A Survey on Floating Point Arithmetic Logic Unit

Shaikh Shoaib Arif
Research Scholar
Dr.B.A.M.U. Aurangabad
Maharashtra, India

Godbole B. B., PhD
Associate Professor of KBPCEP
Satara Shivaji University Kolhapur

ABSTRACT
Floating-point operations are of great use for many computing applications involving large dynamic range, but importantly it needs more resources as compared to integer operations. The progressive demand in FPGA innovation makes such gadgets progressively alluring for designing FP units. With the expanding limitations on delay, more attention is being given to configuration of quicker FP units. To improve speed a wide range of mechanisms/methods are being utilized for the designing of FPU (Floating point math unit) with the aim of reducing latency, area, power consumption and increasing the throughput. Some of the algorithms are presented in this paper, which enlightens the above-mentioned aim.

Keywords
Floating Point Adder, Floating Point Subtraction, Floating Point Multiplier, Floating Point Square, Vedic Sutras

1. INTRODUCTION
The floating-point unit (FPU) was developed for complex and precise point operations. Floating point arithmetic unit (FPU) has a key importance in various computational and scientific applications such as 3D graphics, High performance computing, Digital signal processing etc. The FP operations have discovered numerous applications in different fields for the needs of highly valuable operations because of its incredible dynamic range, highly precise nature and simple working rules. The designing and exploration of the FPUs has been highly focused. The fast progress of very large scale integration (VLSI) technique drove the gadgets like Field Programmable Gate Arrays (FPGAs) to be the best alternatives to execute and implement FPU operations. FPGAs offer decreased time and costs than some particular IC applications.

1.1 Vedic Mathematics
Vedic arithmetic is a piece of 4 Veda [1]. "Vedic" is gotten from "veda" which implies the storage facility of all information. It is a subpart of Veda which holds clarification of a few scientific computations like integration, algebra, quadratics, geometry etc. Vedic science is fundamentally in light of 16 formulas (sutras) managing different branches of maths. These algorithms (formulas) with their brief meaning are given as below [2]:

1. Anurupye Shunyamanyat– If one is in ratio, the other is zero.
2. Chalana-Kalanabyham– Differences and Similarities.
3. Ekadhikina Purvena– By one more than the previous one.
4. Ekanyunena Purvena– By one less than the previous one.
5. Gunakasamuchyah– The factors of the sum is equal to the sum of the factors.
6. Gunitasamuchyah– The product of the sum is equal to the sum of the product.
7. Nikhilam Navatashcaramam Dashatah– All from 9 and the last from 10.
8. Paraavartya Yojayet– Transpose and adjust.
9. Puranapuranabyham– By the completion or non completion.
10. Sankalana-vyavakalanabhyam– By addition and by subtraction.
11. Shesanyankena Charamena– The remainders by the last digit.
12. Shunyam Saamyasamuccaye– When the sum is the same that sum is zero.
13. Sopaantyadvayamantyam– The ultimate and twice the penultimate.
14. Urdhva Tiryakbyham– Vertically and crosswise.
15. Vyashtisamanstih– Part and Whole.
16. Yaavadunam– Whatever the extent to fits deficiency.

1.2 FPU operations
In digital technology, FP is the mathematic representation that is close to a real number in order to make an exchange amongst range and accuracy. The vast majority of the FP numbers are shown to as

$$(-1)^s 2^e (1+f)$$

Where, sign bit is denoted by s
exponent is denoted by e
mantissa is denoted by f

The sign bit is a very basic logic. 0 means a positive number, and 1 signifies a negative number. Flipping the estimation of this bit changes the sign of the number. The exponent must exhibit both positive and negative values. In order to actualize this, a bias is added to the real exponent with a specific end goal to get the stored exponent. An exponent can be negative as well as positive, thus it needs some technique for representing negative exponents using unsigned integers. This strategy is called "biasing": a positive number is added to the exponent before it is stored into the FP number. Due to this, the stored exponent is now known as "biased-exponent". For single precision, the exponent is of 8 bits, and a bias is of 127 bits and for double precision, exponent is of 11 bits, and a bias is of 1023 bits. The mantissa, also known as the significant, depicts the precision bits. It comprises an implicit leading bit to the LSB and the fraction bits to the MSB. The IEEE format for single and double precision is shown in fig.1. According to the IEEE standards

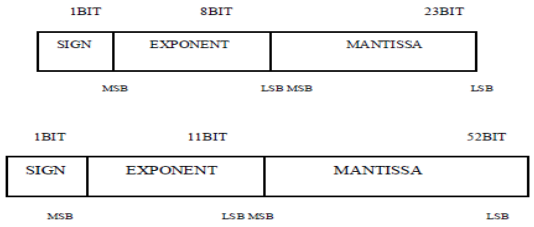

Fig.1. Illustration of single and double precision format [1]

The below table depicts the format for single (32-bit) and double (64-bit) precision FP values. The no. of bits for each

field is shown (square brackets represent the range of bits, 00 = LSB):

Table 1. IEEE 754 Format for Single and Double precision

Precision	Sign	Exponent	Fraction
Single Precision	**1[31]**	**8[30-23]**	**23[22-00]**
Double Precision	**1[63]**	**11[62-52]**	**52[51-00]**

The FPU arithmetic operations consist of addition, subtraction, multiplication and division. The essential approach for addition process:

1. First, convert the two notations into scientific expressions. So that, it can denote the hidden 1.
2. For the addition of two numbers, the exponent of those no's needs to be the same, by aligning the decimal points. The result of this is that the Y is not normalized, but its value is equal to the normalized Y. Add x-y to Y's exponent. Adjust the radix point of the Y to compensate for the change in exponent. Shifting the mantissa left by 1 bit decreases the exponent by 1 and it's vice versa increases the exponent by 1.
3. Add X's two significant fused with modified Y.
4. If the preceding stage of addition do not contain single bit of value 1, then the exponent and radix point needs to be shifted until it achieves the value.
5. Now normalize the result if hidden bit is "1" and de normalize if the hidden bit is "0". Change it again into the 1 byte FP expression.

