Realization of FIR Filter using High Speed, Low Power Floating Point Arithmetic Unit

[1]Srikanth Immareddy, [2] Sravan Kumar Talusani, [3]Rayavarapu Prasad Rao
[1] *Electronics and Communication Engineering Dept, Methodist College of Engineering and Technology, Hyderabad 500001, India. Email: siri.vlsi@gmail.com.*
[2] *Electronics and Communication Engineering Dept, Methodist College of Engineering and Technology, Hyderabad 50001, India. Email : sravankumartalusani@gmail.com.*
[3] *Electronics and Communication Engineering Dept, Avanthi Institute of Engineering and Technology, Viskapatanam, India. Email:prasadrao.rayavarpu@yahoo.com.*

***Abstract* - In Digital Signal Processing, filtering is one of the major task, where the inputs to the filter are floating point numbers. This paper discuss about realization of a digital Finite Impulse Response (FIR) filter using high speed, low power floating point arithmetic unit on an Field Programmable Gate Array. High speed is achieved by using a modified normalization unit along with ripple carry adder. A new array multiplier using the concept of carry generation and propagation is used, which reduces the power consumption. The average speed and power requirements of implemented filter are compared with a conventional FIR filter.**

***Keyword:* Finite Impulse Response (FIR) Filter, Floating Point Arithmetic Unit(FPAU), Normalization unit, Array multiplier.**

I. INTRODUCTION

Most of the computer applications in recent years need complex computations which demand for accurate results. Example for these kinds of applications includes graphical applications and Digital Signal Processing applications which resulted in the need of including Floating Point Arithmetic Unit (FPAU) [1]-[2]. In Filter design the FPAU is an important block where addition and multiplication operations are frequently used.

Speed is considered as the primary parameter and the power as secondary parameter in designing digital systems nowadays. The feature size of the integrated circuit is reducing almost at nano scale, the motive behind the miniaturization is to improve the speed, but this higher level of integration is increasing the power consumption [3]. Hence heat removal and low power dissipation are the primary goals of the designer and also enhancing the speed of the integrated circuit.

Power reduction can be achieved using different levels of abstraction of a VLSI design. This includes fabrication process level, circuit design level, algorithm level and architecture level. In algorithm level this is achieved with a slight modification in the existing algorithm [4]-[5]. Reduction of threshold voltages will improve the power reduction at circuit level. Parallel and pipelined architectures can be used at architecture level.

A parallel FIR filter is implemented using even symmetric coefficients reducing the number of multipliers, where the multipliers are implemented using adders reducing the hardware requirements [6]. A carry save adder (CSA) is used in implementing FIR filter that gives High Speed and Low power constant multiplier, where the FIR filter is used in signal processing applications [7]. An FIR filter is designed using the concept of faithfully rounded truncated multipliers, where the emphasis is on the low cost [8]. A modified multiplication technique is proposed that uses redundant multiplication of higher order bits avoided by separating multiplication into higher and lower parts that can reduce the power dissipation in FIR filters in [9]. The aim is to design a FIR filter, with low power consumption and high speed, which can be achieved using a special floating point arithmetic unit.

The logic implementation style of an adder can vary the power dissipation, in order to get high performance and low power, a Bridge style adder is proposed in [10], that uses the concept of high degree of regularity. Hybrid carry-select modified tree adder architecture is investigated to minimize power consumption using multiplexers is proposed in [11]. A carry-look ahead adder technique is used in describing leading-zero count, significantly reduces power consumptions in both static and dynamic CMOS Logic [12]. A scalable leading zero detector is described in [13]. The pass transistor logic is used in leading zero detector design to yield high speed at the cost of high power dissipation is proposed in [14]. The ripple carry adder is selected in this work as it consumes low power than their counterpart adders; the normalization is modified to achieve the speed in the floating point addition.

