International Journal of Engineering Research and General Science Volume 2, Issue 3, April-May 2014
ISSN 2091-2730

# Design of Floating Point Multiplier for Fast Fourier Transform Using Vedic Sutra

Yashkumar. M. Warkari[1], Prof L.P.Thakare[2], Dr. A.Y. Deshmukh[3]

[1]Research Scholar (M.Tech), Department of Electronics Engineering, G.H.Raisoni college of Engineering, Nagpur

[2]Assistant Professor, Department of Electronics Engineering, G.H.Raisoni college of Engineering, Nagpur

[3]Professor, Department of Electronics Engineering, G.H.Raisoni college of Engineering, Nagpur

E-mail- yashwarkari@gmail.com

**Abstract-** The need of multipliers in mathematics seems to be a very important aspect .Likewise not only in maths but also the technical applications based on maths ,the multipliers are used many number of times. Floating point multiplier is also one of the sinequanon design used in FFT's digital filters, various transforms etc. The aim of the proposed design in the paper is to provide the multiplication of floating point numbers within less possible time with more accuracy. The lacuna elicits by conventional method has been obliterated by the aid of vedic sutra in order to reduce the complexity of design as well as functionality of entire circuit. The device used in this proposed design is 7a30tcsg324-3.

**Keywords**- DSP, VHDL, IEEE, MSB, CLA, Rounding, Normalize

## 1. INTRODUCTION

The rudimentary designs which exhibits the behaviour similar to the floating multipliers has been incorporated for the reckon purpose for high range numerical values. Basically any digital designs has to be imited through some fixed steps of designing method. Conventional methods are used most frequently right from myriads to smaller numbers. In mathematical domains like algebra, calculus one thing has got obsession apropos to multiplication which is popularly known as multiplication of floating point numbers. It is doddle to solve the multiplication operation of such numbers by aid of paper & pen method. But when question comes to digital circuits then the decimal point in floating number plays a vital role. Vedic Mathematics is one such division that involves thinking and mind is used at its best. In India most of the students studying the conventional mathematics can solve problems that are taught to them at school, but they are unable to solve the problems that are new and not taught to them. Author has proposed The comparison & description of basics of multiplication & different algorithms for designing. of based circuit designs In [1].

It proposed that the p bit multiplication elicits the 2p bit result so to rehash the result again into p bit range the rounding techniques plays an vital role. The different algorithms of sticky bit generation have elucidated .So it can be discussed In [2].

It proposed the method & apparatus for generation of efficacy in the obtained results. It has also elucidated the distinct blocks importance for bringing the more appropriate results of each block. It heeds on rounding and controlling operation in design which has discussed In[3].

## 2. IMPORTANCE OF VEDIC MATHEMATICS

High speed arithmetic operations are very important in many signal processing applications. Speed of the digital signal processor (DSP) is largely determined by the speed of its multipliers. In fact the multipliers are the most important part of all digital signal processors; they are very important in realizing many important functions such as fast Fourier transforms and convolutions. Since a processor spends considerable amount of time in performing multiplication operation of numbers, an improvement in multiplication speed can greatly improve by the system performance. Multiplication can be implemented using many algorithms such as array multiplication, booth multiplication, carry save adder, and Wallace tree algorithms used for myriads.

The multiplier architecture is based on this sinequanon Urdhva tiryakbhyam sutra. The prime advantage of this algorithm is that partial products and their sums are calculated in parallel manner. This parallelism makes the multiplier clock independent than event clock. The other main advantage of this multiplier as compared to other multipliers is its regularity of reckon. Due to this modular type of

nature the lay out design will be easy. The defined architecture can be explained with aid of two eight bit numbers i.e. the first multiplier number and second multiplicand number are eight bit numbers.

Urdhava Tiryakbhyam is a Sanskrit word which means vertically defined as urdhavya and crosswise as triyakbhayam in English. The method is a general multiplication formula applicable to all cases of multiplication examples. It is based on a novel & rudimentary concept through which all kind of partial products are generated concurrently. Demonstrates a 4 x 4 binary multiplication using this method.

