# High Speed Multiplier Using Vedic Mathematics

**Aishwarya Nawandhar[1], IshwariPamu[2], PranaliKanampalliwar[3], Prof. V. N. Jirafe [4]**
[1, 2, 3, 4] Dept of Electronics and Telecommunication
[1, 2, 3, 4] PES's Modern College of Engineering, Pune, Maharashtra, India

***Abstract***- *Multipliers have a lot of applications like in DSP, FIR filters, Image Processing, Microprocessors and communication. Multipliers for higher order multiplications are not very efficient in terms of speed. Also, a huge number of compressors or adders are required to perform the partial product generation and addition. Due to the increase in the need of high-speed processors, the need of high speed multipliers is also increasing. High speed and high performance architecture of a multiplier implemented on the Field Programmable Gate Array (FPGAs) is proposed in this paper. The proposed paper is based on ancient Indian Vedic mathematics which provides us with a fast multiplication method. In this, the whole system of mathematics is covered in terms of 16 sutras (word formulae). Among all of these various methods of multiplication in Vedic mathematics, UrdhavaTiryakbhyam is discussed in detail. It is a general multiplication formula and can be applied to all the cases of multiplication. The most significant aspect of UrdhavaTiryakbhyam sutra is its vertical and crosswise structure. It generates all the intermediate products parallel and eliminates unwanted steps of multiplication with zeros. The proposed multiplier is very efficient, gives minimum delay for multiplication of small as well as large numbers.This particular Vedic multiplier has various advantages over the previously proposed Vedic multipliers. One of the advantage is that it is easy to design and easier to implement as it does not use complex circuits. The code is easy to understand and modify according to the requirement. Unlike other proposed multipliers, the aim is solely to increase the speed and make it better compared to conventional multiplier. The paper, therefore, thoroughly lists out advantages of Vedic Multiplier in terms of speed. In the proposed multiplier based on UrdhavaTiryakbhyam, the two 8-bit numbers to be multiplied are grouped into 4-bit numbers so that the multiplier decomposes into 4×4 multiplication modules, which in turn are converted to 2x2 multiplication modules. The coding is done in VHDL (very high speed integrated circuit hardware description language) and synthesis is done using Xilinx ISE series. A FPGA is used to display the results for the same.*

***Keywords***- Vedic Mathematics, UrdhavaTiryakbhyam, VHDL, FPGA.

## I. INTRODUCTION

Out of all the arithmetic operations, multiplication is considered to be the most fundamental function. The search for new algorithms and the ever increasing need for faster processing speed is perpetually driving major developments in processor technologies. One of the key hardware blocks in most Digital Signal Processing Systems is the multiplier. Multiply and Accumulate (MAC) and inner product are multiplication based operations which are among some of the most frequently and commonly used Computations Intensive Arithmetic Functions (CIAF), at present, implemented in many Digital Signal Processing (DSP) applications such as Fast Fourier Transform (FFT), filtering, convolution and in microprocessors in its arithmetic and logic unit [8]. Since multiplication dominates the execution time of most algorithms, there is a never ending need of high speed multipliers. Reducing the time delay and power consumption are very essential requirements in many applications [2, 9]. Many engineers, with the recent developments in technology, have tried to design multipliers which offer either high speed, low power consumption, area efficiency or an even combination of them.

Array multiplication algorithm and Booth multiplication algorithm are two most common multiplication algorithms used in digital hardware. Due to the partial products independently calculated in parallel, the computation time taken by the array multiplier is comparatively less. The delay associated with array multiplier is the time taken by the signals to propagate through the gates that form the multiplication array. Another important multiplication algorithm is Booth's multiplication algorithm. They are used for high speed multiplication and exponential operations. Large partial sum and partial carry registers are required to carry out these operations. The approach taken in Vedic Mathematics is totally unique in itself and is considered to be very similar to the way a human mind functions. In this project, multiplication operations and its implementation are done using Vedic Mathematics method in VHDL language [1]. Vedic Mathematics is an ancient Indian system of mathematics or a unique method to make calculations based of simple rules and principles with the help of which any mathematical problem can be solved – be it algebra, geometry, trigonometry or an arithmetic problem. It existed in ancient

India and was reintroduced to the world by a popular mathematician, Sir Bharati Krishna Tirthaji. It deals with several basic as well as complex mathematical operations. Methods used for basic arithmetic calculations are extremely simple and powerful [2, 9]. Vedic Mathematics is based on 16 sutras which are word formulae describing the methods of solving a whole range mathematical problems. The simplicity of the sutras of Vedic Mathematics lays the foundation for its application in several major-league domains of engineering such as Signal Processing, VLSI and Control Engineering.