There are various high-speed multipliers proposed but it should be recognized that power consumption needs to be reduced when there is a tradeoff between speed and power as given in [15]. The multiplication operation can be performed using a Booth multiplier algorithm where the addition can is performed using ripple carry adder, dissipating less power than carry- look ahead adder [16]. The majority of these floating point ALUs can handle at most 32-bit wide floating point numbers particularly, ALUs introduced in DSP processors intended for audio applications [17]-[18]. There are different levels of abstraction where the Low-power multipliers have been studied. In [19]-[20], architecture level method is proposed where clock gating few functional units of a multiplier is carried out. To do the reliable design evaluation, each and every detail of the arithmetic elements needs to be considered rather than going at a high-level approach. For example, the assumption that power consumption in multipliers proportionally varies with the width of data path is not true always [21]. An array multiplier is selected in this work as it is having low hardware requirements which reduce the power consumption.

978-1-479-7678- 2/15/$31.0 ©2015 IEEE

II. FLOATING POINT ARITHMETIC UNIT

The Floating point Arithmetic Unit defined in this paper operates on floating point numbers and performs operation such as addition and multiplication. IEEE 754 standard is followed to represent floating point numbers as shown in fig.1 [22]. Here in our discussion we use Single- precision floating point number formant, which consists of three fields Sign bit(s), Biased exponent (e) and Mantissa (m). The single-precision number format has 1-bit sign, 8-bit exponent, and 23 bit mantissa. Floating point numbers cover a wide range compared to fixed point numbers. But the complexity of implementation is more.

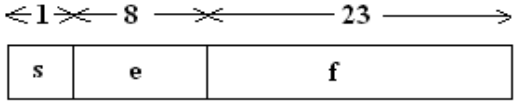

Fig.1 IEEE 754 Single-precision floating point number format.

A. FLOATING POINT ADDITION

The floating point addition can be performed in five stages they are Exponent difference, pre-alignment, addition, normalization and rounding off stage. Let X1, X2 be two floating-point numbers X1 = (S1, E1) and X2 = (S2, E2). The five stages for addition are.

1. Finding the difference between exponents i.e. D = E1-E2. If D < 0, then swap the mantissas, the resultant exponent is the larger exponent among E1 and E2.
2. Pre-aligning the mantissa with lesser exponent by shifting it right by D bits.
3. The temporary mantissa is obtained by addition or subtraction of the mantissas based on sign bits of the operand.
4. Normalization is performed to represent the operand in floating point IEEE 754 standards.
5. Rounding off is performed on the resultant mantissa. The Fig. 2 explains the addition algorithm.

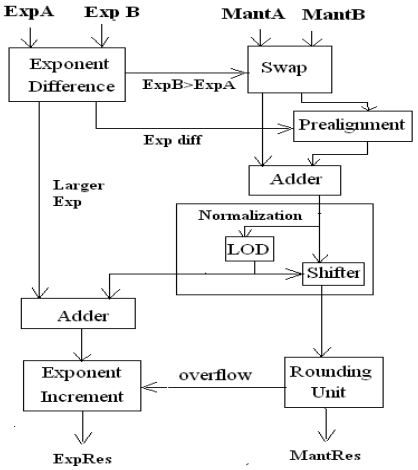

Fig. 2 Floating point Addition data path.

A ripple carry adder is used as adder in FPAU, which consumes less power among all the adders, and the ripple carry adder is simple and slower compared to their counterparts [16]. The Fig.3 is a ripple carry adder used for addition of two 4 bit numbers.

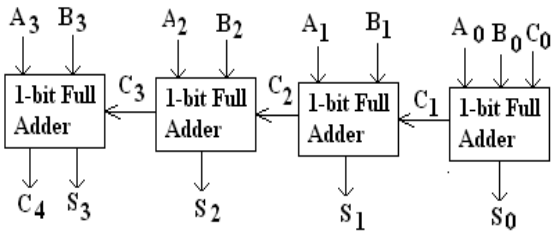

Fig .3 Ripple carry adder.

B. FLOATING-POINT NORMALIZATION UNIT:

Normalization unit is an important stage in the floating point addition. Where leading zeros are counted till a first one bit is encountered. A barrel shifter is used to left shift the data depending on the count. The Fig.4 shows about a normalization unit. A modified normalization unit is designed based on partition method is discussed in the next section.