The method can be generalized for any N x N bit multiplication. This type of multiplier is independent of the clock frequency of the processor because the partial products and their sums are calculated in parallel manner. The net advantage is that it reduces the need of microprocessors to operate at increasingly higher possible clock frequencies. As the depending operating frequency of a processor increases the number of switching instances also increases. There are again several methods that can be followed to reduce logical expression such as Boolean algebra , tabulation method etc. It is tedious task to make a use of basic Boolean reducing rules to apply over a huge logical terms in such a lengthy expression. It proposed the significance of methods based on vedic sutra In[4].

## 3. METHODOLOGY OF FLOAING POINT MULTIPLIER

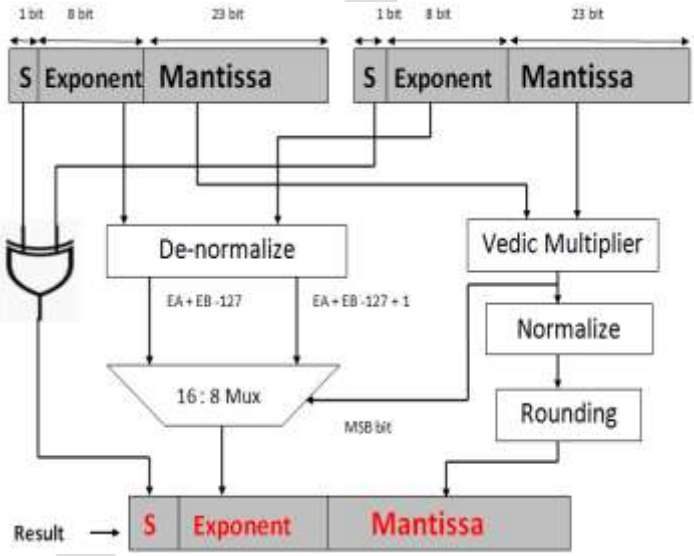



Fig1- Floating multiplier circuit

Basically the above delineated circuit is used for multiplication of two floating point numbers. There is no definite logic level for representation of decimal point in digital circuit. So it is herculean to store the decimal point into the storing elements like flip flops, registers, memories etc in true form. So we ought to cogitate that how we can store the floating number. So we have a IEEE formats for different ranges like single precision, double precision, quad precision etc. In this paper single precision format is preffered.

| S | Exponent | Fraction |
|---|---|---|
| 1 bit | 8 bits | 23 bits |

Fig2- Single precision format

**Ex-1 convert 6.75 into single precision format**

6=110 in binary

.75 * 2 = 1.5

.5 * 2 = 1.0
.0 * 2 = 0.0
.0 * 2 = 0.0
110.11000000000000000000000
=1.10110000000000000000000 * $2^2$

Exponent = 127 + 2= 129 or 10000001 in binary

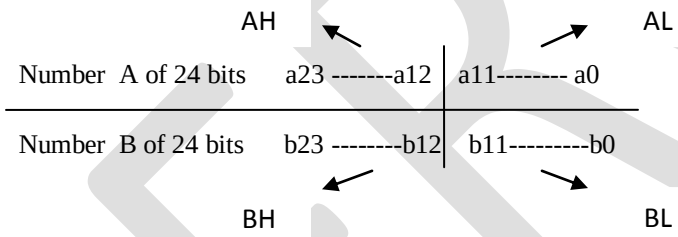
Mantissa = 10110000000000000000000

**6.75 in 32 bit floating point IEEE representation:-**
01000000110110000000000000000000

In above circuit the mantissas of two input numbers have to be multiplied by 24 bit vedic multiplier. Normaliser is followed vedic multiplier, which is again followed by rounding block. The exponent is reckon by de-normalisation followed by one 2:1 mux which uses MSB bit of vedic multiplier output as a select line. Eventually the sign of result can be determined by the Xoring of two sign bits of given input numbers.

## 4. METHODOLOGY OF 24-BIT VEDIC MULTIPLIER

The 24-bit vedic multiplier design is a block which can be formed of progression of four 12-bit vedic multiplier design blocks. The 12-bit multiplier design can be modeled first by lower range of vedic multipliers. Then eventually the final design can be obtained by structural modeling in VHDL code.