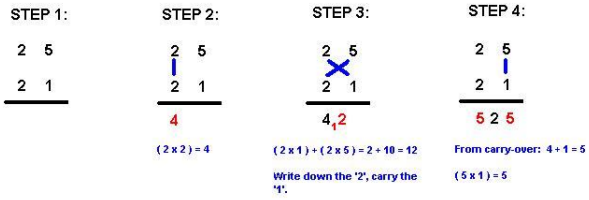
The partial products required for multiplication are calculated well in advance, much before the actual multiplication operation begins. Addition of these partial products takes place based on the Vedic Mathematics algorithm to obtain the final product. This in turn results in a very high speed approach to perform multiplication operations.

The Vedic multiplier is based on the Vedic Mathematics formulae i.e. sutras, traditionally used in the decimal number system for the multiplication of two or more digits. In this project, the same ideas and methods are applied to the binary number system to make the proposed algorithm compatible with the digital hardware. The sutra used in the proposed algorithm is called UrdhavaTiryakbhyam.

**UrdhavaTiryakbhyam sutra:**

The proposed multiplier is based on UrdhavaTiryakbhyam algorithm which is one of the sutras of ancient Indian Vedic Mathematics. It is applicable to all cases of multiplication. Originating from two Sanskrit words, the term 'UrdhavaTiryakbhyam' literally means 'vertically' and 'crosswise' respectively. In this sutra, generation of all products is done with the concurrent addition of all the partial products which are generated way before the actual multiplication process starts. The partial products and their sums are calculated in parallel hence making the proposed multiplier independent of the processor's clock frequency. By increasing the widths of the input and output data buses, the processing power can be easily increased. The gate delay and area consumption of the multiplier increases very slowly on the increase in the number of bits which proves to be an advantage. Therefore, the proposed multiplier is time, power and area efficient.

Multiplication of two decimal number:




## II. DESIGN AND IMPLEMENTATION

To design N-bit Multiplier based on UrdhavaTiryakbhyam Sutra of Vedic Mathematics, we use N/2-bit Multiplier based on the same technique. In this paper, one can see how a Vedic Multiplier can be build. The proposed paper has the design and simulation results of 8-bit Vedic multiplier. The similar concept can be extended to design higher order multipliers.

**2x2 Vedic Multiplier:**

First, the LSBs were multiplied to produce the LSB of the final product (vertical). Next, the least significant bit is multiplied with the next nearer bit of the multiplier and then sum with the product of least significant bit of multiplier and next nearer bit of the multiplicand (crosswise). The resulted sum gives the second bit of the final product. The resulted carry is added with the partial product which is attained by multiplying the MSB to give the sum and carry.In the final product, the carry be fourth bit and the sum is the third bit. This 2X2 Vedic multiplier module can be implemented using four input AND gates & two half-adders as shown in the figure below.

