IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p-ISSN: 2278-8735
PP 44-51
www.iosrjournals.org

Vedic Mathematics Based Floating Point Multiplier Implementation for 24 Bit FFT Computation

Athira Menon M S[1], Renjith R J[2]

[1]*(PG Student, Dept. of Electronics and Communications, SCTCE Pappanamcode, Kerala, India)*
[2]*(Assistant Professor, Dept. of Electronics and Communications, SCTCE Pappanamcode, Kerala, India)*

Abstract: *The computational complexity of various data processing applications is vastly reduced when signals are represented in the frequency domain. Fourier analysis converts a signal from its original domain to a representation in frequency domain. In launch vehicle systems, FFT is required for telemetry data processing applications. Since the systems work in real time, a fast and efficient computation of the FFT is called for. Vedic mathematics based on ancestral Indian Vedas gives a different multiplication algorithm to carry out fast multiplication. The idea for designing the multiplier unit is adopted from ancient Indian mathematics "Vedas". The Urdhva-Tiryakbhyam sutra (method) was selected for implementation since it is applicable to all cases of multiplication. DSP applications essentially require the multiplication of binary floating point numbers. The IEEE 754 standard provides the format for representation of Binary Floating point numbers. Vedic Multiplication Technique is used to implement IEEE 754 Floating point multiplier. The Urdhva-Tiryakbhyam sutra is used for the multiplication of Mantissa. This paper deals with the 24 bit FFT implementation using IEEE754 multiplication based on vedic mathematics and compare the result with conventional multiplier. Design and HDL coding was carried out using Verilog using the Libero IdeV9.1 project environment, natively used for the Actel Pro-Asic devices. The code synthesis was done using Synplify and simulation was done using Modelsim*

Keywords - *Fast Fourier Transform, IEEE754 multiplication, LiberoIdeV9.1. Urdhava-Tiryakbhyam, Vedic mathematics.*

I. INTRODUCTION

Nowadays the computational complexity of various data processing applications is vastly reduced when signals are represented in the frequency domain. Fourier analysis converts a signal from its original domain often in time or space to a representation in the frequency domain and vice versa. A direct implementation of the Discrete Fourier Transform (DFT) of an N-point sequence requires $0(n)^2$ computations. Fast Fourier Transform (FFT) algorithm rapidly computes the DFT by factorizing the DFT matrix into a product of sparse (mostly zero) factors, thereby reducing the computational complexity to 0(n logn). Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics.

In present scenario every process should be rapid, efficient and simple. Fast Fourier transform (FFT) is an efficient algorithm to compute the N point DFT. It has great applications in communication, signal and image processing and instrumentation. But the Implementation of FFT requires large number of complex multiplications, so to make this process rapid and simple it's necessary for a multiplier to be fast and power efficient. To tackle this problem Urdhva Tirvagbhyam in Vedic mathematics is an efficient method of multiplication [1].

The IEEE Standard for Binary Floating Point Arithmetic (IEEE 754) is the most widely used standard for floating point computation. Scientific notation represents numbers as a base number and an exponent. Floating-point solves a number of representation problems. Fixed-point has a fixed window of representation, which limits it from representing very large or very small numbers. Also, fixed-point is prone to a loss of precision when two large numbers are divided.

Vedic mathematics mentioned on ancestral Indian Vedas gives a different multiplication algorithm to carry out fast multiplication. The Sutras Urdhava-Tiryakbhyam and Nikilam Sutras give easiest way of mental calculation when performing multiplication. Among these 2 sutras, Urdhava-Tiryakbhyam employs parallel multiplication and exhibits high degree of parallelism compared to other parallel multipliers.

In this paper, we propose the Vedic Multiplication algorithm for multiplication of 24 bit mantissa. The details of Vedic Multiplication with their advantages over the conventional multiplication method are discussed. The paper describes the implementation and design of IEEE 754 Floating Point FFT Multiplier based on Vedic Multiplication Technique.

II. VEDIC MATHEMATICS

Vedic mathematics[2] is the name given to the ancient system of mathematics which was rediscovered from the Vedas. In compare to conventional mathematics, Vedic mathematics is simpler and easy to understand. Swami Bharati Krishna Tirthaji Maharaj (1884-1960), re-introduced the concept of ancient system of Vedic mathematics. The word 'Vedic' is resultant from the word 'Veda' which means the store-house of all knowledge. Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya - Veda (book on civil engineering and architecture), which is an upa-veda of Atharva Veda. It gives explanation of several mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus.