Similar is the steps (i.e. aligning, normalization, shifting and rounding) for subtraction, multiplication and division.

Aligning: In order make the exponents of the two numbers to be same, the proper alignment of decimal point is mandatory.

Normalization: If traditional8 bit scheme is used. For example as shown in fig.2, i.e. 1 sign bit, 3 integer bits, 3 exponent bits then a single fractional bit is left and hence it can be represent largest value of integer and exponent up to 111 and largest fraction value 0.5 because any other fractional value is not possible since it is left with only 1 fraction bit. Thus, this scheme is not suitable concerning precision. Thus if it is swap this scheme a little and have three fractional bits and 1 integer bit, it can represent various fractional parts. The technique of permitting a single bit for the integer part is called "Normalization". This approach results in precision on the loss of a few bits.

Shifting: Shifting (moving) the mantissa is an important task in order to compensate the change in exponent. Moving the significant 1 bit towards left, reduces the exponent value by 1 and moving the significant 1 bit towards right, increases the exponent value by 1.

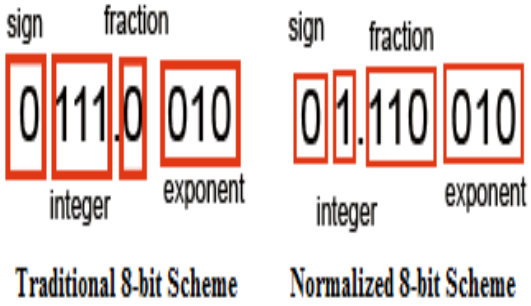

Fig.2. Normalized 8-bit scheme

Rounding: With the allotted amount of precision, the FPU results can't be processed. Hence, the rounding comes into picture. These results need to be rounded off. There are 3 strategies for rounding:

1. Round towards 0: In this approach need to discover what number of bits are accessible. Take that number of bits as the outcome and discard the remaining. The impact of this is similar to making the value more like 0.
2. Round towards $+\infty$: Without considering the value, round towards $+\infty$.
3. Round towards $-\infty$: Without considering the value, round towards $-\infty$.

But these steps affect various parameters such as area, latency, power consumption, throughput etc. As to implement these functions extra circuitry i.e comparator, data extractor, SWAP circuitry, shifter [3] etc. would be required which increase the requirement of area on FPGA chip. Increase in the circuit will definitely demand more power which in turn increase the power consumption. The differences in exponent comparison and mantissa alignment, the control logic creates delay which in turn leads to the latency in system. As latency decides the throughput, because of the latches and control signals in pipeline architecture, the area and power utilization are rises higher than the non-pipelined model.

2. APPROACHES FOR HIGH SPEED ARITHMETIC OPERATIONS

ADDITION/SUBTRACTION

The maximum used computation in FPUs is addition. Thoughtfully, it is the most basic computation i.e. computing either the addition or subtraction of given FP numbers. But practically, alignment, normalization, shifting and rounding which might be a necessity can make the floating point adders quite slow. A quick FP adder is essential to the execution of a FPU, and in this way methods to higher the rate of FP addition is presented in many surveys which is discussed below. Fig.3 demonstrates the general sum/difference process.

Many individual operations are involved in FP addition/subtraction process. Higher performance can be reach by decreasing the no. of executions. Further, discuss the related work done by the experts as follows:

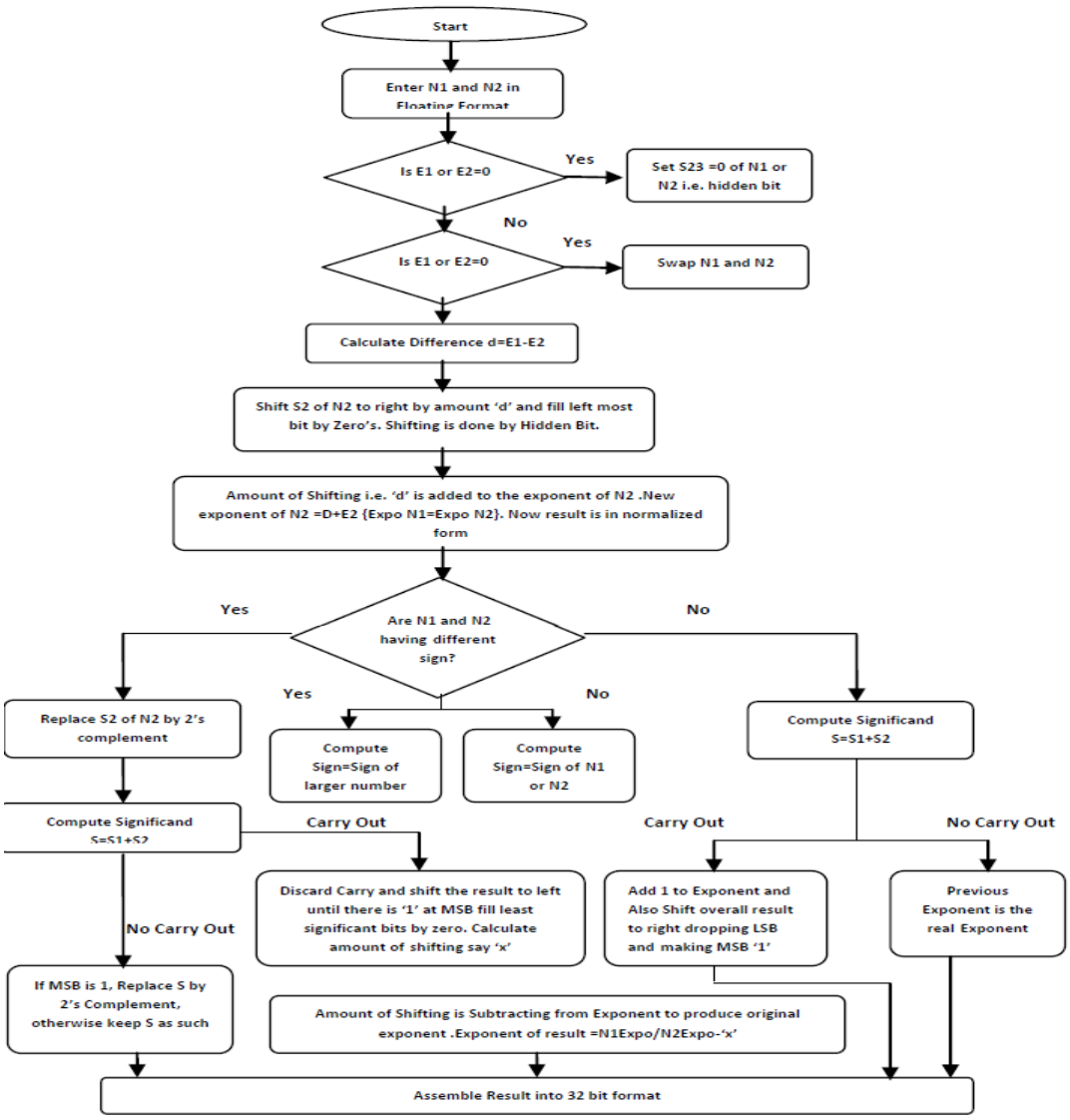

Fig.3. Floating Point Addition/ Subtraction [4]

The survey reveals different algorithms/ techniques used for the implementation of adder/subtractor with reduced latency on FPGA. Optimization procedures empowers inventor to show how these techniques can be assembled to accomplish a general quick FP adder layout. Generally, impressive decrease in latency through simultaneous ways needs the delay to be adjusted. The FP adder layout given in [5] accomplished a low latency by combining different enhancement procedures.

To maximize the adder's throughput, a standard method is to pipeline the unit such that every pipeline stage contains the small atomic operation. Whereas, the FP summation may need many cycles to give the result, another operation can start every cycle, giving high throughput [6]. The benefit of the pipeline design process is that, in spite of having a high input and output successive length, it gives an unmatched throughput [7]. A double precision FPU is presented in this paper [7] which when compared with single precision FPU provides rapidness and more preciseness. Even a pipelined algorithm with a packet forwarding approach is also used to implement the adder in [8]. In this new approach addition and rounding is responsible for a 4-stage pipeline execution with a minimum clock period.

The paper [9] presents a combined FP 3-term-adder which executes 2 summations in one unit to accomplish more satisfactory execution and more satisfactory precision contrasted with a system of conventional FP 2-term adders. With the objective of enhancing the execution of the 3-term adder, a few improvement methods are utilized including another exponent compare and mantissa alignment and normalization, 3 input driving zero anticipation, compound expansion and pipelining. This adder decreases the area and power utilization by around 20% and latency by around 35%.

Architecture of a double precision (DP) adder is proposed in [3], which bolster a dual SP feature. As to implement these functions extra circuitry i.e. comparator, data extractor, LOD, SWAP circuitry, shifter etc. would be required. LOD algorithm is implemented in this adder/subtractor design.LOD approach is described by its clarity and decreased area. To accomplish high frequency, the layout was profoundly pipelined. After the sum/difference process is executed, the consequent mantissa is normalized through LOD technique. The leading-one-detector identifies the MSB '1' from the quantity of no. of zeros (nz) prior to MSB '1'. The mantissa is then moved left "nz" times [10].

3. MULTIPLICATION AND SQUARE:

The second most used FP operation is multiplication. Thus, quick multiplication is additionally crucial task for fast FPUs. The summation of a many partial products is included in a multiplication process, each of which is a result of the multiplicand and 1 bit of the multiplier. Partial product generation (carried out by AND logic operation and Booth's algorithm), partial product reduction (carried out by adders which are connected in different topologies), and final carry-propagate-addition (this is an integer addition so it doesn't involve FP operation) are the main steps that are involved in multiplication process [6]. Fig.4 shows the general execution of FP multiplication process.

The multiplication algorithm used in Vedic Mathematics is known as "Urdhva Triyagbhyam" (introduced earlier in section II, sutra 14).The same thought is utilized for FPU arithmetic's to make it suitable for computer hardware. This is the general rule applied for all multiplication calculations.

Urdhva implies Vertical and Triyagbhyam implies Crosswise. Different methodologies are proposed for rapid vedic multiplier.

Some methodologies are mentioned in [2], one of these is Vedic Multiplier With

Ripple Carry Adder. In this methodology, three 4-bit ripple carry adders are utilized.

While the execution, the carry originating prior to the adder requires some time to travel which makes the carry ripples and full adders wait for its arrival.

