International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 3, Issue 3, pp: (162-169), Month: July - September 2015, Available at: www.researchpublish.com

Design of Double Precision Floating Point Multiplier Using Vedic Multiplication

[1]D.Heena Tabassum, [2]K.Sreenivas Rao

[1, 2] Electronics and Communication Engineering,
[1, 2] Annamacharya institute of technology and sciences, Rajampet, Andhra Pradesh, India

Abstract: **The real numbers represented in binary format are known as floating point numbers. Based on IEEE-754 standard, floating point formats are classified into single precision and double precision. The advantage of floating-point representation over fixed point representation is that it can support a much wider range of values Floating point multipliers are key components of many high performance systems such as FIR filters, Microprocessors, Digital Signal Processors etc. Hence, this project aims to design a 64-bit double precision floating point multiplier by using vedic multiplication. Vedic multiplier is a special kind of multiplier which is capable to multiply more number of bits at a time, at faster rate than conventional multiplier. The proposed multiplier can be designed using Verilog HDL and it is implemented on a Xilinx ISE 14.3 tool targeting the Spartan device.**

Keywords: **Single Precision, Double Precision, Field Programmable Gate Array, Multiplier, Vedic multiplier, Urdhava Triyagbhyam.**

I. INTRODUCTION

Floating point number system is a common choice for many scientific computations due to its wide dynamic range feature. For instance, floating point arithmetic is widely used in many areas, especially in scientific computation, numerical processing, image processing and signal processing. The term floating point is derived from the fact that there is no fixed number of digits before and after the decimal point, that is, the decimal point or binary point can float. There are also representations in which the number of digits before and after the decimal or binary point is fixed; called fixed-point representations. The advantage of floating-point representation over fixed point representation is that it can support a much wider range of values. The floating point numbers is based on scientific notation. A scientific notation is just another way to represent very large or very small numbers in a compact form such that they can be easily used for computations. The floating point multiplication operations are greatly affected by how the floating point multiplier is designed. Floating point number consists of three fields:

1. Sign (S): It used to denote the sign of the number i.e. 0 represent positive number and 1 represent negative number.

2. Significand or Mantissa (M): Mantissa is part of a floating point number which represents the magnitude of the number.

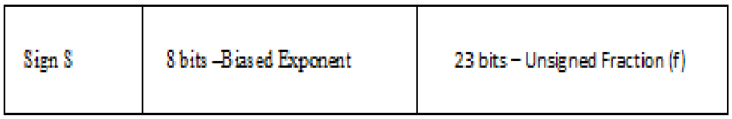
3. Exponent (E): Exponent is part of the floating point number that represents the number of places that the decimal point (binary point) is to be moved.

Number system is completely specified by specifying a suitable base β, significand (mantissa) M, and exponent E. A floating point number F has the value

$$F=(-1)^{S} M \beta^{E} \quad \text{..... (1)}$$

The way floating point operations are executed depends on the data format of the operands. IEEE standards specify a set of floating point data formats, single precision and double precision. The Single precision consists of 32 bits and the Double precision consists of 64 bits. Figure 1 shows the IEEE single and double precision data formats.

(a)IEEE single precision data format



(b) IEEE double precision data format

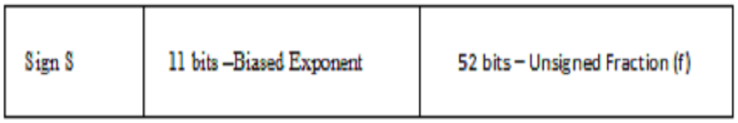


Figure 1: IEEE Single and Double Precision data Format

The value of the floating point number represented in Single precision format is

$$F=(-1)^{S}\,1.f\,2^{E-127} \quad (2)$$

Where 127 is the value of bias in single precision data Format and exponent E ranges between 1 to 254, and E = 0 and E = 255 are reserved for special values.

The value of the floating point number represented in double precision data format is

$$F=(-1)^{S}\,1.f\,2^{E-1023} \quad (3)$$

Where 1023 is the value of bias in double precision data format Exponent E ranges between 1 to 2046, the values of E = 0 and E = 2047 are reserved for special values.