Based on ancestral Indian Vedas swami Jagad guru shankaracharya and Bharathi Krishna Theerthaji worked together and developed sixteen sutras and thirty two sub sutras. Vedic mathematics is based on 16 Sutras dealing with various branches of mathematics like arithmetic, algebra, geometry, factorization etc. The Sutras Urdhava-Tiryakbhyam and Nikilam Sutras give easiest way of mental calculation when performing multiplication. Vedic mathematics is a domain which presents various effective algorithms that can be applied in different branches of engineering such as digital signal processing and computing. The number of logic levels and logic delay is being reduced using the Urdhva- Tiryakbhyam sutra.

Vedic mathematics is not only a mathematical wonder but also it is logical method. That's why it has such a degree of eminence which cannot be disapproved. Due to these phenomenal characteristics, Vedic maths has crossed the boundaries of India and has become an interesting topic of research. Vedic maths deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful.

III. URDHVA-TIRYAKBHYAM METHOD

The Sanskrit word "Urdhva" means "Vertically" and "Tiryagbhyam" means "crosswise". This multiplication formula is applicable to all cases of algorithm for N bit numbers. Traditionally the sutra is used for the multiplication of two numbers in decimal number system. Advantage of using this type of multiplier is that as the number of bits increases, delay and area increases very slowly as compared to other multipliers.

Urdhava-Tiryakbhyam employs parallel multiplication and exhibits high degree of parallelism compared to other parallel multipliers. In conventional parallel multiplication method partial products get summed up after the generation of all partial products. In the case of Urdhava-Tiryakbhyam, multiplication vertically and crosswise means summation will take place just after partial products for a column gets generated. This high degree parallelism gives better speed compared to other parallel multiplication. Implementation of fast vedic multiplier will improve the performance of the current processors. Important feature of Urdhava-Tiryakbhyam is, multiplication of numbers in any radix can be implemented easily. This work gives detailed review about highly efficient architecture for multiplication algorithm urdhava-tiryakbhyam based on Indian ancestral Vedas. It gives high modular approach for implementing higher order bits. The line diagram of Urdhva-Tiryakabhyam (UT) sutra for 2 bit, 3 bit and 4 bit is shown in figure 1.

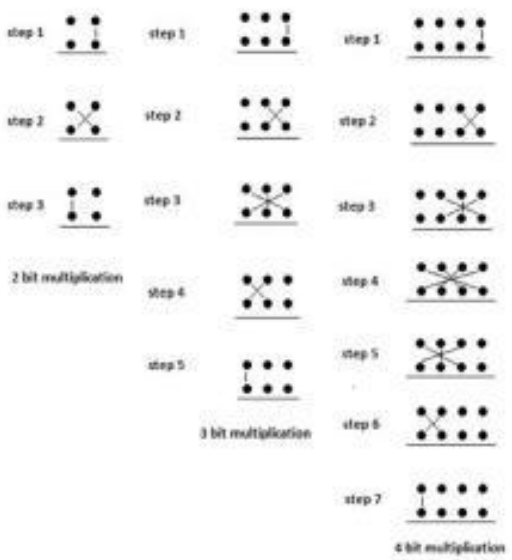


Fig 1 Line diagram of UT Sutra

This Sutra shows how to handle multiplication of a larger number (N x N, of N bits each) by breaking it into smaller numbers of size (N/2 = n, say) and these smaller numbers can again be broken into smaller numbers (n/2 each) till we reach multiplicand size of (2 x 2) Thus, simplifying the whole multiplication process [3]. The processing power of this multiplier can easily be increased by increasing the input and output data bus widths since it has a quite regular structure. Due to its regular structure, it can be easily layout in a silicon chip.

In the figure 2, 4-bit binary numbers A0A1A2A3 and B0B1B2B3 are considered. The result obtained is stored in R0R1R2R3R4R5R6R7.In the first step [A0, B0] is multiplied and the result obtained is stored in R0. Similarly in second step [A0, B1] and [A1, B0] are multiplied using a full adder and the sum is stored in R1 and carry is transferred to next step. Likewise the process continues till we get the result.

