0b1;

Here c1 is carry and s1 is sum. Next step is to add c1 with the multiplication results of a0 with b2, a1 with b1 and a2 with b0.

c2s2=c1+a2b0+a1b1 + a0b2;

Next step is to add c2 with the multiplication results of a1 With b2 and a2 with b1.

c3s3=c2+a1b2+a2b1;

Similarly the last step

c4s4=c3+a2b2;

Now the final result of multiplication of A and B is

c4s4s3s2s1s0.

By using this technique we design the 3*3 bit multiplier From that 9*9 multiplier and by using this 9*9 we design the 27*27 at last by the use of this 27*27 we design the 54*54 mantissa multiplier. Below

figure shows the block Diagram of 54*54 bit mantissa multiplier.

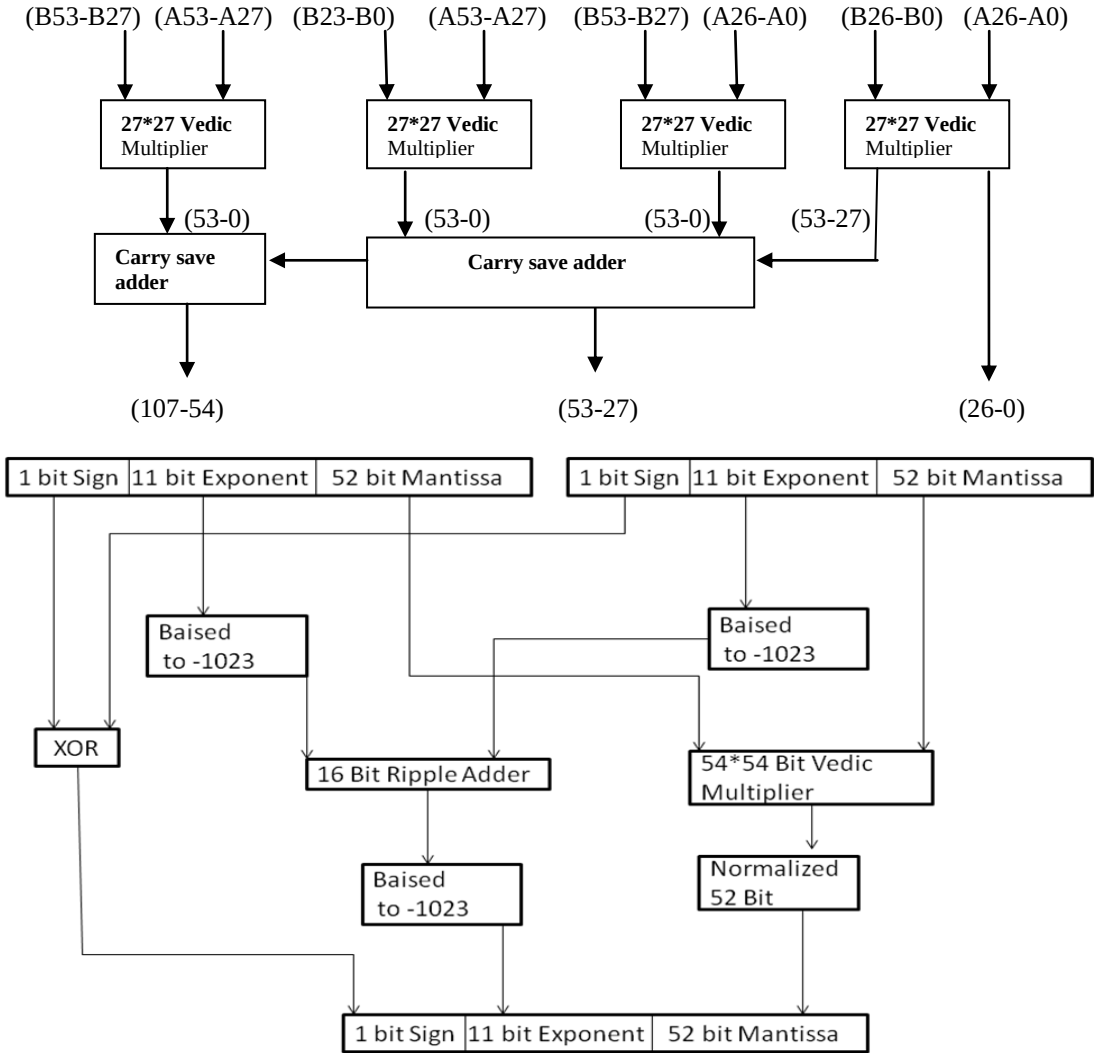

Fig-3: Architecture for 64 Floating point multiplier

Consider the one example:

suppose A and B are 2 floating-point numbers, where

A= −13.5 and B=2.0

Binary form of the numbers A and B:

A = -1101.1 B = + 10.0

Shifting decimal point and representing in "1." from

A = −1 · 1011×23 B = +1 · 00×21

IEEE 754 double precision representation of the numbers:

S E M

A=1 10000000010 1011000000000000000000000000000000 00000000000000000000

S E M

B= 0 10000000000 00000000000000000000000000000000 00000000000000000000

Now first Multiplication of the mantissa and result is

10110000000000000000000000000000000000000000000000 0000000000000000000

From this resulted 108 bit mantissa only the most significant bits 52 bits are considered we get the 52-bit mantissa of the result. This normalization can lead to correction of the result's exponent

10110000000000000000000000000000000000000000000000000000

Now for resultant exponent the sum of the operands exponents.

the biased exponent fields (Ea and Eb) are made unbiased in order to do the addition. And then, we must to add again to the exponent the bias, to get the exponent field of the result (Er):

Er = (Ea− 1023) + (Eb −1023) + 1023 = Ea + Eb – 1023

we have:

Ea =10000000010 Eb = 10000000000

Er = 10000000010 + 10000000000 -1023

Er= 100000000010-1023

Er= 2050-1023=1027

Er =10000000011, the resulted exponent

At last Calculation of the sign of the result:
sign of the result (Sr) is given by the exclusive-or of the numbers signs (Sa and Sb)

Sr = Sa xor Sb

we have

Sr =1 xor 0 =1

i.e a negative sign

Composition of the 3 intermediate results

sign(Sr)	Exponent(Er)	Mantissa
1	10000000011	10110000000000000000000 00000000000000000000000000000000

Decimal representation of the resultant valve:

As sign bit 1 therefore result is negative (-)

Exponent (Er)= 10000000011=1027

Therefore result is= -1. 1011000*2e1027-1023

-1. 10110000*2e4 so shift decimal point 4 digits in right -11011

i.e. -27 (-13.5*2.0= -27)