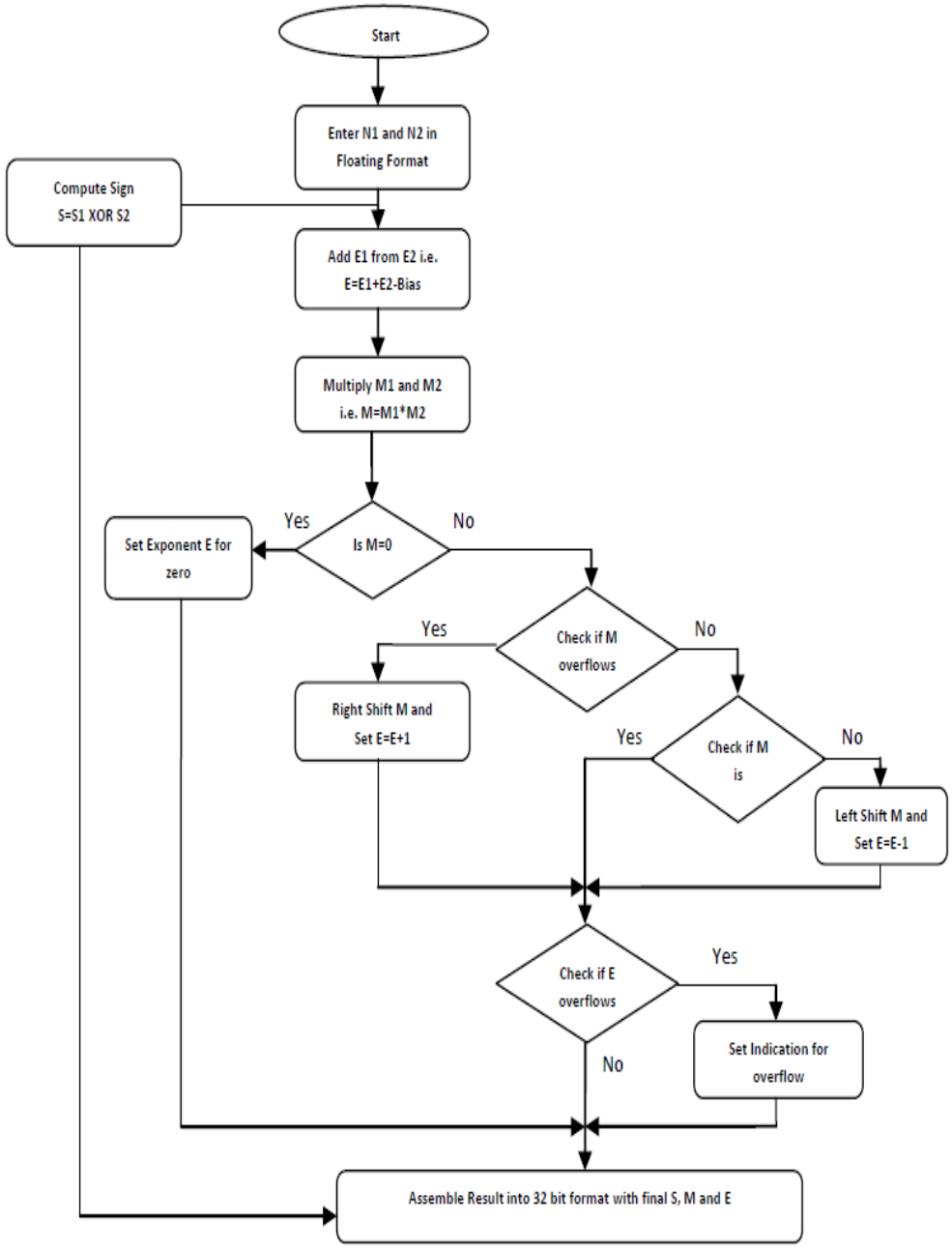

Fig.4. Floating Point Multiplication [4]

This, thus causes decrease in operating time. Second is the Low Power and High Speed Vedic Multiplier. It is a 16-bit multiplier layout which substitutes the with the ripple carry adder with the carry Look ahead Adder. As a result, use of

Look ahead adder increases speed and reduces power dissipation. Third is Fast Vedic Multiplier Using Carry Save Adder. In this Carry Save Adder takes little move include and enhances speed because in Carry Save Adder, summation is done simultaneously without sitting tight for the outcome. Fourth is Speed Efficient Design for Vedic Multiplier. It uses another tree structure for addition of partial products. This layout produced better results in contrast to other architecture. It has brought about a less delay time. Fifth is a methodology utilizing 7:2 Compressor Adders as a part of which 4:2 compressors and 7:2 Compressors are utilized for 4-bit and 8-bit multiplication. A 7:2 Compressor is composed with the assistance of changed 4:2 Compressor which has been utilized to implement 8X8 multiplier. Such a layout requires just 12 parallel stages while traditional Vedic multiplier requires 15 stages.

This paper[11] presents 2x2 High-Speed matrix multiplier. Here every matrix element is exhibited by 16-bit. Hierarchical structuring concept is used for designing matrix multiplier. It uses the structural modelling which permits the less calculation time for computing the multiplication outcome. This in turn provides an opportunity for modular layout where bigger blocks can be made from smaller one.

