# Design of Vedic Multiplier for Digital Signal Processing Applications

R.Naresh Naik[1], P.Siva Nagendra Reddy[2], K. Madan Mohan[3]

[1]*P.G. Scholar (M. Tech), Dept. of ECE, Intell Engineering College, Anantapur*

[2]*P.G. Scholar (M. Tech), Dept. of ECE, Intell Engineering College, Anantapur*

[3]*Asst.Professor, Dept. of ECE, Intell Engineering College, Anantapur*

***Abstract*— Multiplier is one of the most important part in any processor speed which improves the speed of the operation like in special application processors like Digital Signal Processor (DSPs).To Increase the speed of operation we should take care of the precision previously we used the floating point multipliers which were consume more silicon area and take more clock frequency than fixed point (Q-format) multipliers. Now we propose a method which is faster multiplication technique by using Vedic mathematics formula Urdhava Tiryakbhyam method which means vertically and cross wire. All the operations in Vedic multiplier were executed concurrently and also we will get the output same as input bit length so Vedic multiplier is time, space and power efficient .The coding is done for 16-bit (Q-15) and 32-bit (Q-31) fractional fixed point multiplications using Verilog and synthesized using Xilinx ISE version 14.3. Further the speed comparisons of this multiplier with Normal Booth multiplier were presented. The results clearly show that our Urdhava Tiryakbhyam multiplier can have great amount of impact on the DSP applications to improve the execution speed of the DSP processors when compared to other multipliers.**

**Keywords- Q-format, Urdhava Tiryakbhyam, Vedic Mathematics, Fractional fixed point.**

## I.INTRODUCTION

The striking feature of Vedic Mathematics is the easiest and fastest way to perform any mathematical calculation mentally. Vedic Mathematics was discovered between 1911 to 1918 by Sri Bharti Krishna Tirthaji (1884-1960).He organized and classified the whole of Vedic Mathematics into 16 formulae or also called as sutras. These formulae form the backbone of Vedic mathematics.

Great amount of research has been done all these years to implement algorithms of Vedic mathematics on digital processors. It has been observed that due to coherence and symmetry in these algorithms it can have a regular silicon layout and consume less area [2,3] along with lower power consumption. Normally signal processing algorithms are developed using high level languages like C or Mat lab using floating point number representations. The algorithm to architecture mapping using floating point number representation consumes more hardware which tends to be expensive.

Fixed point number representation is a good option to implement at silicon level.Hence our focus in this work is to develop optimized hardware modules for multiplication operation which is one of the most frequently used operation in signal processing applications like Fourier transforms, FIR and IIR filters, image processing systems, seismic signal processing, optical signal processing etc. Any attempt to come out with an optimized architecture for this basic block is advantageous during the product development stages. Considering fixed point representation, 16 bit Q15 format and 32 bit Q31 format provide required precision for most of the digital signal processing applications and it is best suited for implementation on processors. The advantage it provides over floating point multipliers is in the fact that Q format fraction multiplications can be carried out using integer multipliers which are faster and consume less die area. DSP Processors like TMS320 series from Texas Instruments work on 16 bit Q15 format. In this paper we propose the implementation of fixed point Q-format [6] high speed multiplier using Urdhava Tiryakbhyam method of Vedic mathematics. Further we have also implemented multipliers using normal booth algorithm and presented a comparative study on maximum frequency or speed of these multipliers.

## II. FIXED POINT ARITHMETIC

An N-bit fixed point number can be interpreted as either an integer or a fractional number. Integer fixed point is difficult to use in processors due to possible overflow. For e.g. In a 16-bit processor for signed integers the dynamic range is from $-2^{15}$ to $2^{15}-1$ i.e. 32768 to 32767. If 500 is multiplied by 800 the result is 40000 which is an overflow. In order to overcome this situation fractional fixed point representation also known as Q-format is used.

### *A. Q-format Representation*

**Q** is a fixed point number format where the number of fractional bits (and optionally the number of integer bits) is specified. For example, a Q15 number has 15 fractional bits; a Q1.14 number has 1 integer bit and 14 fractional bits. Q format is often used in hardware that does not have a floating-point unit and in applications that require constant resolution.