II. DOUBLE PRECISION FLOATING POINT MULTIPLIER

Multipliers are key components of many high performance systems such as FIR filters, Microprocessors, Digital Signal Processors etc. A system's performance is generally determined by the performance of the multiplier, because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a critical issue for an effective system design.

A. Floating point multiplication algorithm:

Multiplication of floating point numbers F1 and F2 is a five step process. To multiply two floating point numbers the following is done:

- Obtaining the sign; i.e. S1 xor S2.
- Adding the exponents; i.e. (E1 + E2 – *Bias*).
- Multiplying the significand; i.e. (1.M1*1.M2).
- Placing the decimal point in the result
- Normalizing the result; i.e. obtaining 1 at the MSB of the results‟ significand.
- Rounding the result to fit in the available bits.
- checking for underflow/overflow occurrence.

B. Floating point Multiplier design:

The multiplier for the floating point numbers represented in IEEE 754 format can be divided into three different units: Mantissa Calculation unit, Exponent Calculation unit and Sign Calculation unit.The Multiplier receives two 64-bit floating point numbers. First these numbers are unpacked by separating the numbers into sign, exponent, and mantissa

bits. The floating point multiplication is carried out in following three parts. Sign calculation unit:- In this unit, we determine the sign of the product by performing a XOR operation on the sign bits of the two operands. Exponent calculation unit:- This unsigned adder is used for adding the exponent of the first input to the exponent of the second input and subtracting the Bias (1023) from the addition result.

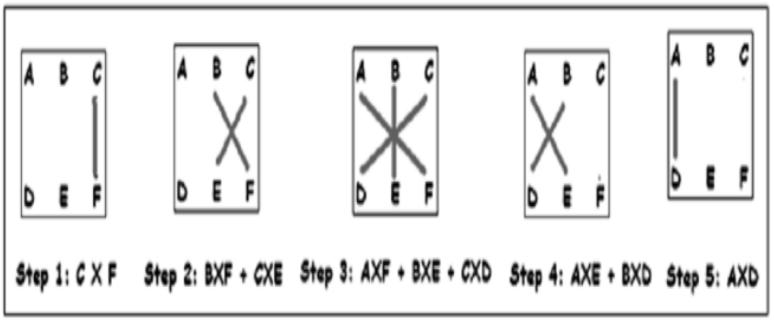
Eresult = A_exponent + B_exponent – Bias

The exponent of the result must be 11 bits in size, and must be between 1 and 2046 otherwise the value is not a normalized one. Overflow/underflow means that the result‟s exponent is too large/small. Mantissa Calculation unit:- The multiplication is done in two steps, partial product generation and partial product addition. For double precision operands (53-bit fraction fields), a total of 53*53-bit multiplier is required. Mantissa multiplier unit performs multiplication operation. After this the output of mantissa is normalized, i.e. if the MSB of the result obtained is not 1, then it is left shifted to make the MSB 1. If changes are made by shifting then corresponding changes has to be made in exponent also. The mantissa of operand A and the leading „1‟ (for normalized numbers) are stored in the 53-bit register (Ma) The mantissa of operand B and the leading „1‟ (for normalized numbers) are stored in the 53-bit register (Mb) Multiplying all 53 bits of Ma by 53 bits of Mb would result in a 106-bit product. Rounding is used to fit the result in available bit then output of mantissa multiplication is 56 bits.

III. VEDIC MULTIPLIER

The “Urdhva Tiryagbhyam” Sutra is a general multiplication formula applicable to all cases of multiplication such as binary, hex, decimal and octal.

The Sanskrit word “Urdhva” means “Vertically” and “Tiryagbhyam‟ means “crosswise”.



Consider ABC as multiplicand and DEF as the multiplier. The steps of multiplication are descriptive in the figure above and the examples are solved below for better understanding. The intermediate carry generated is appended to the very next bit.