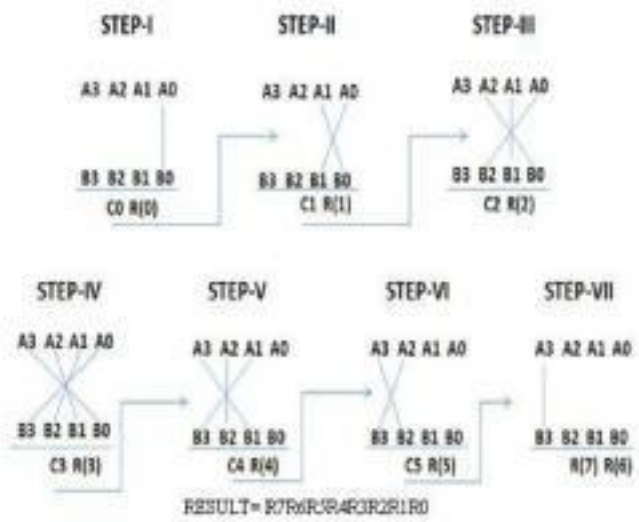


Fig 2 Multiplication method of Urdhva-Tiryakbhyam

In the figure 3, alternative way of multiplication using Urdhva-Tiryakabhyam sutra is carried out. Figure 4 shows the vedic multiplier flow chart.

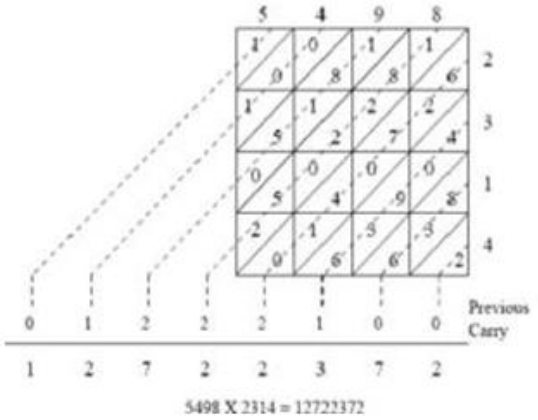


Fig 3 Alternative way of multiplication of UT Sutra

Consider two 4-bit binary numbers a3a2a1a0 and b3b2b1b0. The partial products (P7P6P5P4P3P2P1P0) generated are given by the following equations [4]:

i. P0= a0b0
ii. P1= a0b1 + a1b0
iii. P2 = a0b2 + a1b1 + a2b0+ P1
iv. P3= a0b3 + a1b2 + a2b1 + a3b0+ P2
v. P4 = a1b3 + a2b2 + a3b1 + P3
vi. P5 = a1b2 + a2b1 + P4
vii. P6 = a3b3 + P5
viii. P7 = carry of P6