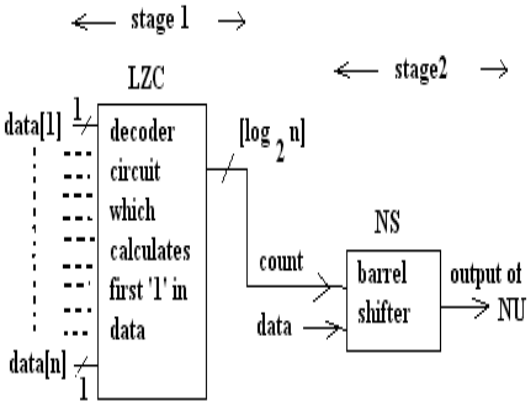

Fig.4. Conventional Normalization Unit.

C. FLOATING-POINT MULTIPLICATION

Normally the floating point multiplier has a complicated circuit than an adder circuit. Let us assume the floating-point numbers X1 = (s1, e1, f1) and X2 = (s2, e2, f2), the resultant number Xm = (sm, em, fm) is calculated below in three steps.

1. Calculating the resultant sign sm= s1 xor s2. And exponent em= e1+e2. Multiply the mantissas fm=f1 * f2.
2. If the result from the multiplier overflows normalization is performed.
3. The mantissa is shifted right by '1' bit and the exponent is incremented by '1' in case of result overflows, in the rounding off step.

The Fig.5 is the algorithm of floating point multiplier. An efficient multiplier improves the overall efficiency of the floating point multiplier.

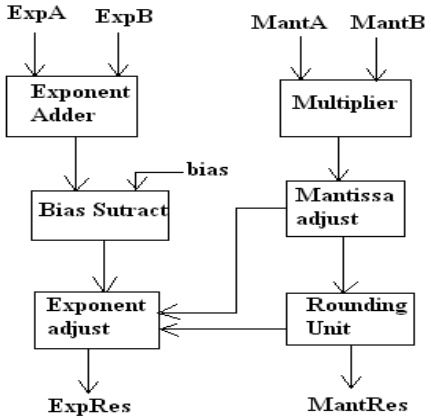

Fig .5. Floating-point multiplier data path.

An array multiplier is used that consumes less power when compared to all other multipliers [20]. It is simply a shift-add algorithm. The Fig.6 is an array multiplier having two inputs of 4 bit size and an output of 8 bit. A modified array multiplier is used in this paper is explained in this next section.

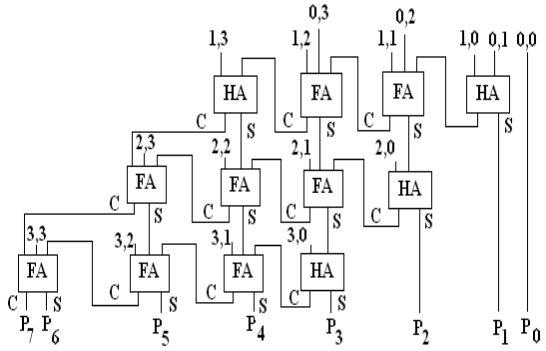

Fig. 6 Conventional 4-bit Array Multiplier.

III. MODIFIED NORMALIZATION UNIT

The modified Normalization Unit uses the concept of partition the input data. The modified floating-point normalization unit is shown in Fig 7. There are three stages and the computation procedure is explained below.

In stage -1, the 25 bits of input data is partitioned into 5 blocks, with each block consisting of 5 bits each. Each group of 5 bits data is applied as inputs to the OR gate. The 5 OR gates generates outputs as y1, y2, y3, y4, y5. These outputs are applied as inputs to the priority encoder circuit. The priority encoder gives two outputs Z and count is 5 bit. In stage-2, the position of first one bit among the input Z (output of stage-1) is calculated using the conventional NU method. And the sub count is calculated. In stage-3, the output of stage-1, stage-2 is added. This generates the shiftamount. A barrel shifter is used to reduce the delay. The single precision floating point input considered is having 25 inputs. Hence we select m=k= 5 bits. From the architecture it is observed that count=5 bit, subcount= 3 bit, and shiftamount=5 bit. The barrel shifter is also having 5 bit input.