Q format numbers are (*notionally*) fixed point numbers (but not actually a number itself); that is, they are stored and operated upon as regular binary numbers (i.e. signed integers), thus allowing standard integer hardware/ALU to perform rational number calculations. The number of integer bits, fractional bits and the underlying word size are to be chosen by the programmer on an application-specific basis — the programmer's choices of the foregoing will depend on the range and resolution needed for the numbers. The machine itself remains oblivious to the notional fixed point representation being employed — it merely performs integer arithmetic the way it knows how. Ensuring that the computational results are valid in the Q format representation is the responsibility of the programmer.

The Q notation is written as Q*m.n*, where:

1. Q designates that the number is in the Q format notation — the Texas Instruments representation for signed fixed-point numbers (the "Q" being reminiscent of the standard symbol for the set of rational numbers).
2. *m* is the number of bits set aside to designate the two's complement integer portion of the number, exclusive of the sign bit (therefore if m is not specified it is taken as zero).
3. *n* is the number of bits used to designate the fractional portion of the number, i.e. the number of bits to the right of the binary point. (If n = 0, the Q numbers are integers — the degenerate case).

Note that the most significant bit is always designated as the sign bit in order to allow standard arithmetic-logic hardware to manipulate Q numbers. Representing a signed fixed-point data type in Q format therefore always requires *m*+*n*+1 bits to account for the sign bit. Hence the smallest machine word size required to accommodate a Q*m.n* number is *m*+*n*+1, with the Q number left justified in the machine word.

For a given Q*m.n* format, using an *m*+*n*+1 bit signed integer container with *n* fractional bits:

- its range is $[-2^m, 2^m - 2^{-n}]$
- its resolution is $2^{-n}$

For example, a Q14.1 format number:

- Requires 14+1+1 = 16 bits
- Its Range is $[-2^{14}, 2^{14} - 2^{-1}]$ = [-16384.0, +16383.5] = [0x8000, 0x8001 … 0xFFFF, 0x0000, 0x0001 … 0x7FFE, 0x7FFF]
- Its resolution is $2^{-1}$ = 0.5

Unlike floating point numbers, the resolution of Q numbers will remain constant over the entire range.

B. *Q-format Multiplication*

When two Q15 numbers are multiplied their product is 32 bits long as illustrated in Fig. 1. The product has a redundant or extended sign bit. Since the product stored in memory should also be a Q15 number we left shift the product by one bit and the most significant 16 bits (including sign bit) is stored in the memory.

Fig. 1 demonstrates multiplication of two Q15 format numbers. The process remains same for Q31 format wherein after left shifting the product by one bit, the most significant 32 bits are stored in the memory. Therefore with Q-format, multiplications of two fractional numbers can be carried out by using integer multiplications. Integer multiplications consume less area and are faster compared to floating point multipliers which is the major advantage of Q- format representation.

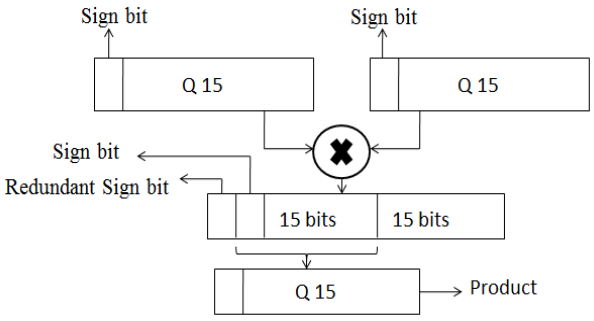


Fig. 1. Multiplication of two Q15 format numbers yielding the product in Q15 formats itself.

III. URDHAVA TIRYAKBHYAM METHOD

Urdhava Tiryakbhyam [2] is based on a novel concept through which all partial products are generated concurrently. Fig.2 demonstrates a 4x4 binary multiplication using this method. The method can be generalized for any N x N bit multiplication. This is independent of the clock frequency of the processor because the partial products and their sums are calculated in parallel. The net advantage is that it reduces the need of microprocessors to operate at increasingly higher clock frequencies. As the operating frequency of a processor increases the number of switching instances also increases. This results in more power consumption and also dissipation in the form of heat which results in higher device operating temperatures. Another advantage is its scalability. The processing power can easily be increased by increasing the input and output data bus widths since it has a regular structure [3]. Due to its regular structure, it can be easily layout in a silicon chip and also consumes optimum area [2]. As the number of input bits increase, gate delay and area increase very slowly as compared to other multipliers. Therefore Urdhava Tiryakbhyam multiplier is time, space and power efficient.