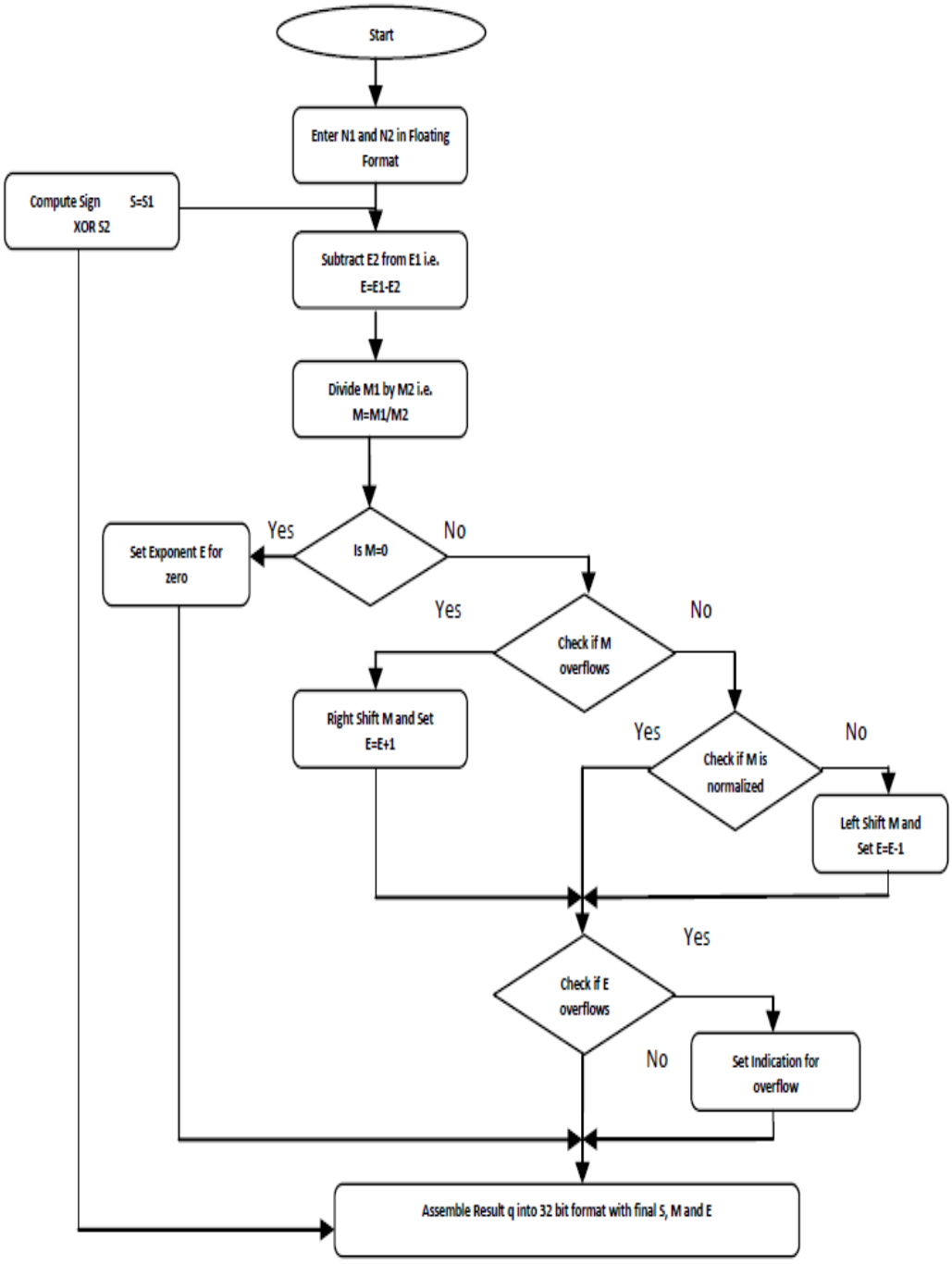

Fig.5. Floating Point Division [4]

A Vedic mathematic *squarer* is proposed in [12]. The proposed squarer works on the basis of Ekadhikena Purvena algorithm(introduced with its meaning earlier in section II, sutra 3). This algorithm is applicable only to find the squares

of decimal numbers ending with "5". Following example depicts this algorithm in depth.
Such as. : 35^2=1225
1. The right part of answer is 25 as it is the square of '5'.
2. According to the formula the 3 is the precedent of '5' and one more than that is '4'. Hence, using formula $n(n+1)=n^2+n$ (i.e. 3×4=12) it is the left part of the answer.

When the proposed squarer was compared with the duplex squarer, it showed better performance than the duplex one. It saved 50% area and also reduced delay by 50%.

4. DIVISION AND SQUARE ROOT

Division and Square Root are very least frequently occurring operations in FPU. A slow divider in FPU can degrade the performance of FPU. The sq. root operation is quite alike to division and therefore many division rules are applicable for sq. root computation. Division calculations can be separated into five classes: digit repetition, functional cycle, high radix, table look-up, and flexible latency [6]. Fig.5 shows the general execution of FP division process.

Division is constantly thought to be massive process and a one amongst the most troublesome computation in number system and thus every application of division calculations in VLSI engineering have high time and space complexities. On the basis of Vedic maths, all work on division strategies, is created for number system with base of 10.The logic which may suit base 10 (decimal) is not feasible for base 2 (binary) number system. Here an another methodology is proposed in this paper [13] for a new division for base 2 number system. Optimized binary division architecture is presented which uses Vedic Mathematics sutras i.e. Nikhilamalgorithm, Dhwajankalgorithm and Paravartyaalgorithm (introduced earlier in section II, sutra 7 and 8 resp.). After the analysis of these sutras individually, Tadas proposed the division algorithm which involves these sutras combine. This results in an efficient division process.

Reciprocal approximations and division assumes a vital part for many applications like DSP and image processing and so on. This technique [14] is particularly prudent when distinct dividend is to be partitioned by the same divisor. These 'reciprocal approximation' strategies are based on Newton-Raphson iteration strategy [6]. The reciprocal algorithm's flowchart is presented in [14]. Last digit of the denominator has been computed through mod-10 operation. If last digits are 2, 4, 6 then it is multiplied by 5 and if the last digit is 5 then it is multiplied by 2. On the off chance that, the last digit is 3 then it is multiplied by 3. On the off chance that the last digit is 0 now it can execute the right move operation directly. On the off chance that last digit is 1 or 7,8,9 then it can execute the immediate appliance of the approach. The multiplication procedure will proceed until the digit lessens in the denominator stage. The calculation will proceed until 16 FP numbers [14].

5. APPLICATIONS

Many practical control applications in the industrial fields advantage from the dynamic extent and accuracy of FP offered by the floating point arithmetic unit (FPU). Due to its vector processing capability, it offers an extraordinary interest for FPU in Image processing, for example, scaling, textual printing, 3D transforms, 3D Graphics, Laser printers, digital cameras, digital video cameras and filtering in graphics. It has a wide demand in digital processing as well such as automotive control applications, digital set-top boxes, gaming consoles and motion controls etc.

Multipliers/Squares are the most widely used functions in DSP applications which are used to perform different calculations. The throughput of the framework significantly relies upon the execution of these operations. Henceforth these functions should work in a streamlined way devouring less area and power, and working at higher pace. With the development of innovation, a fast executing processor is a necessity. Multiplication is a basic computation of DSP applications(like DFT, FFT, convolution and so forth), Arithmetic and logic unit (ALU), and Multiply and Accumulate(MAC) unit. The convolution is an essential part in DSP and Image Processing. Convolution is vital idea for outlining the finite IR filter, DFT and FFT. Different DSP operations on basis of vedic science are executed in [15]. These are linear and circular convolution, correlation, cross-correlation, autocorrelation etc.