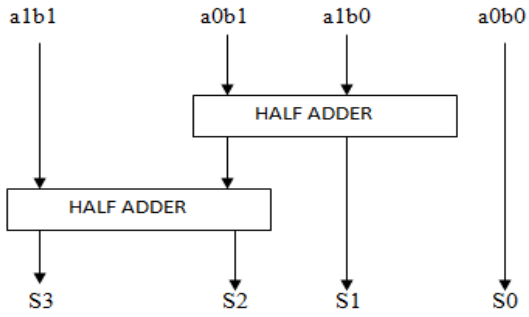



**Fig. (b): 2x2 Vedic multiplier**

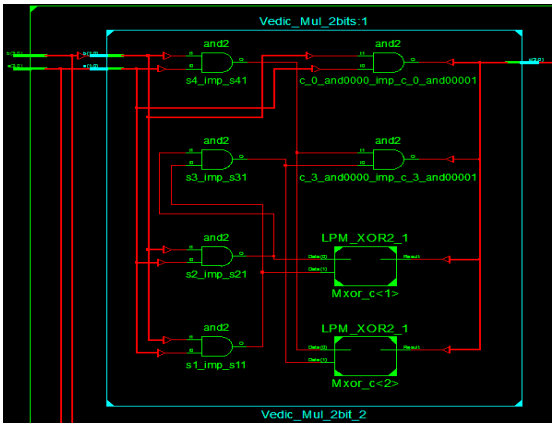



**Fig. (c): Schematic of 2x2 Vedic Multiplier**

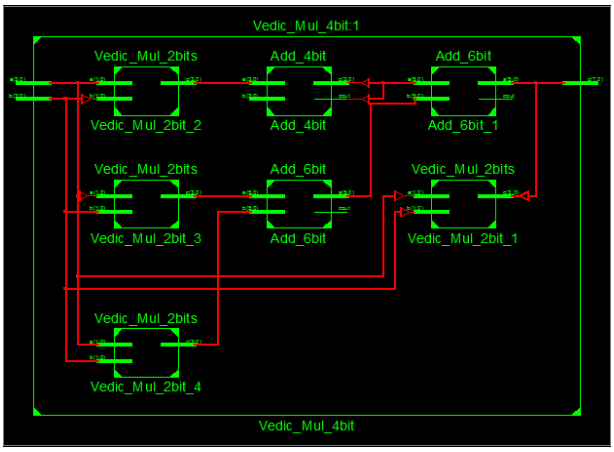



**Fig. (e): Schematic of 4x4 UrdhavaTiryakbhyam Multiplier**

**4x4 Vedic Multiplier:**

Divide the no. of bits in the inputs equally in two parts.First 2x2 multiplier inputs are “A1 A0” and “B1 B0”. The last block is 2x2 bit multiplier with inputs “A3 A2” and “B3 B2”. The middle one shows two, 2x2 bit multiplier with inputs “A3A2” & “B1B0” and “A1A0” & “B3B2”. The first RC Adder is used to add two 4-bit operands obtained from cross multiplication of the two middle 2x2 bit multipliers. The second RC Adder is used to add two 4-bit operands, i.e. concatenated 4-bit (“00” & most significant two output bits of right hand most of 2x2 multiplier) and one 4-bit operand we get as the output sum of first RC Adder. Its carry is forwarded to third RC Adder. Now the third RC Adder is used to add two 4-bit operands, i.e. concatenated 4 - bit (carry, “0” & most significant two output sum bits of 2nd RC Adder) and one 4-bit operand we get as the output sum of left hand most of 2x2 multiplier module.

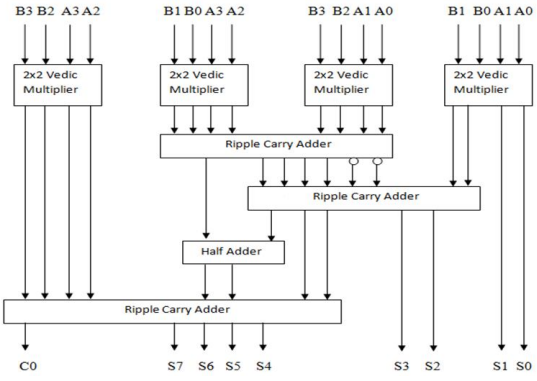



**Fig. (d):4x4 Vedic Multiplier**

**8x8 Vedic Multiplier:**

Divide the no. of bits in the inputs equally in two parts. First 4x4 multiplier inputs are “A3 A2 A1 A0” and “B3 B2 B1 B0”. The last block is 4x4 bit multiplier with inputs “A7 A6 A5 A4” and “B7 B6 B5 B4”. The middle one shows two, 4x4 bit multiplier with inputs “A7 A6 A5 A4” & “B3 B2 B1 B0” and “A3 A2 A1A0” & “B7 B6 B5 B4”.The first RC Adder is used to add two 8-bit operands obtained from cross multiplication of the two middle 4x4 bit multipliers. The second RC Adder is used to add two 8-bit operands, i.e. concatenated 8-bit and one 8-bit operand we get as the output sum of first RC Adder. Its carry is forwarded to third RC Adder. Now the third 8-bit RC Adder is used to add two 8-bit operands, i.e. concatenated 8-bit and one 8-bit operand we get as the output sum of left hand most of 4x4 multiplier module.