The line diagram in fig. 2 illustrates the algorithm for multiplying two 4-bit binary numbers $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$. The procedure is divided into 7 steps and each step generates partial products. Initially as shown in step 1 of fig. 2, the least

significant bit (LSB) of the multiplier is multiplied with least significant bit of the multiplicand (vertical multiplication)

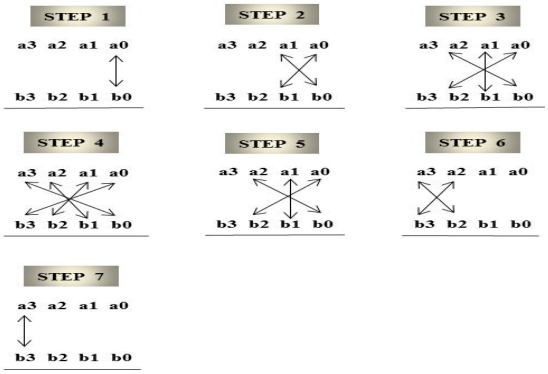


Fig. 2 Multiplication of two 4 bit numbers using Urdhava Tiryakbhyam method.[7]

This result forms the LSB of the product. In step 2 next higher bit of the multiplier is multiplied with the LSB of the multiplicand and the LSB of the multiplier is multiplied with the next higher bit of the multiplicand (crosswire multiplication). These two partial products are added and the LSB of the sum is the next higher bit of the final product and the remaining bits are carried to the next step. For example, if in some intermediate step, we get the result as 1101, then 1 will act as the result bit(referred as rn) and 110 as the carry (referred as cn). Therefore cn may be a multi-bit number. Similarly other steps are carried out as indicated by the line diagram. The important feature is that all the partial products and their sums for every step can be calculated in parallel.

Thus every step in fig. 2 has a corresponding expression as follows:

r0=a0b0……..(1)

c1r1=a1b0+a0b1……...(2)

c2r2=c1+a2b0+a1b1 + a0b2…….(3)

c3r3=c2+a3b0+a2b1 + a1b2 + a0b3……(4)

c4r4=c3+a3b1+a2b2 + a1b3…..…..…..... (5)

c5r5=c4+a3b2+a2b3……..(6)

c6r6=c5+a3b3…….(7)

With c6r6r5r4r3r2r1r0 being the final product [5].

Hence this is the general mathematical formula applicable to all cases of multiplication and its hardware architecture is shown in fig. 3.In order to multiply two 8-bit numbers using 4-bit multiplier we proceed as follows. Consider two 8 bit numbers denoted as AHAL and BHBL where AH and BH corresponds to the most significant 4 bits, AL and BL are the least significant 4 bits of an 8-bit number. When the numbers are multiplied according to Urdhava Tiryakbhyam (vertically and crosswire) method, we get,

AH AL

BH BL

$(AH \times BH) + (AH \times \overline{BL + BH} \times AL) + (AL \times BL)$.

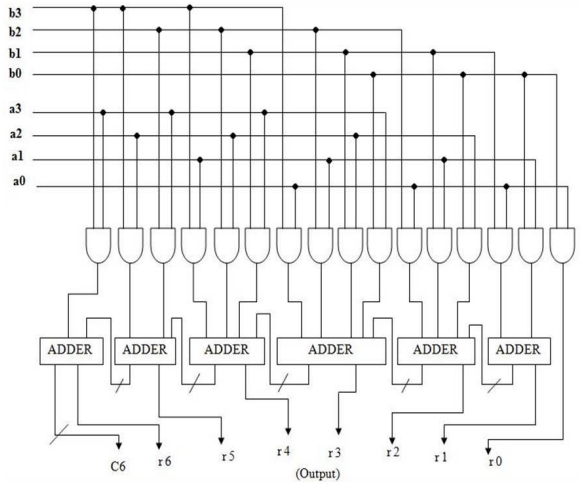


Fig. 3.Hardware architecture of 4 X 4 Urdhva Tiryakbhyam multiplier.