A. Illustration:

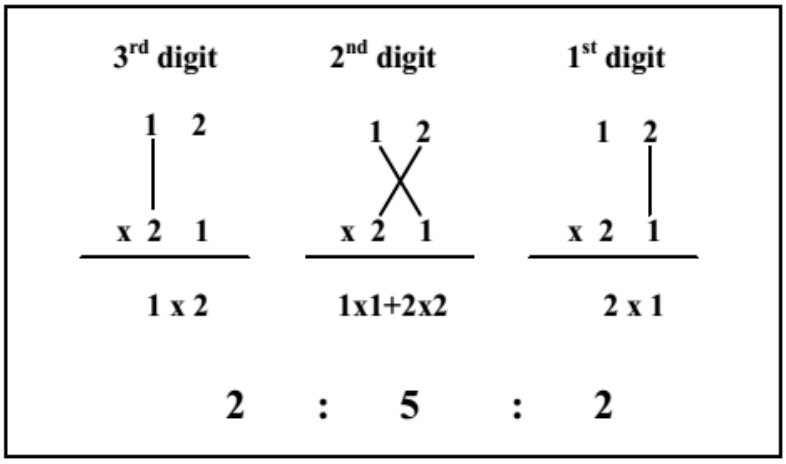


Figure 2: Illustration of decimal multiplication using Vedic technique

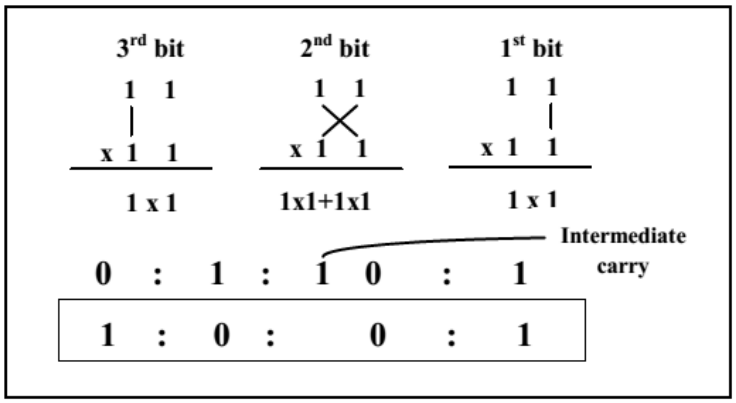


Figure 3: Illustration of binary multiplication using Vedic technique

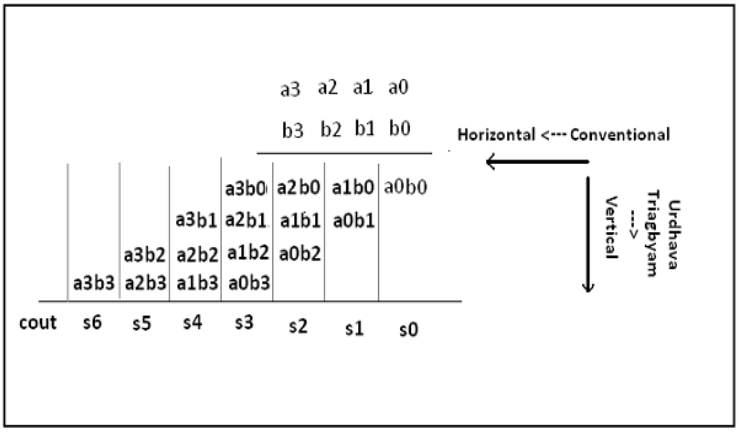


Figure 4: Difference between Conventional Multiplication and Vedic technique

In conventional method, partial products are summated only after every partial product is obtained. Whereas, in Vedic technique, partial products are obtained vertically as shown in the figure above and simultaneously once all the elements of a column are obtained, respective partial products are added. Hence, leads to advancement in speed over the conventional method.

B. Proposed Multiplier:

Proposed multiplier architecture of 2x2, 4x4, and 8x8 bit VM module are displayed below. The basic architecture was comprehended from the base paper and modified to obtain the right output as well as gain speed. The major change adopted here in the architecture is that we have used Kogge stone algorithm to add partial products rather than RCA, CLA and CSA.