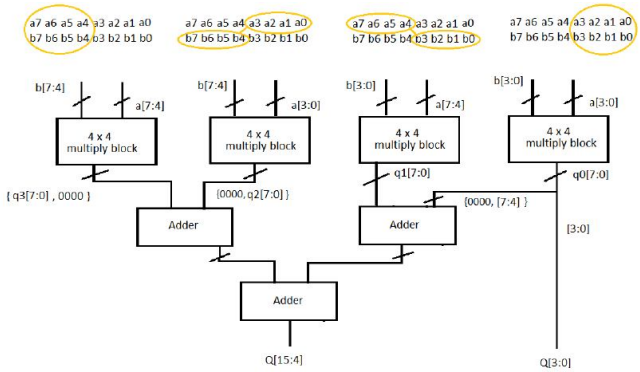



**Fig. (f):8x8 Vedic Multiplier**

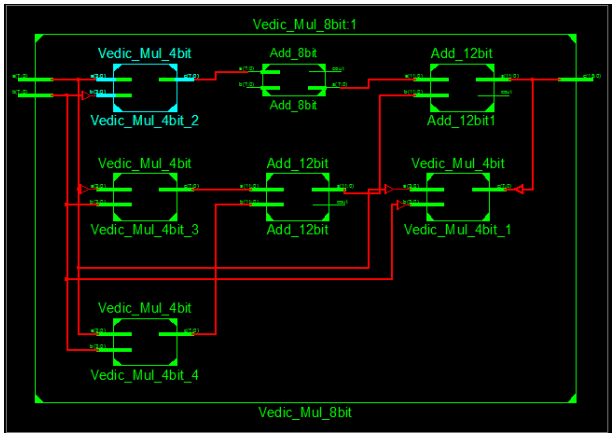
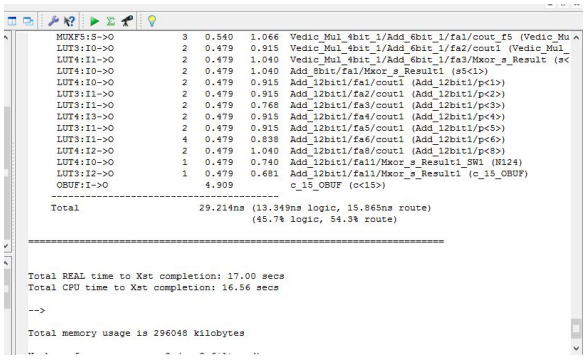