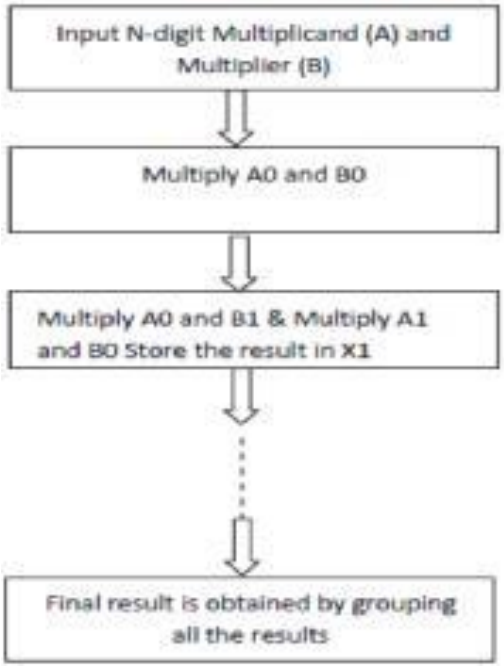


Fig 4 Vedic Multiplier Flowchart

IV. FLOATING POINT MULTIPLICATION

IEEE 754 floating point standard is the most common representation today for real numbers on computers. The IEEE (Institute of Electrical and Electronics Engineers) has produced a Standard to define floating-point representation and arithmetic. Although there are other representations, it is the most common representation used for floating point numbers. The standard brought out by the IEEE come to be known as IEEE 754. The IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754) is the most widely- used standard for floating point computation. The general representation of IEEE754 single precision floating point format is shown in figure 5. Dealing with fixed-point arithmetic will limit the usability of a processor. If operations on numbers with fractions (e.g. 10.2445), very small numbers (e.g. 0.000004), or very large numbers (e.g. 42.243x105) are required, then a different one representation is in order is the floating-point arithmetic.[5] The floating point is utilized as the binary point is not fixed, as is the case in integer (fixed-point) arithmetic. Consider a simple floating-point number, -2.42x103. The '-' symbol indicates the sign component of the number, while the '242' indicate the significant digits component of the number, and finally the '3' indicates the scale factor component of the number. It is interesting to note that the string of significant digits is technically termed the *mantissa* of the number, while the scale factor is appropriately called the *exponent* of the number. The general form of the representation is the following:-

$$(-1)^S * M * 2^E \qquad (1)$$

Where ,
S represents the sign bit,
M represents the mantissa and
E represents the exponent

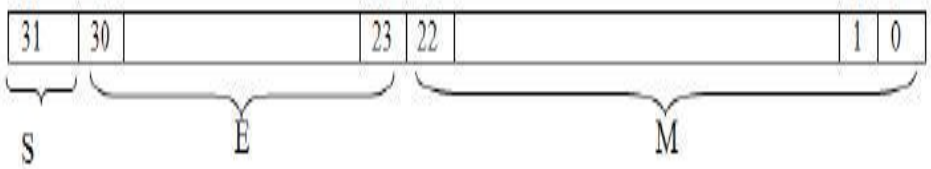


Fig 5 IEEE single precision floating point format

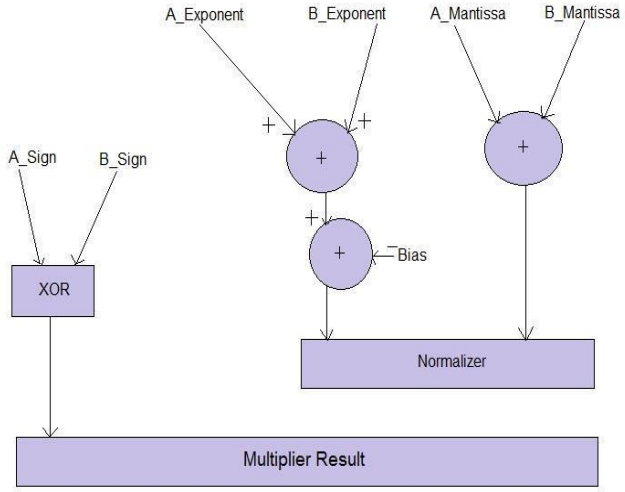


Fig 6 Floating point multiplier block diagram

Normalized floating point numbers have the form**:-**

$$Z= (-1S) * 2 (E - Bias) * (1.M).$$

Figure 6 shows the floating point multiplier diagram. The flowchart for the multiplication of floating point numbers is shown in figure 7. To multiply two floating point numbers the following is done:
1. Multiplying the significand; i.e. (1.M1*1.M2).
2. Placing the decimal point in the result.
3. Adding the exponents; i.e. (E1 + E2 – *Bias*).
4. Obtaining the sign; i.e. s1 xor s2.
5. Normalizing the result; i.e. obtaining 1 at the MSB of the results‟ significand.
6. Rounding the result to fit in the available bits.
7. Checking for underflow/overflow occurrence.