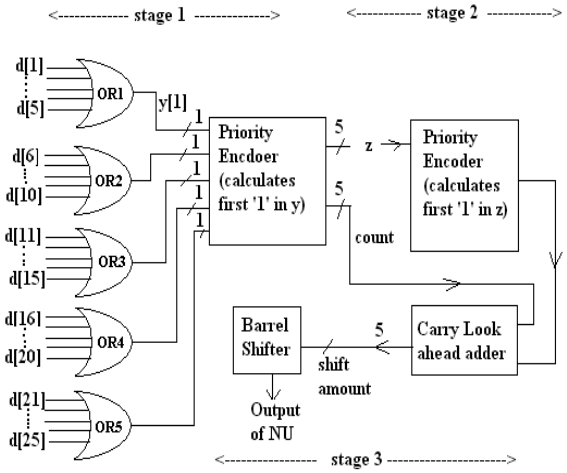

Fig. 7. Modified Normalization Unit.

TABLE I
COMPARISON OF MODIIFIED AND CONVENTIONAL NORMALIZATION UNIT.

Parameter	Conventional NU	Modified NU
Speed (n sec)	39.92	23.63
No. of slices (FPGA)	184	84
Dynamic power(m watts)	34	29

The modified normalization unit using partitioned method shows reduction in average power, number of LUTs and improvement in average speed than the conventional normalization unit. Optimal performance can be achieved when the number of blocks and number of bits in each block are approximately equal. Table I show the comparison between the modified normalization unit and conventional normalization unit.

IV. MODIFIED ARRAY MULTIPLIER

A Conventional 'n' bit array multiplier gives '2n' bits of output. For Floating point numbers the last 'n' bits contribute to the rounding off process. The modified array multiplier does not calculate these 'n' LSB bits. This reduces the hardware resulting in low power design at the cost of rounding off error. To correct the obtained 'n' bit MSB result, only carry is generated and propagated from the LSB side. By which additional hardware required for computing the sum bits is reduced. In this architecture, changes are made in Lower half of the multiplier by using carry generators in place of Half adders and Full adders.

The carry is generated using carry generation half (CGH) and carry generation full (CGF) circuits using the equation 1, 2.

$C_{CGH} = A.B$ (1)

$C_{CGF} = A.B+B.C+A.C$ (2)

The modified arrary multipler design has no rounding off step after calculating the output. The rounding off error is ignored as it is of the order of 10^{-6}. The data path after removing this step looks like fig.8.

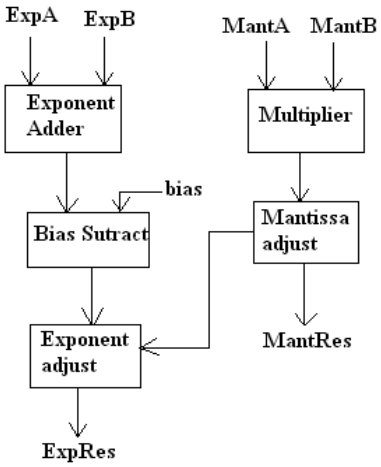

Fig. 8. Modified array multiplier data path.

The modified array multiplier for a single- precision floating point numbers has 24 bit input and output is also 24 bit (MSB), having negligable round off error. The hardware required for generating 24 bit(LSB) is eliminated. The Fig .9 shown below is the block diagram of modified array multiplier, where the generated carry is propagted to the MSB for reducing the error.

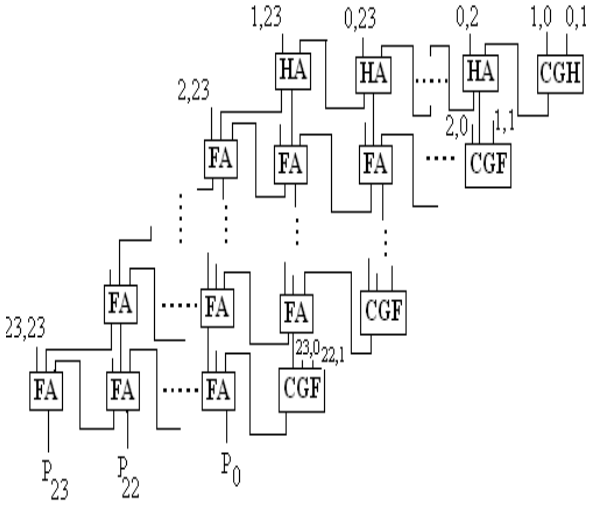

Fig. 9 Architecture of 24 bit Modified Array Multiplier .