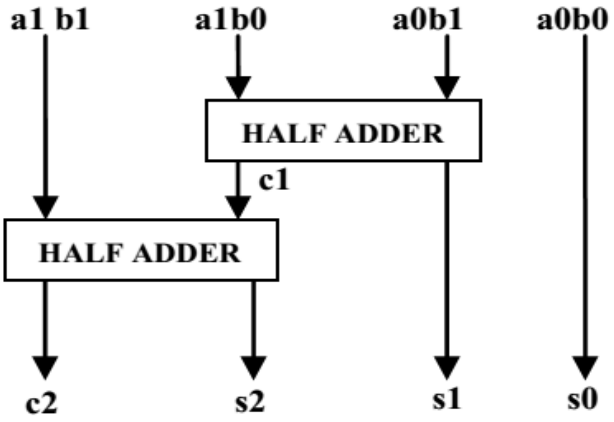


Figure 5: 2x2 Vedic Multiplier

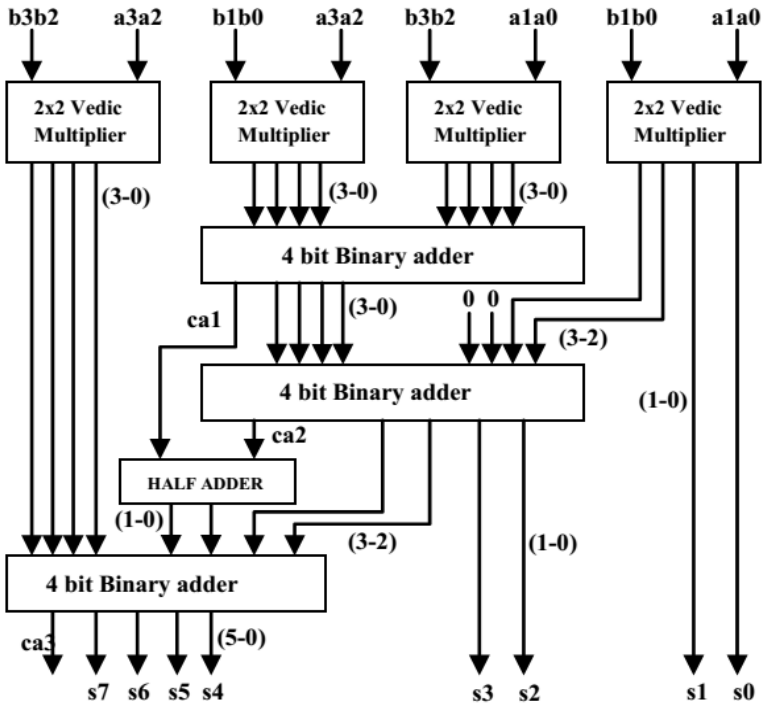


Figure 6: 4x4 Vedic Multiplier

Each block as shown above is 2x2 bits multiplier. First 2x2 multiplier inputs are a1 a0 and b1 b0.The last block is 2x2 multiplier with inputs a3 a2 and b3 b2. The middle one shows two, 2x2 bits multiplier with inputs a3a2 & b1b0 and a1a0 & b3b2. So the final result of multiplication, which is of 8 bit, s7s6s5s4s3s2s1s0.

IV. PROPOSED ARCHITECTURE OF FLOATING POINT MULTIPLIER

The proposed architecture for Double Precision Floating Point Multiplier using Vedic Multiplier is shown in figure