6. RESULTS STUDIED

The simulation results as shown in Fig. 6 from [1] shows that when two inputs 'A' (unsigned FP number) and 'B' (signed FP number) are taken with values 134.0625 and -2.25 respectively, then its individual conversion to hexadecimal format gives 0x43061000 and 0xC0100000 respectively. The result of multiplication of these two hexadecimal numbers yields the outcome 0xC396D200 which when converted into integer format is valued equal to -301.640625. In [2], a 8X8 bit multiplier shown in Fig. 7 has been designed using 7:2 compressor which is a modified version of 4:2 compressor. The vedic multiplier works on fifteen stages on the other hand, the compressor based multiplier requires only 12 stages which improves the speed by the factor of 1.12 as compared to vedic multipliers, 2.112 as compared to Booth and 1.509 as compared to modified Booth multiplier.

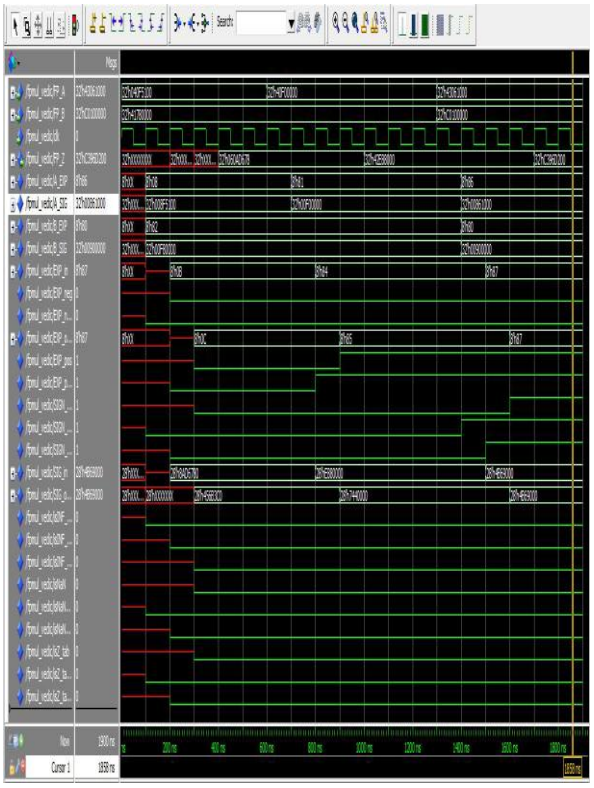

Fig.6.Simulation Result of vedic multiplier

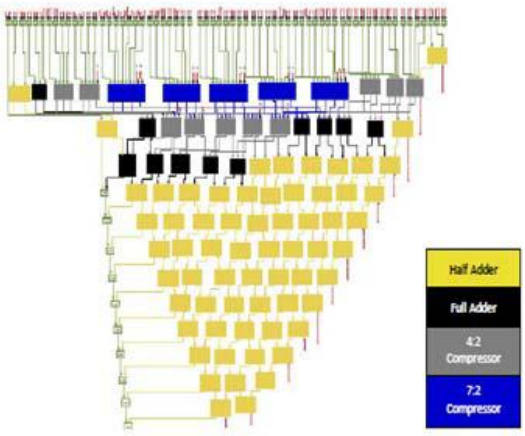

Fig.7. 8X8 bit compressor based multiplier

In [3], it is seen DPdSP adder which can work in both double and single precision mode. The Double precision or dual Single precision (DPdSP) can operate in Double precision mode and can also operate parallel in dual Single precision mode. It can perform almost all computations such as rounding-off etc. The multiplexing circuit required for tuning is minimum. As a result when compared to other multi precision techniques, DPdSP architecture gives 37% reduced area and 40% to 50% lessened area-delay product but to achieve this it requires 15% extra resources as compared to double precision. The comparison results are shown in the table below. This architecture provides better computation.

Table 2. Comparison Results with other techniques

	0.25µm	**0.11 µm**		**0.18 µm**	
Latency	3	5	3	1	4
Area OH Period OH	24% 9%	26% 9.6%	33% 13.3%	15% 7%	17% 3.5%
Scaled Area Gate Count	- 13224	0.370 -	0.433 -	0.164 10288	0.172 10794
Period Total Delay	49 140	18 65	24.7 55	87 28.4 87	
Area X Delay Area X Delay	- 1851360	24.05 -	23.815 -	14.268 895056	

Table 3 shows the results of fused floating point 3-term adder from [9]. Traditional fused FP adder gives accuracy with reduction in area, power consumption (5–8%) and latency (3–14%). Proposed fused FP 3-term adder gives reduction in area and power consumption by15–20% and reduction in latency by about 35%.

Table 4. Pipeline fused FP 3-term adder

Single Precision			
	Stage 1	**Stage 2**	**Stage3**
Area(µm$^{2)}$	5,800 (37%)	6,400 (41%)	3,5000 (22%)
Latency (ns)	0.52 (33%)	0.54 (34%)	0.52 (33%)
Power (mW)	2.89 (35%)	3.35 (41%)	2.01 (24%)
Double Precision			
	Stage 1	**Stage 2**	**Stage3**
Area	12,500 (36%)	14,600 (42%)	7,600 (22%)
Latency (ns)	0.62 (32%)	0.64 (34%)	0.62 (32%)
Power (mW)	6.43 (35%)	7.59 (42%)	4.26 (23%)