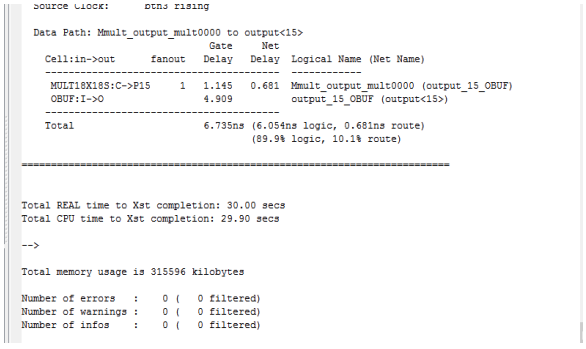

**Fig. (e): Schematic of 8x8 UrdhavaTiryakbhyam Multiplier**

**Comparision of Vedic and array multiplier in terms of time:**



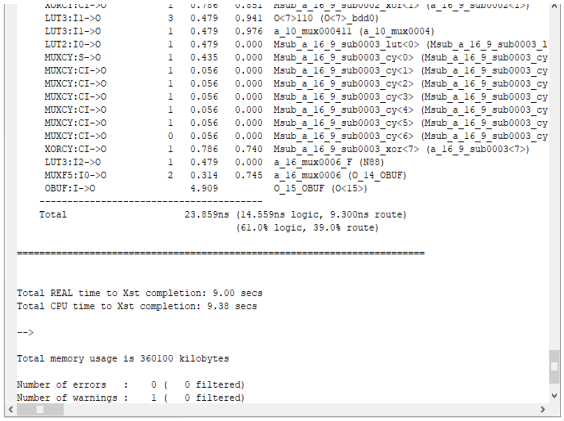
```
    MUXF5:S->O            3   0.540   1.066  Vedic_Mul_4bit_1/Add_6bit_1/fa1/cout_f5 (Vedic_Mu
    LUT3:I0->O            2   0.479   0.915  Vedic_Mul_4bit_1/Add_6bit_1/fa2/cout1 (Vedic_Mul_
    LUT4:I1->O            2   0.479   1.040  Vedic_Mul_4bit_1/Add_6bit_1/fa3/Mxor_s_Result (s<
    LUT4:I0->O            2   0.479   1.040  Add_8bit/fa1/Mxor_s_Result1 (s5<1>)
    LUT4:I0->O            2   0.479   0.915  Add_12bit1/fa1/cout1 (Add_12bit1/pc1>)
    LUT3:I1->O            2   0.479   0.915  Add_12bit1/fa2/cout1 (Add_12bit1/pc2>)
    LUT3:I1->O            2   0.479   0.768  Add_12bit1/fa3/cout1 (Add_12bit1/pc3>)
    LUT4:I3->O            2   0.479   0.915  Add_12bit1/fa4/cout1 (Add_12bit1/pc4>)
    LUT3:I1->O            2   0.479   0.915  Add_12bit1/fa5/cout1 (Add_12bit1/pc5>)
    LUT3:I1->O            4   0.479   0.838  Add_12bit1/fa6/cout1 (Add_12bit1/pc6>)
    LUT4:I2->O            2   0.479   1.040  Add_12bit1/fa8/cout1 (Add_12bit1/pc8>)
    LUT4:I0->O            1   0.479   0.740  Add_12bit1/fa11/Mxor_s_Result1_SW1 (N124)
    LUT3:I2->O            1   0.479   0.681  Add_12bit1/fa11/Mxor_s_Result1 (c_15_OBUF)
    OBUF:I->O             4.909          c_15_OBUF (c<15>)
    ----------------------------------------
    Total                    29.214ns (13.349ns logic, 15.865ns route)
                                      (45.7% logic, 54.3% route)

==========================================================================


Total REAL time to Xst completion: 17.00 secs
Total CPU time to Xst completion: 16.56 secs

-->

Total memory usage is 296048 kilobytes
```

**Fig (f): Timing Report of 8-bit Vedic Multiplier**



```
Source Clock:      btns rising

Data Path: Mmult_output_mult0000 to output<15>
                           Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   MULT18X18S:C->P15     1   1.145   0.681  Mmult_output_mult0000 (output_15_OBUF)
   OBUF:I->O                 4.909          output_15_OBUF (output<15>)
  ----------------------------------------
  Total                      6.735ns (6.054ns logic, 0.681ns route)
                                     (89.9% logic, 10.1% route)

==========================================================================


Total REAL time to Xst completion: 30.00 secs
Total CPU time to Xst completion: 29.90 secs

-->

Total memory usage is 315596 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

**Fig (g): Timing Report of 8-bit Array Multiplier**