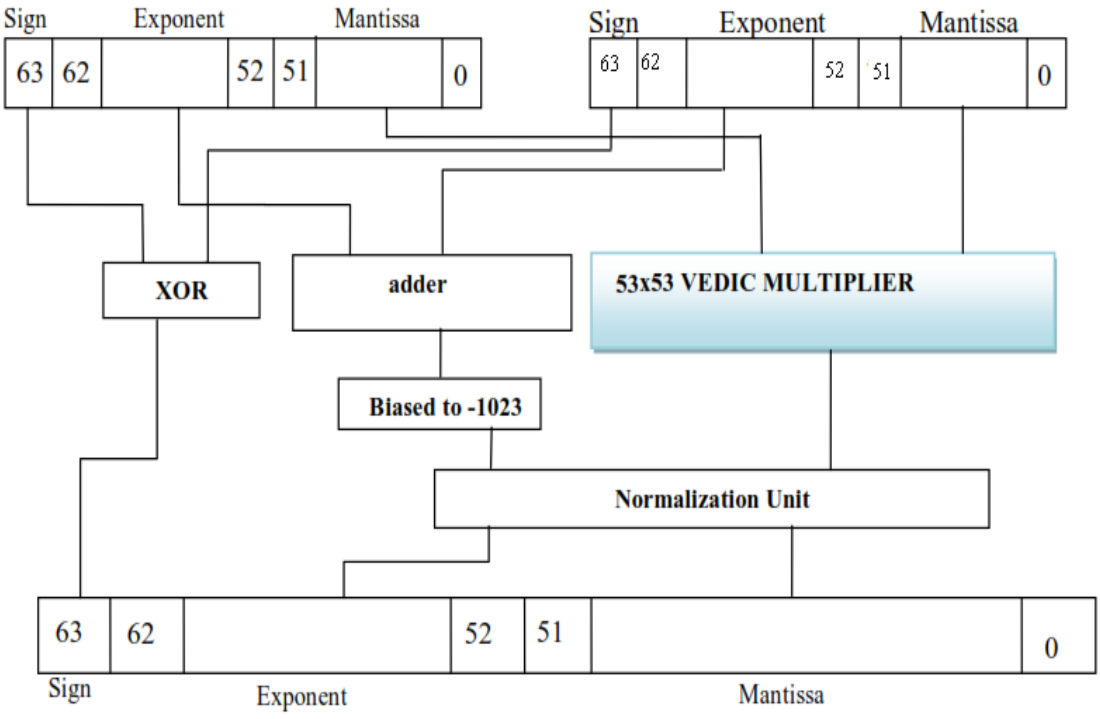


Figure 7: Proposed Architecture of Double Precision Floating Point Multiplier

From the calculation perspective whole floating point multiplication is divided into four sections.

A. Sign section

B. Exponent section

C. Mantissa section

D. Normalization Section

A. Sign output:

Calculation of the sign of the result: Sign = Sa ⊕ Sb The sign of the result is given by the XOR of the operands signs (Sa and Sb).

B. *Exponent output:*

Calculation of the exponent field of the result: Er (exponent_5) = (Ea-1023) + (Eb-1023) + 1023 =Ea + Eb– 1023

C. *Mantissa output:*

Multiplication of the mantissas: we must extract the mantissas, adding a 1 as most significant bit, for normalization. Ma and Mb is 53 bit i.e. Mr = Ma*Mb (53*53)

D. Normalization Section:

In the Normalization section, normalization of exponent and mantissa are performed. According to the 105^{th} bit (result of the 53x53 bit binary multiplier) normalization is done.

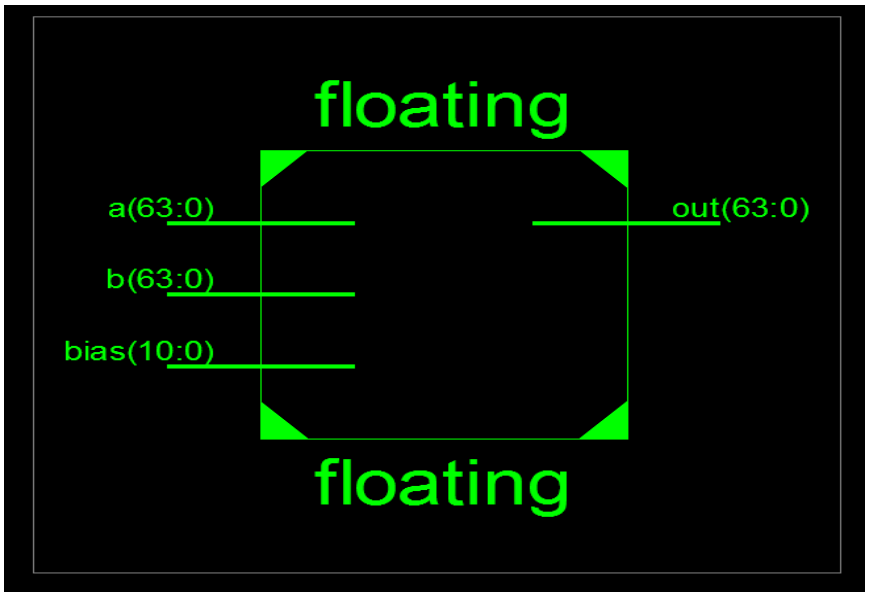
i. When 105^{th} bit of 53X53 bit binary multiplier is binary one ,mantissa is normalized to 52 bit by taking 104^{th} to 53^{th} bit position number and exponent is increased by decimal value one.

ii. When 105^{th} bit of 53X53 bit binary multiplier is binary zero, mantissa is normalized to 52 bit by taking 103^{th} to 52^{th} bit position number and there is no increment in the exponent value.