Thus we need four 4-bit multipliers and two adders to add the partial products and 4-bit intermediate carry generated. Since product of a 4 x 4 multiplier is 8 bits long, in every step the least significant 4 bits correspond to the product and the remaining 4 bits are carried to the next step. This process continues for 3 steps in this case. Similarly, 16 bit multiplier has four 8 x 8 multiplier and two 16 bit adders with 8 bit carry. Therefore we see that the multiplier is highly modular in nature. Hence it leads to regularity and scalability of the multiplier layout.

## IV. ARCHITECTURE

Urdhava Tiryakbhyam integer multiplier [4] is faster since all the partial products are computed concurrently. Considering a 16 bit Q15 multiplier, the product is also a Q15 number which is 16 bits long. Firstly, if the MSB of input is 1 then it is a negative number. Therefore 2's complement of the number is taken before proceeding with multiplication. Since the MSB denotes sign it is excluded and a '0' is placed in this position while multiplying. A Q15 format multiplier consists of four 8 x 8 Urdhava multipliers and the resulting product is 32

bits long as shown in fig. 4. But the product of a Q15 number is also a Q15 number which should be 16 bits long. Therefore the 32 bit product is left shifted by 1 bit to remove the redundant sign bit and only the most significant 16 bits of this product are considered which constitute the final product. An xor operation is performed on the input sign bits to determine the sign of the result.

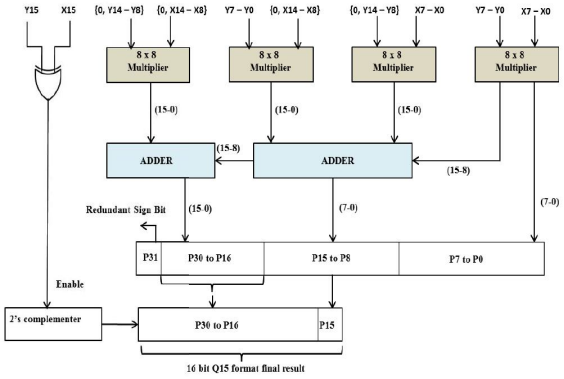


Fig. 4.Architecture of a Q15 format multiplier. Multiplication of two Q15 numbers X and Y results in a Q15 product denoted by P in the figure

If the output is '1' it enables the conversion of the 16 bit final result to its 2's complement format indicating a negative product. Similarly, for a 32 bit Q31 format multiplier as shown in fig. 5, four 16 X 16 Urdhava multipliers are used and only the most significant 32 bits after left shifting by one bit are considered which constitute the final 32 bit Q31 format product. An xor operation similar to Q15 multiplier is used to change the result to 2's complement format if it is negative.

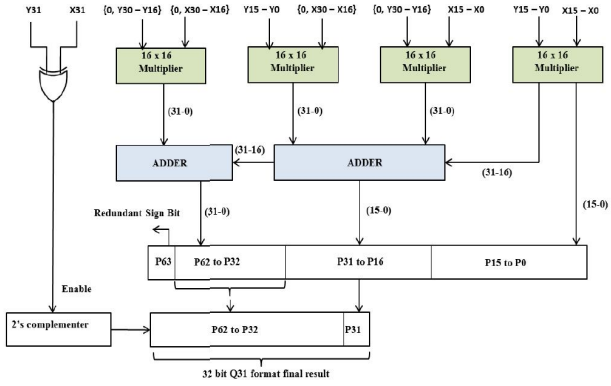


Fig. 5 Architecture of a Q31format multiplier. Multiplication of two Q31 numbers X and Y results in a Q31 product denoted by P in the figure

## V. IMPLEMENTATION AND RESULTS

The proposed Urdhava Tiryakbhyam Q-format multiplier is designed using Verilog hard ware description language and structural form of coding. The basic block of both Q15 and Q31 multiplier is a 4 x 4 Urdhava Tiryakbhyam integer multiplier which in turn is made up of two 2 x 2 multiplier blocks. The design is completely synchronized by the clock. Further, the Q-format multipliers were also implemented using normal booth's algorithm.

The code is completely synthesized using Xilinx XST and implemented on device family Virtex-5, device XC5VL50, package FF324 with speed grade -2.

*Simulation Results:*

The design was simulated using Isim on Xilinx ISE 14.3 version. For Q15 format multiplication as shown in fig. 6,

Input1 = -0.45 = 0011 1001 1001 1001

Input2 = - 0.65 = 0101 0011 0011 0011

Output = 0.2925 = 0010 0101 0111 0001

For Q31 format multiplication as shown in fig. 7.