Equation => $(A_H * B_H) + (A_H * B_L) + (A_L * B_H) + (A_L * B_L)$

The two numbers 24-bit each is ramified into four parts i.e $A_H$, $A_L$, $B_H$, $B_L$ . Each subpart is of 12-bit which means four 12-bit vedic multiplier again.

When the 12-bit vedic multiplier properly map then it elicit the 24-bit output from all four 12-bit vedic unit in Fig3. After that the role of adders comes into picture which exhibits through Equation elucidated above .
Then the output elicited from middle two 12-bit vedic units is endowed to 24bit CLA adder-1.The all output bits excluding carry has used as one input of 24bit CLA adder-2 and the second input of 24bit CLA adder-2 is framed by concatenating 12 MSB output bits of last 12-bit vedic multiplier unit with twelve leading zeroes. The output elicited from First 12-bit vedic unit is endowed to 24bit CLA adder-3 as one input and the second input of 24bit CLA adder-3 is framed by concatenating 12 MSB output bits of last 24-bit CLA adder-2 unit with eleven leading zeroes and output of OR gate. Output carries of 24bit CLA adder-1 & 24bit CLA adder-2 has used as an input of OR gate .Eventually the 48 bit output can be obtained by concatenating the bits shown.

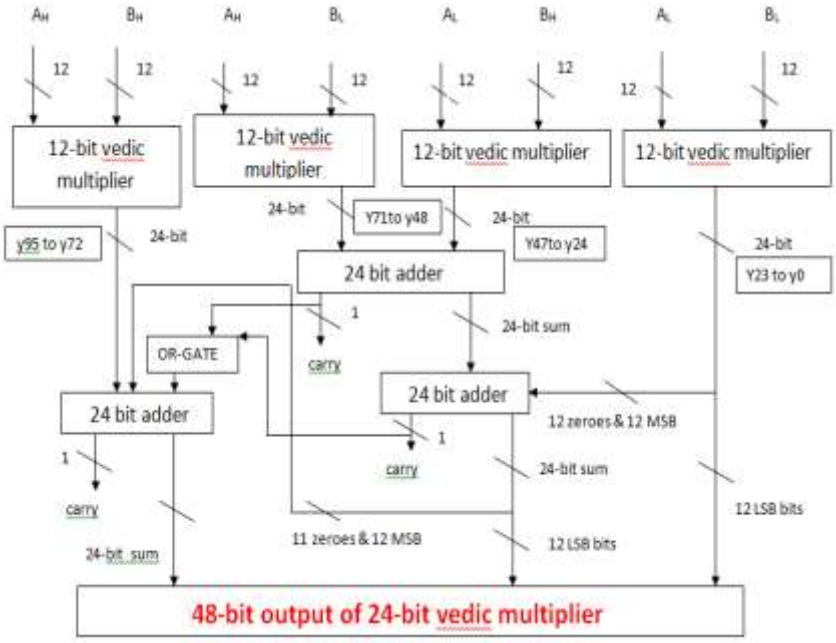



Fig3- The 24 * 24 bit vedic multiplier

## 5. NORMALIZE & ROUNDING

Normalization is quoth to be a sinequanon block for the entire design. When 48 bit output product is obtained from the vedic multiplier block, then the entire 2p bit result has to be normalized first in order to get correct answer. The output should be in 1. Form .The decimal point is suppose to be place after first Two MSB bits. Accordingly the output has to be metamorphose into the above alluded form. When decimal point shifts towards left hand side of result accordingly add 1 to the output of denormalizer & if the decimal point shifts towards right hand side of result .

EX - The output of vedic multiplier block is as follows

10.011101011111010111101010000100000000000000000

The output of normalizer
1.001110101111101011110101000010000000000000000

Now in above example the decimal point has shifted to left by one bit position in order to rehash into normalized form as alluded above. Therefore the addition of 1 is needed to the output of denormalizer.