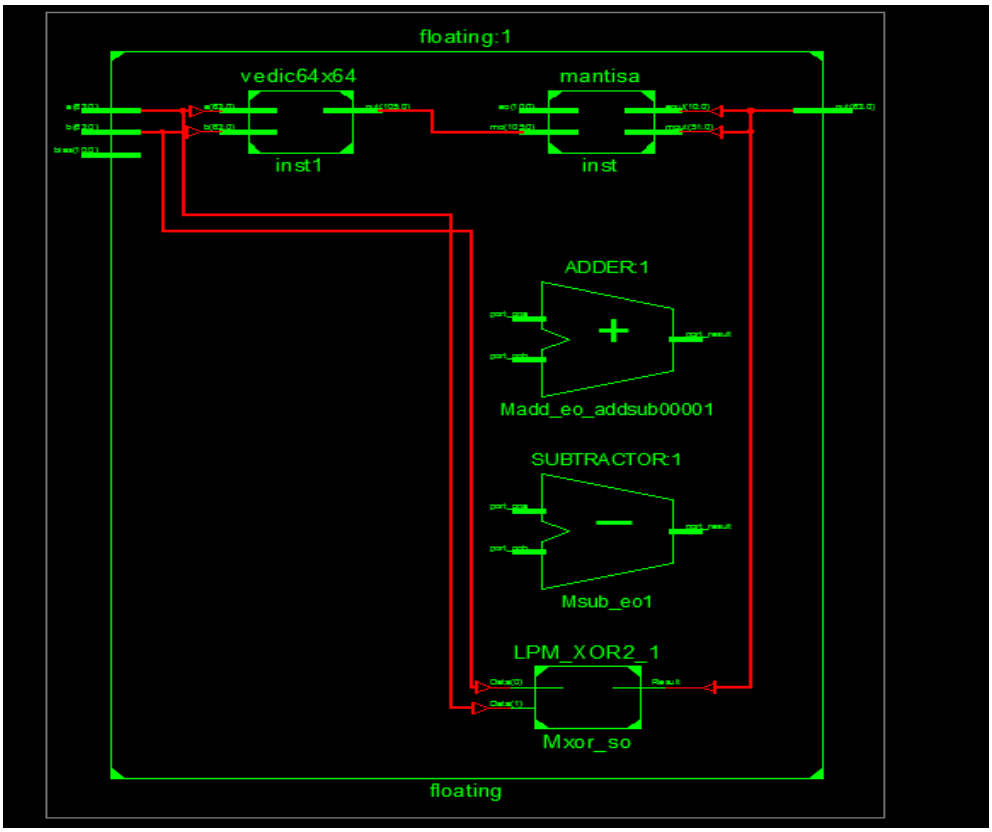
V. ANALYSIS OF RESULT

The FPGA-based implementations of the double precision floating point multiplier have been presented in this section. For implementation purpose, Xilinx Integrated Software Environment ISE 14.3 software tool has been used. The double precision floating point multiplier component has been coded in Verilog HDL has been synthesized and simulated using Xilinx ISE 14.3 which are mapped on to Spartan 3E FPGA. The RTL view, design summary and simulation result of the floating point multiplier are shown in following section.

Block diagram:



RTL schematic:



vice Utilization Summary (estimated values)	[-]