In the modified array multiplier architecture as shown in Fig.4.2. The carry generation unit replaces one HA with one CGH, 23 Full adders with 23 CGF. The modified array multiplier is implemented using HDL and simulated. The fig.10 shows the simulations of a sample data.

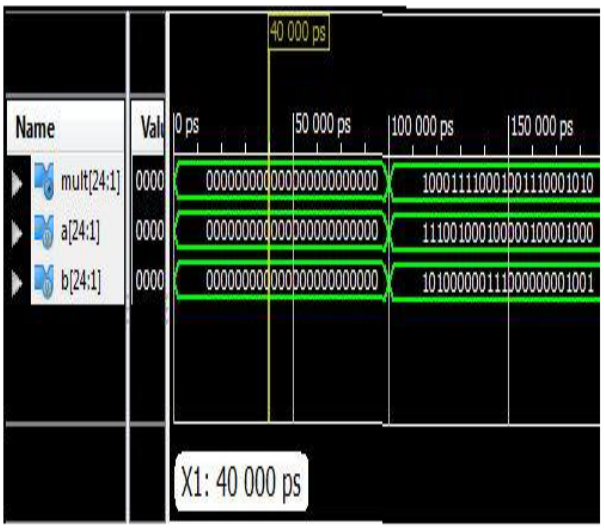

Fig.10 Simulation of Modified Array Multiplier.

Both the conventional and modified array multiplier is synthesized using Xlinix Spartan. Area and timing information is obtained, the power dissipation information is obtained using Xpower Analyzer. Speed and power dissipation are tabulated in Table II.

TABLE II
COMPARISON OF CONVENTIONAL ARRAY MULTIPLIER AND MODIIFIED ARRAY MULTIPLIER

Parameter	Conventional multiplier	Modified multiplier	Percentage improvement
Speed (ns)	75.2	69.97	2.4 %
Dynamic Power(mw)	993	927	6.7 %

The modified array multiplier design for a single precision floating point input shows an improvement in the power saving.

V. PROPOSED DIGITAL FIR FILTER

In digital signal processing, a FIR filter is a filter whose response to any finite length input is of finite duration, because it settles to zero in finite time. The FIR filter is computed mathematically as summation of input data multiplied with filter coefficients. The input data and filter coefficients are generally floating point numbers. Let us assume X[n] is the input data vector and H[n] is the filter coefficients vector, the output vector of digital FIR filter is Y[n],

The 'm^{th}' output of a Digital FIR is computed as

$$Y[m] = X[n].H[0] + X[n-1].H[1] + X[n-2].H[2] + \ldots\ldots X[n-m] H[m] \quad (3)$$

The number of taps of the filter is given by 'm'. So for each value of m the input data is multiplied by a filter coefficient and the result is accumulated with the previous sum as shown in Eq.3. The main idea is to multiply and add. This multiplication and Addition is achieved using the Floating Point Arithmetic Unit. The operation of multiply and add is not only restricted to FIR filter, this is used in many signal processing concepts like convolution , IIR filter, Fourier transforms etc.

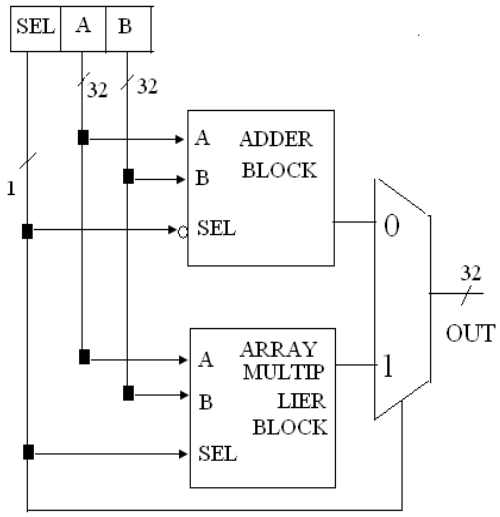

Fig. 11 Modified Floating point Arithmetic Unit.

The adder block and multiplier block of a FPAU makes use of modified Normalization Unit and modified multiplier as shown in the fig.11 The SEL bit of the instruction tells about the operation type. The adder block is designed using ripple carry adder and modified Normalization Unit. An array multiplier is used in the multiplier block. The FPAU operates on floating point numbers. The conventional and proposed FIR filter are simulated and synthesized on XILINX SPARTAN-3 FPGA board. The results of average speed and power dissipation of conventional and proposed FIR filter are graphically represented in Fig.12 and table III.