Input1=0.333333 = 00101010101010101010011111011111

Input2= -0.666666 = 10101010101010101011000001000010

Output= -0.22222177777439355850219726556 25= 1111 1000 1110 0011 0001 1011 1000 0101

But the actual value of the product is -0.222221777778. Therefore precision loss is involved in this multiplication and is found to be 3.60644E-12 which is less than the resolution of Q31 representation i.e $2^{-31}$. Thus it provides 32 bit accurate product which is acceptable for most of the DSP applications.

As shown in table 1, the comparison report suggests that a Q31 format Urdhava Qformat multiplier is faster by 2.61 times than Normal Booth Multiplier . For a Q-15 format multiplier, as seen in table 2 the speed factor improvement is 1.84 times compared to booth multiplier.When Virtex-5 DSP48E slices were used with Normal Booth multiplier, Urdhava multiplier still proved to be faster indicating that it is the best choice for implementing faster multipliers on FPGA.

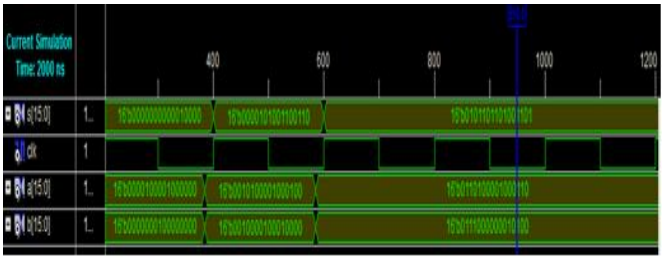

Fig.6.Q15multiplication

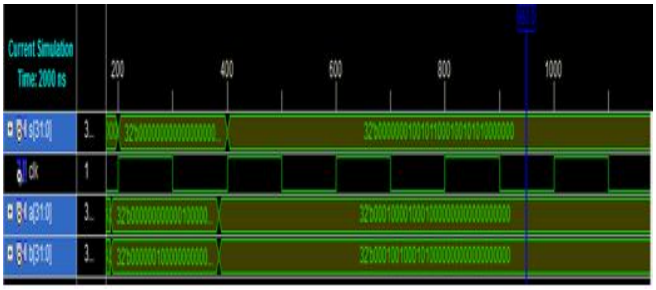

Fig.7.Q31multiplication

Table-1Comparison of 32 bit q31-format multipliers

| | **Maximum Frequency (in MHz)** | **4-input Slice LUT Usage** | **Factor by which Urdhava Multiplier is faster** |
|---|---|---|---|
| **Urdhava Multiplier** | 233.3 | 3055/17344 | ------- |
| **Normal Booth Multiplier** | 100 | 2206/17344 | 2.33 times |

Table-2Comparison of 16 bit q15-format multipliers

| | **Maximum Frequency (in MHz)** | **4-input Slice LUT Usage** | **Factor by which Urdhava Multiplier isFaster** |
|---|---|---|---|
| **Urdhava Multiplier** | 165.3 | 355/17344 | ------ |
| **Normal Booth Multiplier** | 100.12 | 285/17344 | 1.65 times |