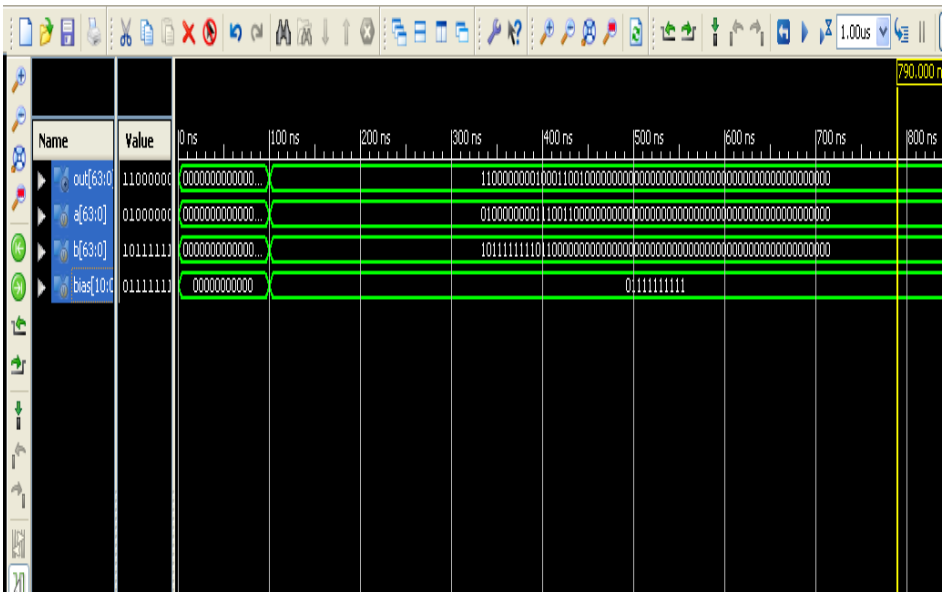
Device utilization summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	4704	14752	31%
Number of 4 input LUTs	8616	29504	29%
Number of bonded IOBs	203	250	81%
Delay	28.731ns		

Extension Device utilization summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	291	14752	1%
Number of 4 input LUTs	561	29504	1%
Number of bonded IOBs	203	250	81%
delay	23.488ns		

Simulation results of floating point multiplier:



VI. CONCLUSION

In this paper, double precision floating point multiplier based on the IEEE-754 format is successfully implemented on FPGA. The modules are written in Verilog HDL to optimize implementation on FPGA. In this implementation they obtained maximum frequency that of the multiplier implemented by using Vedic algorithm. Since the main idea behind this implementation is to increase the speed of the multiplier by reducing delay at every stage using the optimal Vedic multiplier design, it gives the advantage of less delay in comparison to other method. The results obtained using the proposed algorithm and implementation is better not only in terms of speed but also in terms of hardware used.

REFERENCES

[1] Purna Ramesh Addanki,Venkata Nagaratna Tilak Alapati and Mallikarjuna Prasad Avana, "An FPGA Based High Speed IEEE – 754 Double Precision Floating Point Adder/Subtractor and Multiplier Using Verilog", International Journal of Advanced Science and Technology, Vol. 52, pp.61 -74, March 2013.

[2] Riya Saini and R.D.Daruwala,"Efficient Implementation of Pipelined Double Precision Floating Point Multiplier",International Journal of Engineering Research and Applications, Vol. 3, Issue 1, pp.1676-1679, January - February 2013.

[3] Anna Jain, Baisakhy Dash and Ajit Kumar Panda,"FPGA Design of a Fast 32-bit Floating Point Multiplier Unit", IEEE Conference Publications, pp. 545 – 547, 2012.

[4] Addanki Purna Ramesh and Rajesh Pattimi," High Speed Double Precision Floating Point Multiplier",International Journal of Advanced Research in Computer and Communication Engineering, Vol. 1, Issue 9, pp. 647 – 650, November 2012.

[5] Xia Hong and Jia Jinging,"Research and optimization on Rounding Alogrithms for Floating-Point Multiplier",IEEE Conference Publications,International Conference on Computer Science and Electronics Engineering, pp. 137 – 142, 2012.

[6] Tashfia.Afreen, Minhaz. Uddin Md Ikram, Aqib. Al Azad, and Iqbalur Rahman Rokon, "Efficient FPGA Implementation of Double Precision Floating Point Unit Using Verilog HDL" ,International Conference on Innovations in Electrical and Electronics Engineering,Oct. 6-7, 2012 .

[7] Puneet Paruthi, Tanvi Kumar and Himanshu Singh,"Simulation of IEEE 754 Standard Double Precision Multiplier using Booth Techniques", International Journal of Engineering Research and Applications, Vol. 2, Issue 5, pp. 1761 - 1766, September- October 2012.4

[8] B.Sreenivasa Ganesh, J.E.N.Abhilash and G. Rajesh Kumar, "Design and Implementation of Floating Point Multiplier for Better Timing Performance", International Journal of Advanced Research in Computer Engineering & Technology, Vol. 1, Issue 7, September 2012.