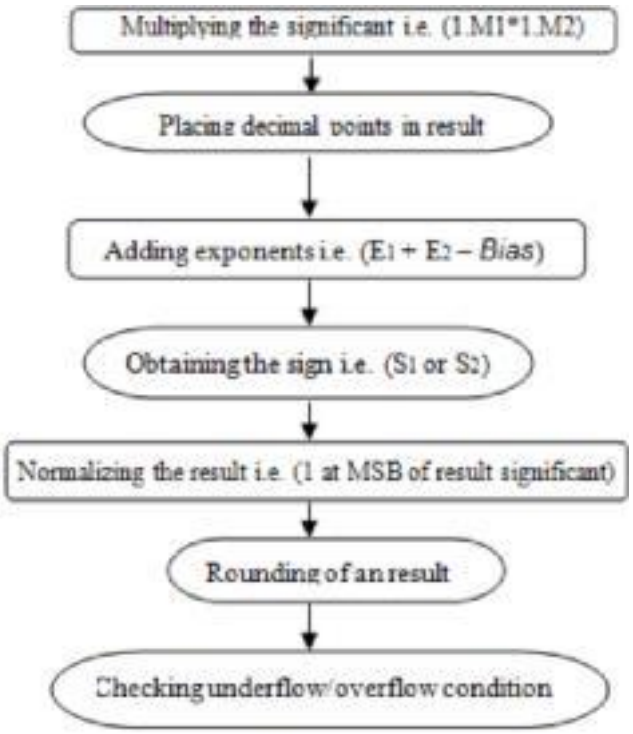


Fig 7 Steps of multiplying Floating Point numbers

V. FAST FOURIER TRANSFORM

Fast Fourier Transform is important data processing technique in communication systems and DSP systems. Fast Fourier Transform (FFT) is widely used in the field of digital signal processing (DSP) and communication system applications with the advancement of VLSI. The cost and characteristic of FFT processor is decided by butterfly processing unit which consists of complex adders and multipliers. The multiplier usually increases the speed of the FFT processor. Usage of complex multiplications using shift and add operation results in higher hardware cost and also limits the performance of FFT. To improve the performance of such complex calculations Vedic algorithm is used. Vedic algorithm gives efficient implementation of complex multiplier.

A fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) of a sequence, or it's inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from, which arises if one simply applies the definition of DFT, to, where is the data size. Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805[5]. In 1994, Gilbert Strang described the FFT as "the most important numerical algorithm of our lifetime" [6]. An FFT is a way to compute the same result more quickly: computing the DFT of N points in the naive way, using the definition, takes 0(N2) arithametical operations, while an FFT can compute the same DFT in only O(N log N) operations. The difference in speed can be enormous, especially for long data sets where N may be in the thousands or millions. In practice, the computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to N / log N.

5.1 Definition and speed

An FFT computes the DFT and produces exactly the same result as evaluating the DFT definition directly the most important difference is that an FFT is much faster. (In the presence of round off error, many FFT algorithms are also much more accurate than evaluating the DFT definition directly Let x0, ..., xN1 be complex numbers. The DFT is defined by the formula:

$$X(k)= \sum_{n=0}^{N-1} x(n) \exp(-i2\pi kn)$$

Evaluating this definition directly requires 0(N2) operations: there are N outputs X(k), and each output requires a sum of N terms. An FFT is any method to compute the same results in 0(N log N) operations. All known FFT algorithms require (N log N) operations, although there is no known proof that a lower complexity score is impossible. To illustrate the savings of an FFT, consider the count of complex multiplications and additions.

Evaluating the DFT's sums directly involves N2 complex multiplications and N(N1) complex additions, of which 0(N) operations can be saved by eliminating trivial operations such as multiplications by 1. The radix2 Cooley–Tukey algorithm, for N a power of 2, can compute the same result with only (N/2)log2(N) complex multiplications (again, ignoring simplifications of multiplications by 1 and similar) and N log2(N) complex additions. In practice, actual performance on modern computers is usually dominated by factors other than the speed of arithmetic operations and the analysis is a complicated, but the overall improvement from 0(N2) to 0(N log N) remains.