TABLE III
COMPARISON OF CONVENTIONAL FIR FILTER AND MODIIFIED FIR FILTER.

Parameter	Normal FIR	Modified FIR	Percentage Enhancement
Speed (ns)	105.5	80.5	23.1 %
Dynamic Power(mw)	7.17	5.76	20.5 %

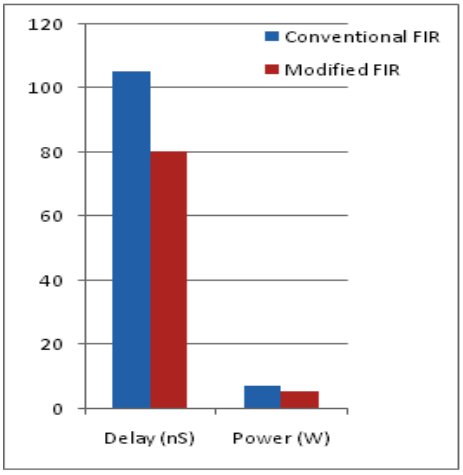

Fig.12 Delay and Power comparison of conventional and Proposed FIR filters.

An FIR filter is realized using the floating point Arithmetic Unit and is implemented on Xilinx FPGA Spartan board. Average speed and power consumption are compared to conventional techniques. This proposed FIR filter has resulted in an average speed enhancement and a power reduction.

VI. Conclusion

In this paper we have designed a digital Finite Impulse Response (FIR) filter using a floating point arithmetic unit (FPAU). The addition unit of the FPAU consists of a ripple carry adder and a modified normalization unit which uses a partitioning technique, enhances the speed of arithmetic unit. The multiplication is performed using a modified array multiplier where the hardware required is reduced, resulting in low power consumption. The FIR filter design is implemented on a FPGA, that resulted in an average speed enhancement of 23.1% and an average power reduction of 20.5% compared with conventional FIR filter.

VII. References

[1] Nabeel Shirazi, Al Walters, Peter Athanas."Quantitative Analysis of Floating Point Arithetic on FPGA based Custom Computing Machines". *IEEE Symposium on FPGAs for Custom Computing Machines*. Apr.1995, pp.155-162.

[2] Gokhale, Maya, Jonathan Cohen, Yoo, W. Marcus Miller, Arpith Jacob, Craig Ulmer, and Roger Pearce, "Hardware Technologies for High-Performance Data-Intensive Computing." *Computer, vol. 41*, no. 4, pp. 60–68, Apr.2008.

[3] P.Lapsley, J.Bier, A.Shoham, and E.Lee, "DSP Processor Fundamentals Architectures and Features". *Piscataway, NJ: IEEE Press*, 1997.

[4] G.K.Yeap, "Practical Low Power VLSI Design", Kluwer Academic Publishers, 1998.

[5] M. Pedram, "Power minimization in IC Design: Principles and applications," *ACM Transactions on Design Automation of Electronic Systems*, vol.1, pp. 3-56, Jan. 1996.

[6] Shikha jain , Prof. Ravi Koneti and Rita Jain, "Analysis of Fast FIR Algorithms based Area Efficient FIR Digital Filters", *International Journal of Computer Science and Information Technologies*, Vol.5,2014, pp . 4124-4127.

[7] V. Anand Kumar and S. Saranya " Less Overhead High Performance Adder Tree for FIR filter architecture in Speech Processing Applications.", *International Journal of Computer Science and Mobile Computing*, Vol.3 Issue.11, November- 2014, pp. 344-350.

[8]. T. Sandhya Pridhini, Diana Aloshius, Aarthi Avanthiga and Rubesh Anand , "Design of Multiple Constant Multiplication algorithm for FIR filter ", *International Journal of Computer Science and Mobile Computing*, Vol.3 Issue.3, March- 2014, pp. 438-444.

[9] R. De Mori, "Suggestion for an IC Fast Parallel Multiplier," *Electronics Letters*, vol. 5, pp.50-51, Feb. 1969.