Rounding is also plays a vital role in the reckon of mantissa. The main moto of rounding is to curtail the plethora bits so that only the desired number of bits can be there to represent the output. In this case the answer should be in 23 bits only but the actual output of vedic multiplier unit is of 48 bits. So extra25 bits needs to be curtail at anyhow. There are several methods for rounding . Accurate rounding of transcendental mathematical functions is difficult because the number of extra digits that need to be calculated to resolve whether to round up or down cannot be known in advance. This problem is known as "the table-maker's dilemma".
The simple and easiest method of rounding is rounding towards zero. In this technique just consider the left part of decimal point & ignore rest part of number. The technique is also known by name “truncation”.
The another technique which is taught at school level popularly known as round towards half way. In this technique a base number is used as a reference & the part after decimal point is to be compared with reference value. If the number found to be greater than or equal to the reference value then round the entire number to its next adjacent greater value. Otherwise round the entire number to its next adjacent smaller value. In this design the rounding towards zero has been used for rounding of 48 bit result int0 23 bit.

**Table No -1. Rounding table**

| Y | Round Down (towards ∞) | Round up (towards +∞) | Round towards zero | Round away from zero | Round to nearest |
|---|---|---|---|---|---|
| + 23.67 | + 23 | + 24 | + 23 | + 24 | + 24 |
| + 23.50 | + 23 | + 24 | + 23 | + 24 | + 24 |
| + 23.35 | + 23 | + 24 | + 23 | + 24 | + 23 |
| + 23.00 | + 23 | + 23 | + 23 | + 23 | + 23 |

EX – The output of normalize is as follows

1.01101001100001011100001000000000000000000000000

The output of rounding
01101001100001011110001

In the above example the first 23 bits after decimal point has considered & plethora bits has curtailed. Therefore the desired mantissa bit range for single precision format is being achieved.

## 6. EXPONENT UNIT

The exponent of two input numbers can be reckon by the aid of an exponent unit. The exponent of the actual decimal point answer should have similarity with the answer obtained from the exponent unit. The exponent has got a nexus mainly with the decimal point. The shifting operation elicits the variation of the value of an exponent. According to the shifting the unit will elicits the desired output. The denormaliszer plays an vital role in reckoning .

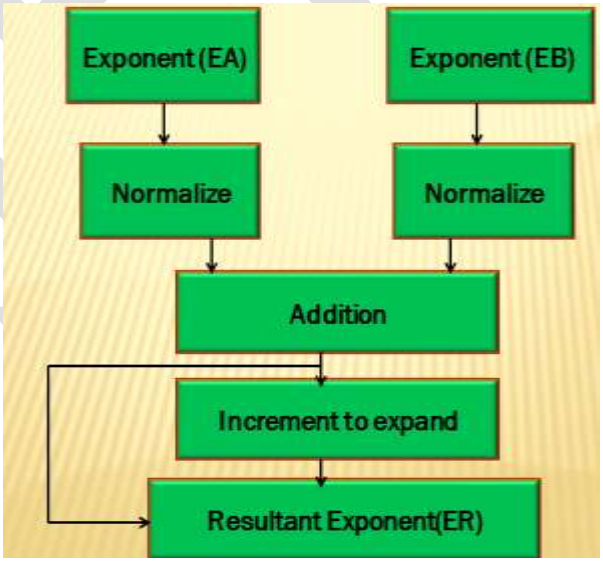



Fig 4 - The Exponent unit

The exponent unit is to reckon the result exponent. During process of metamorphose of single precision format the count of number of shifted positions of decimal point is being added to the bias, which is 127 in case of single precision format.

At the time of reckoning the exponents of input numbers has to be subtracted from the bias always. Eventually the correct exponent will be obtained. The mathematical equation deals with reckon is EA + EB – 127. But the exponent unit has sort of nexus with mantissa unit. The mux will build a nexus of these two units. While normalizing the result of vedic multiplier sometimes it is found that the decimal point has to shift to one bit position left & during this shift the 1 has to be added to the exponent of result. Sometimes The normalized result may be obtained then no need to add 1. The mux delineated in main floating multiplier will determine the need of 1. Thus the resultant exponent can be obtained.