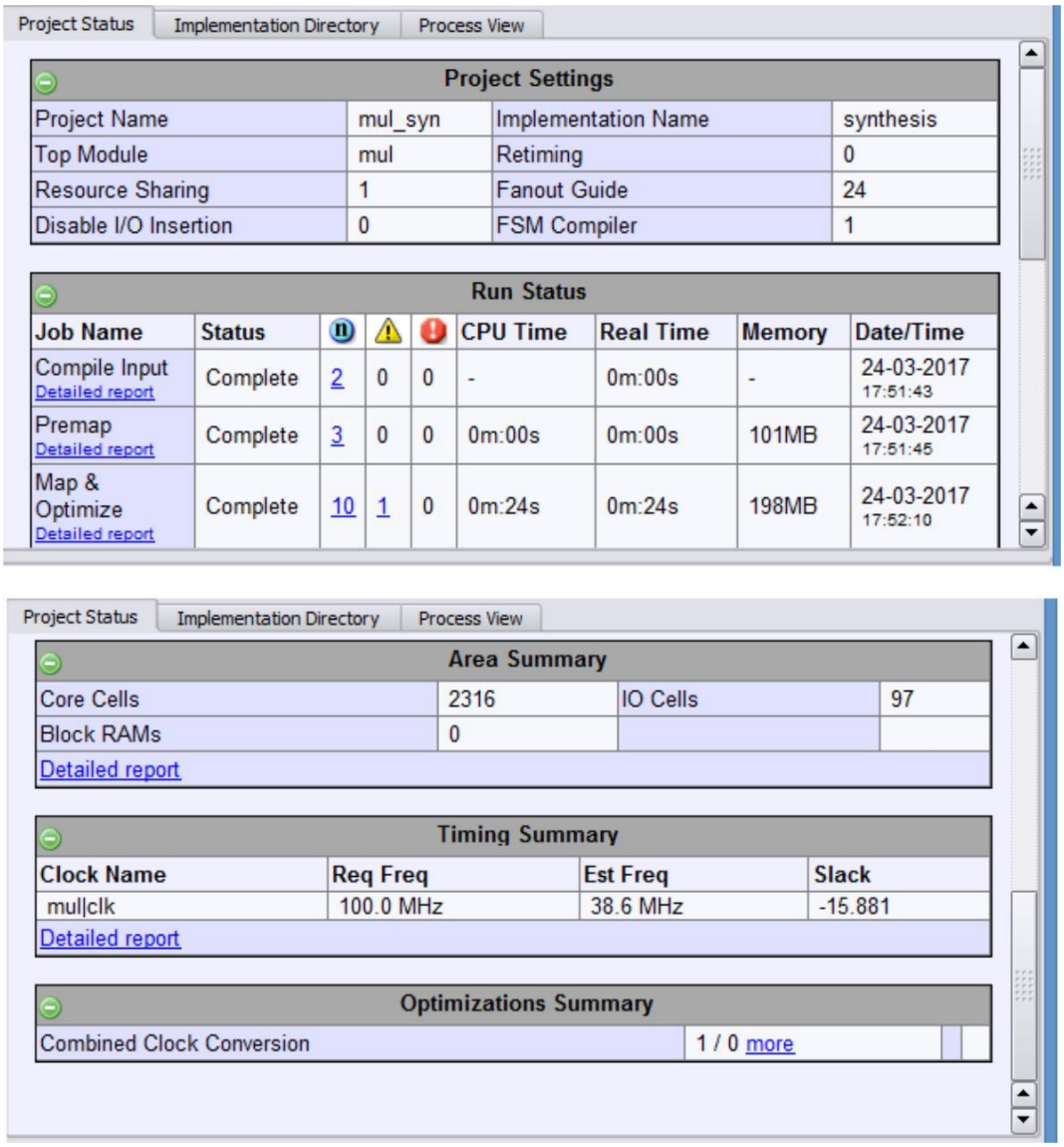
5.2 Combined Approach of FFT with Vedic Mathematics

A 24*24 Vedic multiplier is design by using four 12*12 Vedic multipliers based urdhva triyakbhyam. Here first block 12X12 multiplier consists of lower 12 bits of x i.e. x(11 down to 0) and y(11 down to 0), second block 12*12 Vedic multiplier inputs are x(23 down to 12) and y(11 down to 0) out off 24 bit output of first block lower adder 12 bits are separated and higher order bits are appended as 12 lower bits in front augmented with "000000000000" now second block 24 bits and above 24 generate bits are added and 24 bits sum is generated using 24 bit ripple carry adder. Higher order 12 bit of y(23 down to 12) and lower order 12 bits of x i.e x(11 down to 0) are multiplier and appended infront with "000000000000" to make 36 data similarly both higher order 12 bits of x and y i.e x(23 downto 12) and y(23 downto 12) are multiplier and 36 bit data is formed by appending "000000000000". The above two 36 bits are added to generate a 36 bit data in the right side resultant 36 bits are added to generate final 36 bits the resultant of 24*24 multiplier is 48 bits consists of 36 higher bits from 36 bit adder output and lower order 12 bits 36+12=48 bits. The number of LUTs and slices required for the Vedic Multiplier is less and due to which the power consumption is reduced. Also the repetitive and regular structure of the multiplier makes it easier to design. And the time required for computing