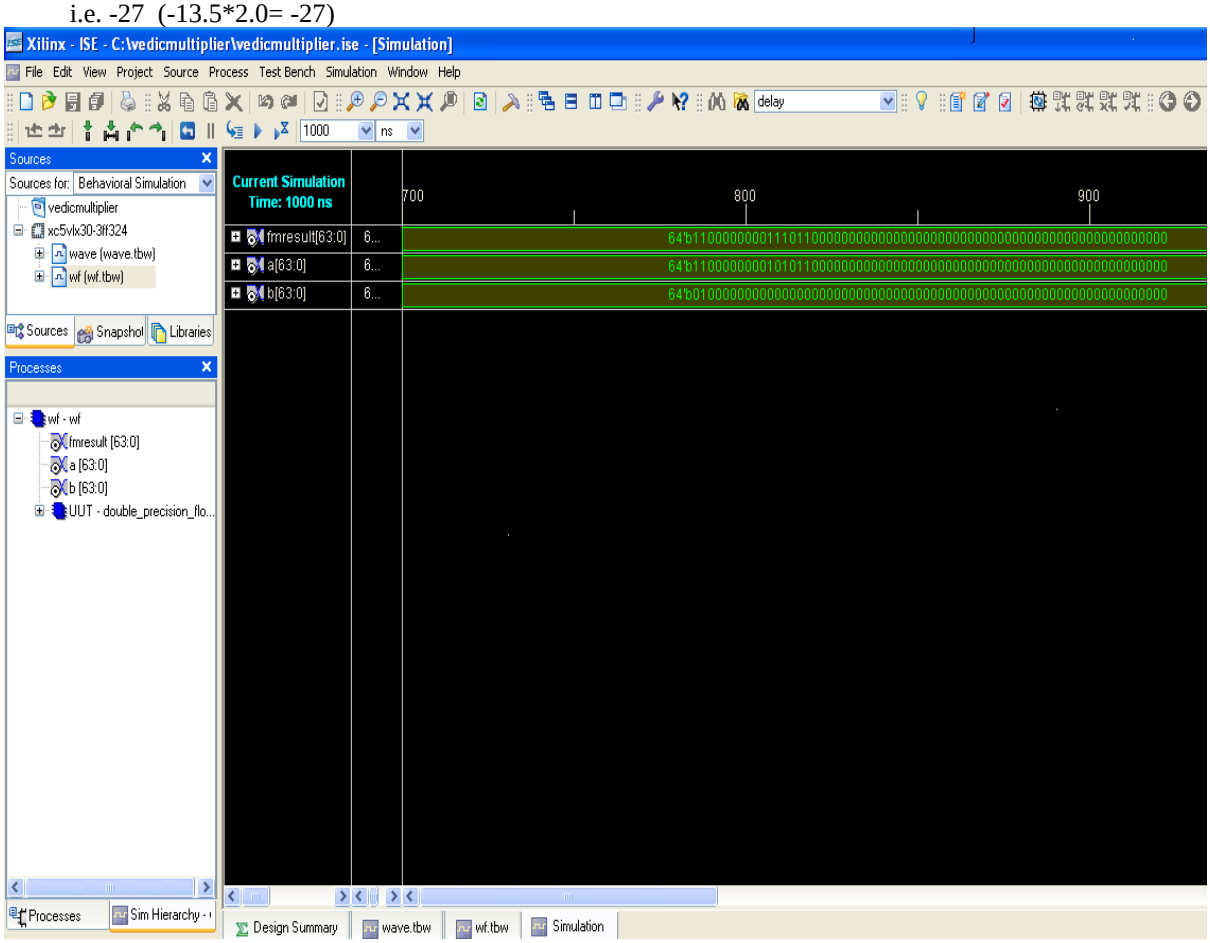

Fig-4: waveforms of double precision floating point multiplier.

Table-2: Design Summary

Parameter	Double precision floating point multiplier by using Vedic multiplication method	Double precision floating point multiplier by using convectional multiplication method
Device	**Virtex 5**	**Virtex 5**
Number of LUT's	**6788**	**10988**
Number of IOBs	**192**	**192**
Time Delay	**44.565ns**	**326.756ns**

IV. CONCLUSIONS

From above table Double precision floating point multiplier by using Vedic multiplication method is better than array multiplication method. The Vedic multipliers are much faster than the conventional multipliers. This gives us method for hierarchical multiplier design. So the design complexity gets reduced for inputs of large no of bits and modularity gets increased. The growing computational demands of scientific applications shows that there is a need for increased precision in floating point calculations like 64 bits and always has scope for improving the speed with the fast multiplication technique by reducing time delay. The paper shows the efficient use of Vedic multiplication method in order to multiply two floating point numbers .The double precision floating point multiplier is designed in VHDL and simulated using Simulator. The design was synthesized using Xilinx ISE 9.1 tool. The Vedic multipliers are much faster than the conventional.

ACKNOWLEDGEMENT

The authors like to wish thanks' to all the supportive teaching staff and faculty members, and reference authors who guided by their papers.

REFERENCES

[1] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur, Girish V A, "Implementation of Vedic Multiplier for Digital Signal Processing," International Journal of Computer Applications (IJCA) 2011.

[2] Al-Ashrafy, M.; Salem, A.; Anis, "An efficient implementation of floating point multiplier," Electronics Communications and Photonics Conference (SIECPC), 2011

[3] Aniruddha Kanhe, Shishir Kumar Das, Ankit Kumar Singh, "Design and Implementation of Floating Point Multiplier based on Vedic Multiplication Technique" 2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, Indi.

[4] Kavita Khare, R.P.Singh, Nilay Khare,"Comparison of pipelined IEEE-754 standard floating point multiplier with unpipelined multiplier" Journal of Scientific & Industrial Research Vol.65, pages 900-904 November 2006.

[5] Manish Kumar Jaiswal, Nitin Chandrachoodan "Efficient Implementation of IEEE Double Precision Floating-Point Multiplier on FPGA" 2008 IEEE Region 10 Colloquium and the Third ICIIS, Kharagpur, INDIA. December 8-10.

[6] B. Lee and N. Burgess, "Parameterisable Floating-point Operations on FPGA," Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems, and Computers, 2002.

[7] Xilinx Floating-Point v2.0. [Online]. Available: http://www.xilinx.xom

[8] Gokul Govindu, L. Zhuo, S. Choi, V. Prasanna, " Analysis of High performance Floating-point Arithmetic on FPGAs", Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS '04), pages 149-156, April-2004

[9] P.V.Krishna Mohan Gupta, Ch.S.V.Maruthi Rao, G.R. Padmini, "An Efficient Implementation of High Speed Modified Booth Encoder for Floating Point Signed & Unsigned Numbers". International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 8, August - 2013

[10] P. Saha, A. Banerjee, A. Dandapat, P. Bhattacharyya, "Vedic Mathe matics Based 32-Bit Multiplier Design for High Speed Low Power Processors" International Journal On Smart Sensing And Intelligent Systems Vol. 4, No. 2, June 2011

[11] G.Vaithiyanathan, K.Venkatesan, S.Sivaramakrishnan, S.Sivaand S. Jayakumar "Simulation And Implementation Of Vedic Multiplier Using Vhdl Code" International Journal of Scientific & Engineering Research Volume 4, Issue 1, January-2013 ISSN 2229-5518.

[12] Himanshu Thapliyal, "Modified Montgomery Modular Multiplication using 4:2 Compressor and CSA Adder", Proceedings of the third IEEE international workshop on electronic design, test and applications (DELTA 06), Jan 2005.

[13] E.M.Saad, M.Taher, "High speed area efficient FPGA based floating point arithmetic modules", National conference on radio science (NRSC 2007), March-2007, pp 1-8. doi:10.1109/DELTA.2008.19