Ex - 8.5 * 240.1 = 2040.85

8.5=1000.101
= $1.000101 * 10^3$
=127+3

EA=130

240.1=11110000.01
= $1.1110000 * 10^7$
=127+7

EB=134

2040.85=11111111000.1010101
= $1.11111110001010101 * 10^{10}$
=127+10

ER=137

Equation as per elucidation
ER= (EA-127) + (EB-127) + 127
= (130-127) + (134-127) + 127
ER = 137

## 7. SIGN GENERATING LOGIC

The Sign of result can be obtained by XORing of signs of input number. The logic 1 sign bit represents negative number, whereas logic 0 sign bit represents positive number. If both the numbers are of different sign then they elicits negative result. Otherwise elicits the form same as form of input number

**Table No -2. Sign table**

| PRODUCT TERMS | SA | SB | SR= SA xor SB |
|---|---|---|---|
| 1. 4 * 5 = 20 | 0 | 0 | 0 |
| 2. -3 * 4 = -12 | 1 | 0 | 1 |
| 3. 3 * -4 = -12 | 0 | 1 | 1 |
| 4. -5 * -5 = 25 | 1 | 1 | 0 |

Where SA is sign of first number. SB is sign of second number. SR is sign of result number.

**Table No -3. Synthesis report values**

| Parameters of floating multiplier | Values |
|---|---|
| 1. IO Buffers | 96 |
| 2. Combinational path delay | 21.132 ns |
| 3. Memory usage | 279516 KB |
| 4. Number of slice LUT'S | 966 out of 21000 |
| 5. 1-bit Xor gates | 1566 |
| 6. Global maximum fanouts | 100000 |

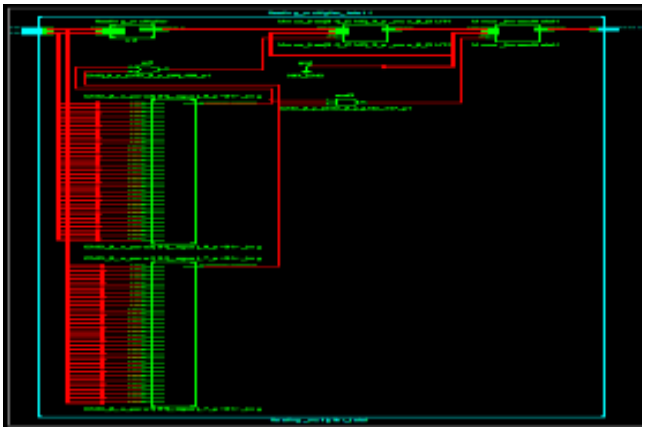
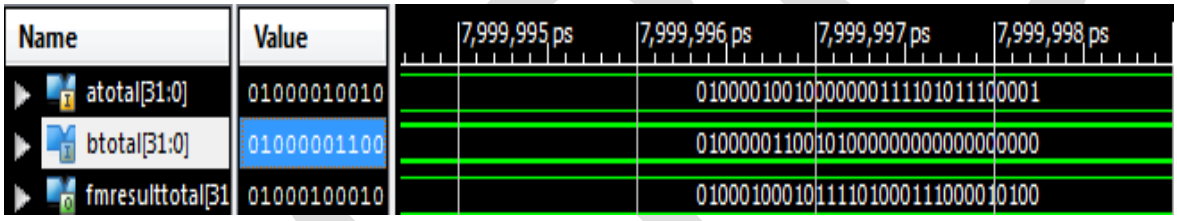


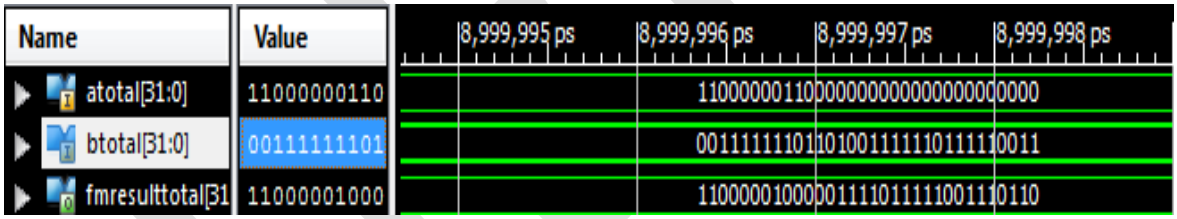
Fig 5 - RTL of floating multiplier

The results shown below is the output simulation of floating multiplier for the two input numbers in IEEE single precision format. Where atotal and btotal are input numbers And fmresulttotal is the output.