multiplication is less than the other multiplication techniques. An Overflow or Underflow case occurs when the result Exponent is higher than the 8 BIT or lower than 8 BIT respectively. Overflow may occur during the addition of two Exponents which can be compensated at the time of subtracting the bias from the exponent result. When overflow occurs the overflow flag goes up. The under flow can occur after the subtraction of bias from the exponent, it is the case when the number goes below 0 and this situation can be handled by adding 1 at the time of normalization. When the underflow case occur the under flow flag goes high.

VI. RESULTS

The implementation of IEEE754 multiplication using conventional multiplication method and vedic multiplication using Urdhva-Tiryakabhyam method is performed and different parameters like CPU speed, memory utilization, area, estimated frequency is compared and analyzed. From the results shown below in figure 8 and 9, we can find that the parameters like area, memory, CPU speed consumed by vedic multiplication technique is more than the conventional technique.



Project Status | Implementation Directory | Process View

Project Settings

Project Name	mul_syn	Implementation Name	synthesis
Top Module	mul	Retiming	0
Resource Sharing	1	Fanout Guide	24
Disable I/O Insertion	0	FSM Compiler	1

Run Status

Job Name	Status				CPU Time	Real Time	Memory	Date/Time
Compile Input Detailed report	Complete	2	0	0	-	0m:00s	-	24-03-2017 17:51:43
Premap Detailed report	Complete	3	0	0	0m:00s	0m:00s	101MB	24-03-2017 17:51:45
Map & Optimize Detailed report	Complete	10	1	0	0m:24s	0m:24s	198MB	24-03-2017 17:52:10

Project Status | Implementation Directory | Process View

Area Summary

Core Cells	2316	IO Cells	97
Block RAMs	0		
Detailed report			

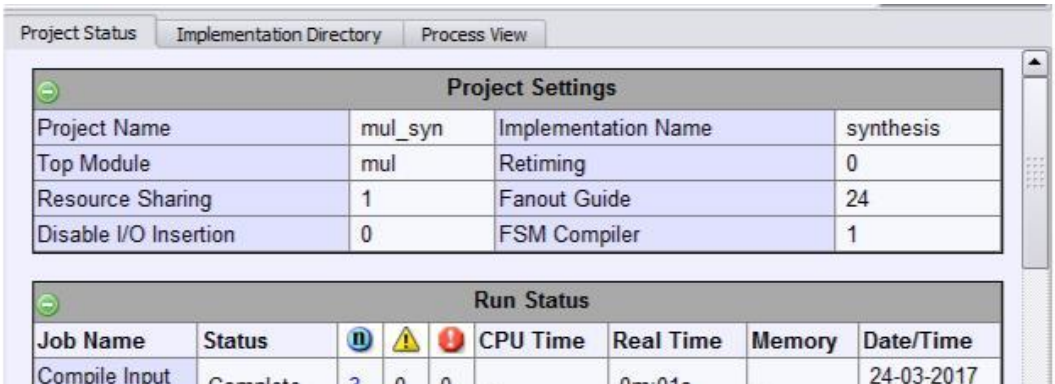
Timing Summary

Clock Name	Req Freq	Est Freq	Slack
mul|clk	100.0 MHz	38.6 MHz	-15.881
Detailed report			

Optimizations Summary

Combined Clock Conversion	1 / 0 more

Fig 8 Synthesis result of 24bit ieee754 multiplication using conventional multiplier



Project Status | Implementation Directory | Process View

Project Settings

Project Name	mul_syn	Implementation Name	synthesis
Top Module	mul	Retiming	0
Resource Sharing	1	Fanout Guide	24
Disable I/O Insertion	0	FSM Compiler	1

Run Status

Job Name	Status				CPU Time	Real Time	Memory	Date/Time
Compile Input								24-03-2017