[10] Haibing Hu, Tianjun Jin, Xianmiao Zhang; Zhengyu Lu, Zhaoming Qian, "A Floating point Coprocessor Configured by a FPGA in a Digital Platform Based on Fixed-point DSP for power Electronics" *IEEE 5th conference on Power Electronics and Motion Control Conference*, 2006.

[11] T.K. Callaway and Jr. Swartzlander, "Power delay characteristics of CMOS multipliers,'' in *Proc.13th Int. Symp. Computer Arithmetic*, pp.26-32, 1997.

[12] Giorgos Dimitrakopoulos, Kostas Galanopoulos, Christos Mavrokefalidis and Dimitris Nikolos," Low-Power Leading-Zero Counting and Anticipation Logic for High-Speed Floating Point Units", *In IEEE Transactions on VLSI Systems*, vol.16, no. 7, pp. 837-850, July 2008.

[13] V.Oklobdzija, 'Comments on leading-zero anticipatory logic for high-speed floating point addition," *IEEE J. Solid State Circuits*, pp. 292-293, Feb. 1997.

[14] N. Quach and M. Flynn, "Leading one prediction implementation, generation, and application." *Technical Report CSL-TR-91-463 Stanford University*, March 1991.

[15] Zhijun Huang and M.D.Ercegovac, "High-performance left-to-right array multiplier design," in *Proceedings of the 16Ih IEEE Symposium on Computer Arithmetic*, pp. 4-11. 2003.

[16]. Sneha Manohar Ramteke,Yogeshwar Khandagre, Alok Dubey "Implementation of Low Power Booth's Multiplier by Utilizing Ripple Carry Adder " , *International Journal of Scientific & Engineering Research*, Volume 5, Issue 5, May-2014 , pp. 145-150.

[17]. J.Eiler, A.Ehliar, D.Liu, "Using Low Precision Floating Point Numbers to Reduce Memory Cost for MP3 Decoding," Proc. IEEE *International Workshop on Multimedia Signal Processing*, Siena, Italy, Sep. 2004.

[18]. J. Fridman, Z. Greenfield, "The Tiger SHARC DSP Architecture," *Proc. IEEE* , pp. 66-76, Jan.-Feb. 2000.

[19] J. Hoffman, G. Lacaze, and P. Csillag, "Iterative Logical Network for Parallel Multiplication," *Electronics Letters*, vol. 4, pp. 178, 1968.

[20] M.S.Elrabaa, I.S. Abu-Khater, M.I. Elmasry, "Advanced Low Power Digital Circuits Techniques", Kluwer Academic Publ., 1997.

[21] P. Burton and D.R. Noaks, "High-Speed Iterative Multiplier," *Electronics Letters*, vol. 4, p. 262, 1968.

[22]. IEEE Standard Board, "IEEE Standard for Binary Floating-point Arithmetic," *The Institute for Electrical and Electronics Engineers*, 1985.

[23] M.Jayaprakash, M.Peer Mohamed and Dr.A.Shanmugam "Design and Analysis of Low Power and Area Efficient Multiplier", *International Conference on Electronic Devices Systems and Applications*, 2011.

SRIKANTH IMMAREDDY Received the Bachelor degree in Electronics and Communication Engineering (ECE) from the JNTU Hyderabad India and Master degree in Digital Systems from Osmania University, India in 2009. He is working as Assistant Professor in ECE Dept at Methodist College of Engineering and Technology, India His main interests are in the fields of Very Large Scale Integration and Digital Signal Processing.

SRAVANKUMAR TALUSANI Received the Bachelor degree in Electronics and Communication Engineering (ECE) from Osmania University, India and Master degree in Digital Systems & Computer Electronics from JNTU Ananthapur, India. He is working as Assistant Professor in ECE Dept at Methodist College of Engineering and Technology, India His major interests are in the fields of Analog Electronics and Signal Processing.

RAYAVARAPU PRASADRAO Received the Bachelor degree in Electronics and Communication Engineering (ECE) from Andhra University, India and Master degree from JNTU, Hyderabad. Pursuing Ph.D from Gitam University, India. He is working as Associate Professor in ECE Dept at Avanthi Institute of Engineering and Technology, India. His major interests are in the fields of Digital Electronics.