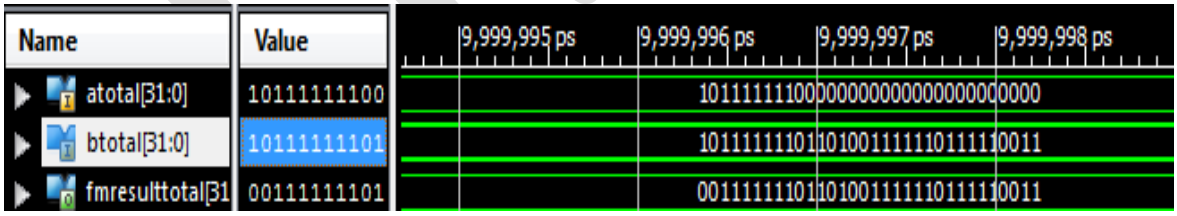
1. 48.12 * 18.5 = 890.22




2. -6 * 1.414 = -8.484




3. -1 * -1.414 = 1.414




4. 6 * 0 = 0

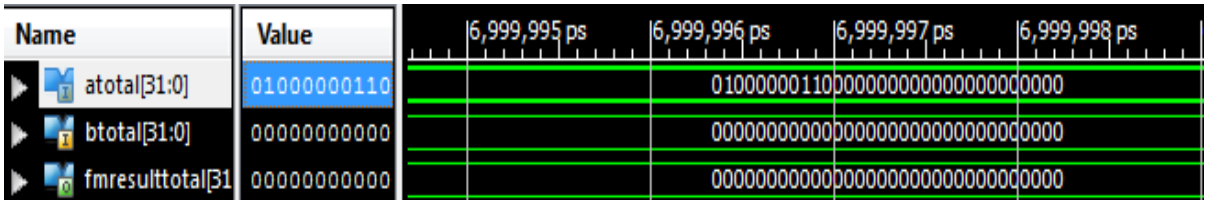



Fig 6 - Simulation results of floating multiplier

## 8. CONCLUSION

Thus we have designed the floating point multiplier using vedic sutra. The different adder designs can be used according to the requirement of particular application. By doing this the desideratum of cull design can be obtained. The implication of OR gate is beneficial for obliterating errors in multiplication operation. The normalization plays an vital role in bringing the correct mantissa of a result & the extra LSB bits can be connived from the mantissa bit stream.

## REFERENCES:

[1] Rajkumar singh, shivananda Reddy, Floating point multipliers simulation & synthesis using VHDL.

[2] Xia Hong, Jia Jingping, Research & optimization on rounding algorithms for floating point multiplier, International conference on computer science and electronics engineering 2012.
[3] Jeffrey D. Brown, Roy R. Faget, Scott A. Hilker, Apparatus for determining sticky bit value in airthmatic operations, United states statutory invention registration in Aug 3, 1993.
[4] Swami Bharti Krishna Tirtha, vedic mathematics, motilal bansidass publication 1992.
[5] Daniel J. Bernstein , fast mathematics and its applications, Algorithmic number theory MSRI publication volume 44,2008..
[6] Robert K. Yu and Gregory B. Zyner. 167 MHz Radix4 Floating Point Multiplier [C]. Proc. 12th Symp. Computer Arithmetic, 1995: 149-154
[7] Stuart Franklin. Oberman .Design Issues in High Performance Floating Point Arithmetic Units[R].Technical Report:CSL-TR-96 711.1996,12
[8] G. Even, S.M. Mueller, and P.M. Seidel. A Dual Mode IEEE Multiplier[C] . Proc. Second IEEE Int'l Conf. Innovative Systems inSilicon, 1997: 282-289
[9] Behrooz Parhami. Computer Arithmetic: Algorithms and Hardware Designs[M]. NewYork, Published by Oxford University Press Inc, 2000
[10] Shlomo Waser and Michael J. Flynn.Introduction to Arithmetic for Digital Systems Designers[M].CBS College Publishing. NY,1982 :139
[11] M.Nagarjuna, R.Surya Prakash, B.Vijay Bhaskar. High speed ASIC design of complex multiplier using Vedic mathematics. IJERA vol 3, Issue 1, January-February 2013, PP. 1079-1084.

[12] Vaijyanath Kunchigi, Linganagouda Kulkarni, Subhash Kulkarni. High speed & area efficient Vedic multiplier. Jawaharlal Nehru technological university.