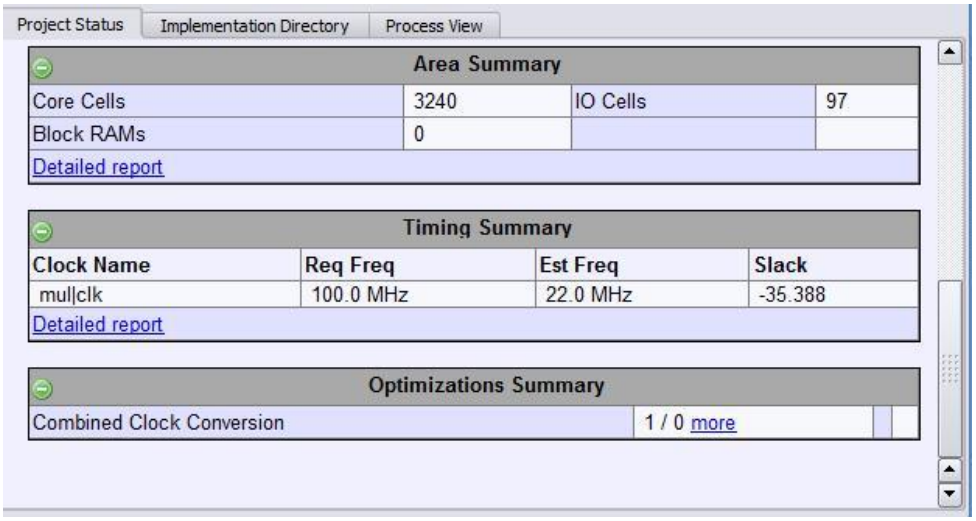


Fig 9 Synthesis result of 24bit ieee754 multiplication using UT Sutra.

VII. CONCLUSION

In this paper a novel technique of Fast Fourier Transform (FFT) design methodology using Vedic mathematics algorithm is discussed and implementation of 24 bit IEEE754 multiplication of both conventional and vedic multiplication technique is performed. The comparison between both techniques is shown in the result. From the result we can find that the parameters like area, memory, CPU speed consumed by vedic multiplication technique is more than the conventional technique. This problem is overcomed by using carry save adder or ripple carry adder at the addition part. This will be done as future work.

The design is based on Vedic method of multiplication that is quite different from the conventional method of multiplication like add and shift. This also gives chances for modular design where smaller block can be used to design the bigger one. This gives method for hierarchical multiplier design. So the design complexity gets reduced for inputs of large no of bits and modularity gets increased. This will help in designing FFT structure, as its give effective utilization of structural method of modelling. An FFT circuit has been described that provides the high performance with Small area which has great applications in communication, signal and image processing and instrumentation that can also benefit future needs of wireless communications systems. Urdhva Tiryakbhyam one of the sutra of Vedic Mathematics, being a general multiplication formula, is equally applicable to all cases of multiplication. The conventional multiplication method requires more time & area on silicon than Vedic algorithms. More importantly processing speed increases with the bit length.

REFERENCES

[1] M.E.Paramasivam, Dr.R.S.Sabeenian, ‘‘An Efficient Bit Reduction Binary Multiplication Algorithm using Vedic Methods”, IEEE pp 25-28, 2010

[2] Amrita Nanda, Shreetam Behera, “Design and Implementation of Urdhva-Tiryakbhyam Based Fast 8×8 Vedic Binary Multiplier,”International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181J. Vol. 3 Issue 3, March - 2014.

[3] Ashish Raman, Anvesh Kumar, R.K.Sarin, “High Speed Reconfigurable FFT Design by Vedic Mathematics”, journal of Computer Science and Engineering, vol.1, pp 59-63 May 2010.

[4] A Debasish Subudhi, Kanhu Charan Gauda, Abinash Kumar Pala, Jagamohan Das, “Design and Implementation of High Speed 4x4 Vedic Multiplier” Volume 4, Issue 11, November 2014

[5] Behrooz Parhami, Computer Arithmetic, Algorithms and Hardware Design Oxford University Press.2000

[6] Heideman, M. T. Johnson, D. H. Burrus, C. S. ”Gauss and the history of the fast Fourier transform”. IEEE ASSP Magazine. 1 (4): 14–21.

[7] Strang, Gilbert (May–June 1994). *”Wavelets”* American Scientist. 82 (3): 253. JSTOR 29775194.