VI. RTL SCHEMATIC

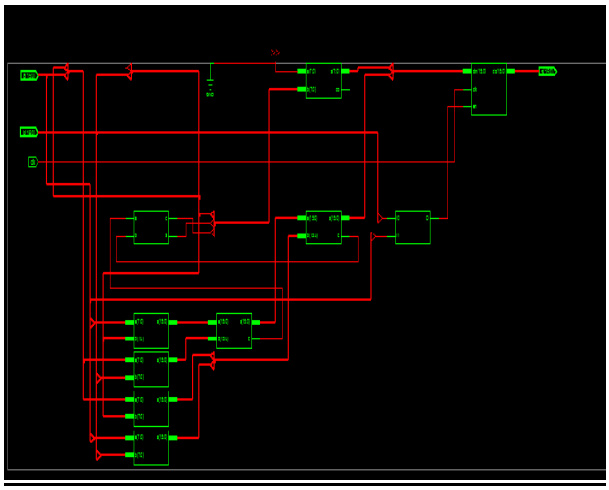

Fig. 8 RTL Schematic of 16 BIT VEDIC Multiplier

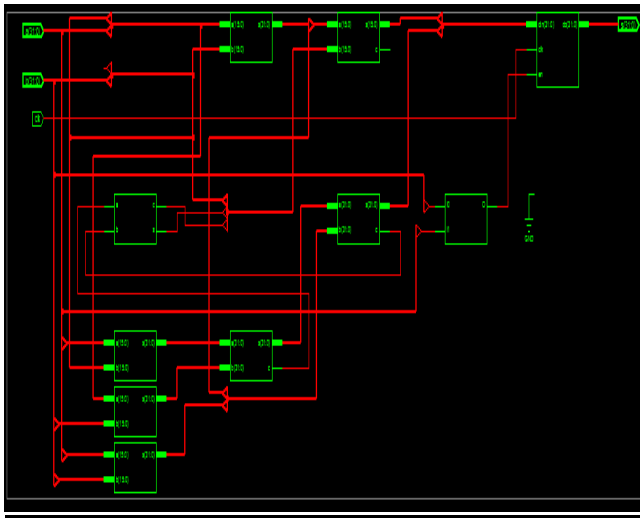

Fig.9 RTL Schematic of 32 BIT VEDIC Multiplier

## VII.CONCLUSION

This method is a fast multiplier architecture for signed Q-format multiplications using Urdhava Tiryakbhyam method of Vedic mathematics. Since Q-format representation is widely used in Digital Signal Processors the proposed multiplier can substantially speed up the multiplication operation which is the basic hardware block. They occupy less area and are faster than the booth multipliers. Therefore the Urdhava Tiryakbhyam Q-format multiplier is best suited for signal processing applications requiring faster multiplications. Future work lies in the direction of introducing pipeline stages in the multiplier architecture for maximizing throughput.

## VIII.ACKNOWLEDGMENT

We place our gratitude on record to the Department of Electronics and Communication Engineering, Intell Engineering College for the support rendered to us in carrying out this work.

## REFERENCES

[1] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, "Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda," Motilal Banarasidas Publishers, Delhi, 2009, pp. 5-45.

[2] Sandesh S. Saokar. R. M. Banakar. Saroja Siddamal., "*High Speed Signed Multiplier for Digital Signal. Processing Applications*,"ieee 2012.

[3] Himanshu Thapliyal and M. B. Srinivas,"An efficient method of elliptic curve encryption using Ancient Indian Vedic Mathematics,"48th IEEE International Midwest Symposium on Circuits and Systems, 2005, vol. 1, pp. 826-828.

[4] M. Pradhan and R. Panda, "Design and Implementation of Vedic Multiplier," A.M.S.E Journal, Computer Science and Statistics, France vol. 15, July 2010, pp. 1-19.

[5] Harpreet Singh Dhillon , Abhijit Mitra, "A Reduced-Bit Multiplication Algorithm for Digital Arithmetics," International Journal of Computational and Mathematical Sciences, Spring 2008, pp.64-69.

[6] Sen-Maw Kuo and Woon-Seng Gan, "Digital Signal Processor, architectures ,implementations and applications," Pearson Prentice Hall, 2005, pp. 253-323.

[7] S. S. Kerur, Prakash Narchi, Jayashree C .N., Harish M.Kittur and GirishV.A."Implementation of Vedic Multiplier for Digital Signal Processing," International Journal of Computer Applications, 2011, vol. 16, pp. 1-5.

## AUTHORS

**Author 1–R. Naresh Naik,** received his Bachelor of Technology in Electronics and Communication Engineering in N.B.K.R. Institute of Science&Technology,Vakadu, A.P., INDIA in 2010. He is doing his research on Advanced Multiplication Techniques using Multipliers for DSP Applications under the guidance of Mr. K. Madan Mohan. His areas of interests are doing VERILOG based projects.

**Author 2–P.Siva Nagendra Reddy,** received his Bachelor of Technology in Electronics and Communication Engineering in C.M.R. Institute of Technology, Kandlakoya, Hyderabad, A.P., INDIA in 2011. He is doing his research on Advanced Multiplication Techniques using Multipliers for DSP Applications under. His areas of interests are doing VHDL and VERILOG based projects.

**Author 3- Mr. K. Madan Mohan**, M.Tech, Asst.Professor in the Department of Electronics and Communication Engineering, Intell Engineering College. Areas of interests are VLSI System Design, wireless sensors, Digital systems and embedded systems.