Table 3. Traditional fused FP 3-term adder

Single Precision					
	Discrete	**Traditional Fused 1**	**Traditional Fused 2**	**Improve Fused**	**Improved + Pipeline**
Area (µm^2)	18,200 (100%)	17,300 (95%)	17,000 (93%)	14,900 (82%)	15,700 (86%)
Latency (ns)	2.24 (100%)	2.20 (98%)	1.98 (88%)	1.46 (65%)	1.62 (72%)
Throughput (1/ns)	0.45 (100%)	0.46 (102%)	0.51 (113%)	0.68 (153%)	1.85 (415%)
Power (mW)	9.46 (100%)	9.17 (97%)	8.90 (94%)	7.91 (84%)	8.25 (87%)
Double Precision					
	Discrete	**Traditional Fused 1**	**Traditional Fused 2**	**Improve Fused**	**Improved + Pipeline**
Area (µm^2)	41,200 (100%)	38,200 (93%)	37,900 (92%)	33,000 (80%)	34,700 (84%)
Latency (ns)	2.80 (100%)	2.72 (97%)	2.40 (86%)	1.76 (63%)	1.92 (69%)
Throughput (1/ns)	0.36 (100%)	0.37 (103%)	0.42 (117%)	0.57 (159%)	1.56 (438%)
Power (mW)	21.46 (100%)	20.30 (95%)	19.95 (95%)	17.52 (82%)	18.28 (85%)

Table 4 [9] shows the results of pipelined fused FP 3-term adder into three pipeline stages. For control management between stages latches are necessarily needed. The latency among the 3 stages is maintained such that the throughput is increased by the factor of 2.7.

In [11], a 2X2 High speed multiplier has been designed on the basis of "UrdhavaTrigyagbhyam" logic. The coding has been done in VHDL software and synthesized using Xilinx. The simulation of 2X2 High speed multiplier is shown in Fig. 8.

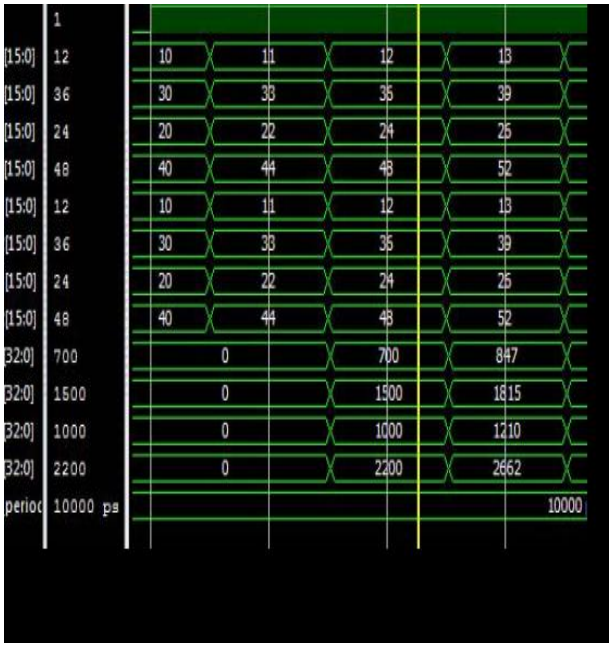

Fig.8.Simulation Result for 2X2 High speed multiplier

The FPGA implementation of this design gives an improved structure with reduced area (shown in Fig. 9), less computation time, less area to speed ratio (shown in Fig. 10) and operating at higher running frequency 373.692MHz (shown in Fig. 11) with the requirement of a single global clock cycle.

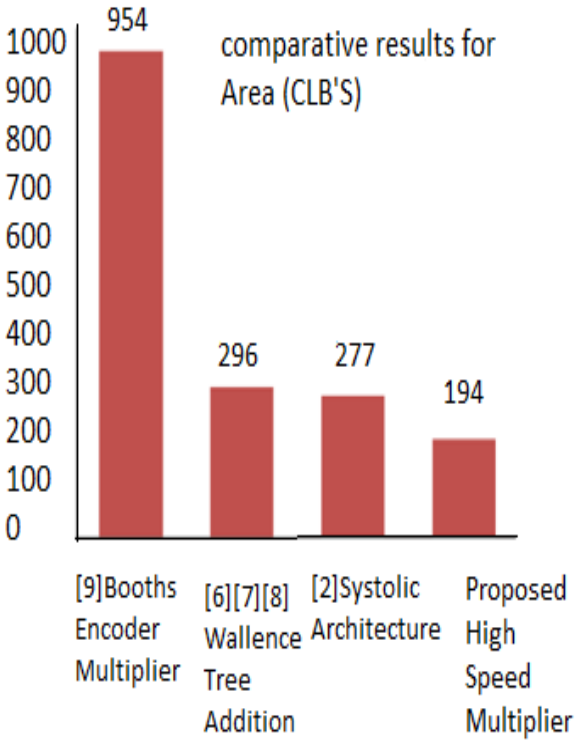

Fig.9.Comparison Result for Area

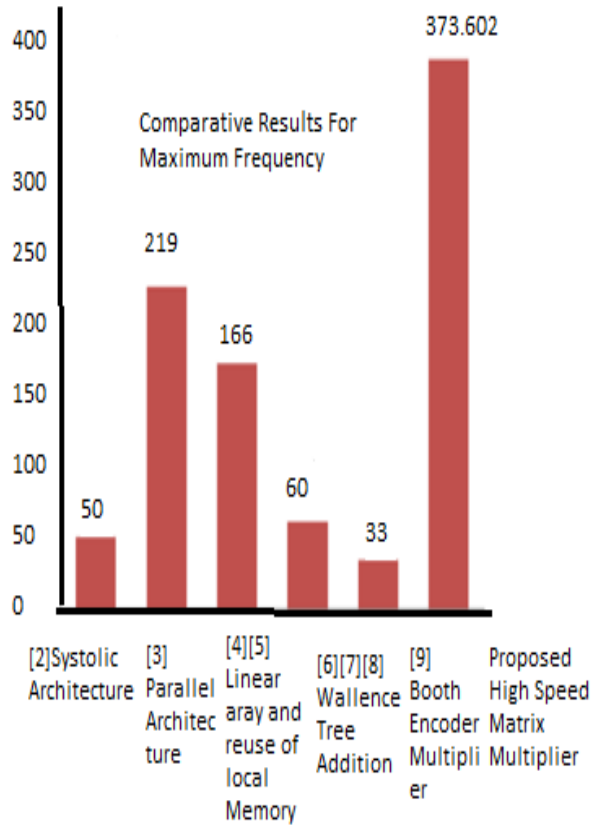

Fig.10.Comparison Result for maximum Frequency (MHz)

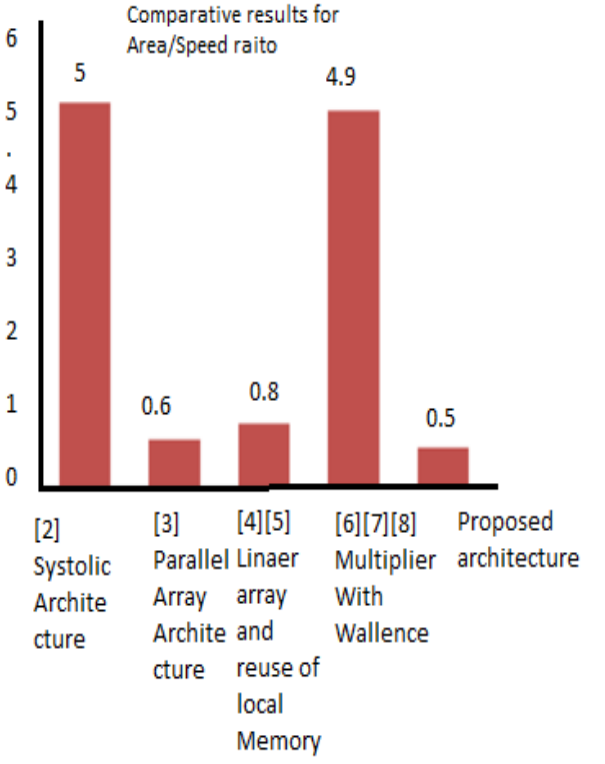

Fig.11.Comparison Result for Area to Speed Ratio

7. CONCLUSION AND FUTURE

In this survey paper various floating-point arithmetic techniques to reduce the area, power consumption by and reduces the latency, delay optimization based on different vedic sutras, Pipeline packet forwarding, fused floating point

three term adder, high speed multiplier divider, squarer. Out of which Urdhva Tiryakbhyamalgorithm proved to be an encouraging strategy concerning speed and area. EkadhikinaPurvena, Nikhilam Dashatah, Paraavartya and Dwanjaka sutras are also very helpful for implementing other FPU operations. In this survey it is seen that multiple FPU configurations i.e. single-precision, double precision, dual-precision adders, multipliers and dividers. For the execution of a combined multiplication-addition operation, the output of FP multiplier should fetched as input to FP adder which will give the required outcome.

Future work includes implementation of these techniques to real time DSP operations like voice morphing to make an IP of voice morphing. It contains the floating point ALU for addition, subtraction, multiplication & divide operations part with minimum power consumption by using techniques discussed in paper and FFT operations for trigonometric part take samples of source and destination voice and converted both to the frequency domain using FFT and extracted features of both. Then the difference between both is calculated, it gives info that how the source is different from the target. Then the full source voice is varied by that difference thus morphed voice to target and work can be validated by using MATLAB tool.

8. REFERENCES

[1] Pratiksha Rai et al, "Design of Floating Point Multiplier Using Vedic Aphorisms", International Journal of Engineering Trends and Technology (IJETT) – Volume 11 Number 3,May 2014.

[2] Yogita Bansal et al, "HIGH SPEED VEDIC MULTIPLIER DESIGNS-A REVIEW", IEEE Proceedings of 2014 RAECS UIET Panjab University Chandigarh, 06 – 08 March, 2014.

[3] M. Jaiswal, "Unified Architecture for Double/Two-Parallel Single Precision Floating Point Adder", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS 521, 2014.

[4] N. Grover and M.K. Soni, "Design of FPGA based 32-bit Floating Point Arithmetic Unit and verification of its VHDL code using MATLAB", I.J. Information Engineering and Electronic Business, 2014.

[5] Peter-Michael Seidel, Guy Even, "Delay-Optimized Implementation of IEEE Floating-Point Addition", IEEE Trans. on Computers, vol. 53, no. 2, pp. 97-113, Feb. 2004.

[6] Stuart Franklin Oberman, DESIGN ISSUES IN HIGH PERFORMANCE FLOATING POINT ARITHMETIC UNITS, Technical Report, Stanford University California, 1996.

[7] Adarsha KM et al, "Double Precision IEEE-754 Floating-Point Adder Design Based on FPGA", International Journal of Advanced Research in Electrical, Electronics and Instrumentation. Engineering, Vol. 3, Issue 4, April 2014.

[8] A. Nielsen, D. Matula, e.N. Lyu, G. Even, "IEEE Compliant Floating-Point Adder that Conforms with the Pipelined Packet-Forwarding Paradigm," IEEE Trans. on Computers, vol. 49, no. 1, pp. 33-47, Jan. 2000.

[9] J. Sohn, "A Fused Floating-Point Three-Term Adder", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 61, NO. 10, OCTOBER 2014.

[10] Hamid et al,"Design of Generic Floating Point Multiplier and Adder/Subtractor Units", 12th International Conference on Computer Modelling and Simulation, 2010.

[11] S.V. Mogre, "Implementation of High Speed Matrix Multiplier using Vedic Mathematics on FPGA", International Conference on Computing Communication Control and Automation, 2015.

[12] Sriraman, "DESIGN AND FPGA IMPLEMENTATION OF BINARY SQUARER USING VEDIC MATHEMATICS", IEEE 4th ICCCNT, July 2013.

[13] Aditi Tadas, "64 Bit Divider using Vedic Mathematics", 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), May 2015.

[14] Saha, "Reciprocal unit based on Vedic mathematics for signal processing applications", International Symposium on Electronic System Design, 2013.

[15] A. Itwadiya, "Design a DSP operations using Vedic Mathematics", International conference on Communication and Signal Processing, April 3-5, 2013.