```
    LUT3:I1->O            3   0.479   0.941  O<7>110 (O<7>_bdd0)
    LUT3:I1->O            1   0.479   0.976  a_10_mux000411 (a_10_mux0004)
    LUT2:I0->O            1   0.479   0.000  Msub_a_16_9_sub0003_lut<0> (Msub_a_16_9_sub0003_l
    MUXCY:S->O            1   0.435   0.000  Msub_a_16_9_sub0003_cy<0> (Msub_a_16_9_sub0003_cy
    MUXCY:CI->O           1   0.056   0.000  Msub_a_16_9_sub0003_cy<1> (Msub_a_16_9_sub0003_cy
    MUXCY:CI->O           1   0.056   0.000  Msub_a_16_9_sub0003_cy<2> (Msub_a_16_9_sub0003_cy
    MUXCY:CI->O           1   0.056   0.000  Msub_a_16_9_sub0003_cy<3> (Msub_a_16_9_sub0003_cy
    MUXCY:CI->O           1   0.056   0.000  Msub_a_16_9_sub0003_cy<4> (Msub_a_16_9_sub0003_cy
    MUXCY:CI->O           1   0.056   0.000  Msub_a_16_9_sub0003_cy<5> (Msub_a_16_9_sub0003_cy
    MUXCY:CI->O           0   0.056   0.000  Msub_a_16_9_sub0003_cy<6> (Msub_a_16_9_sub0003_cy
    XORCY:CI->O           1   0.786   0.740  Msub_a_16_9_sub0003_xor<7> (a_16_9_sub0003<7>)
    LUT3:I2->O            1   0.479   0.000  a_16_mux0006_F (N88)
    MUXF5:I0->O           2   0.314   0.745  a_16_mux0006 (O_14_OBUF)
    OBUF:I->O                 4.909          O_15_OBUF (O<15>)
    ----------------------------------------
    Total                    23.859ns (14.559ns logic, 9.300ns route)
                                      (61.0% logic, 39.0% route)

==========================================================================


Total REAL time to Xst completion: 9.00 secs
Total CPU time to Xst completion: 9.38 secs

-->

Total memory usage is 360100 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    1 (   0 filtered)
```

**Fig (h): Timing Report of 8-bit Booth Multiplier**

From the timing reports above, it can be seen that the difference between Vedicand Array is almost about 10 seconds for 8-bit multipliers. As the number of bits increase, this difference increases further. Hence, Vedic multipliers reduce the delay and increases the speed.

Based on the study of the summary report of all the three multipliers, they can be compared based on the following grounds:

| Parameter | Array Multiplier | Booth Multiplier | Vedic Multiplier |
|---|---|---|---|
| Operation speed | Less | Highest | High |
| Area | Maximum area | Medium area | Minimum area |
| Time delay | More | Less | Less |
| Complexity | Less complex | More complex | More complex |
| Power consumption | Most | More | Less |
| FPGA implementation | Less efficient | More efficient | Most efficient |

This comparison is essential in choosing the type of multiplier to be used based on the requirement. It shows that Vedic multiplier is better than the existing multipliers in more than one aspects. For deciding between Vedic and booth multiplier, the multiplier best suitable for the current application should be considered as there are trade-offs between various factors.

## III. SIMULATION RESULTS

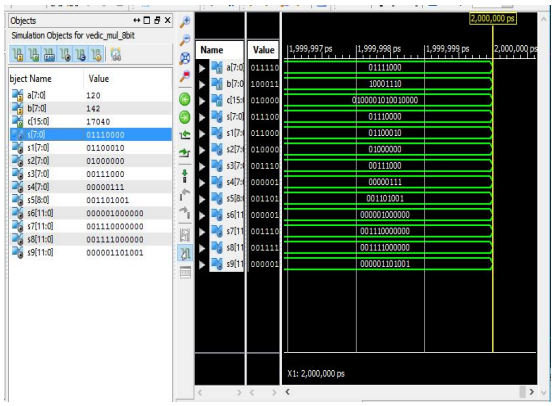


**Fig (i): Simulation result of 8bit Vedic Multiplier**

## IV. CONCLUSION

Ancient mathematics which is also known as Vedic mathematics deals with various topics such as basic arithmetic, geometry, trigonometry, calculus etc. All these methods are very efficient as far as manual calculations are concerned. It gives us a clue of symmetric computation. Vedic multiplier has one main advantage, delay increases slowly as the input bits increases. The proposed Vedic multiplier proves to be highly efficient in terms of the speed. Since the multiplication time is generally far greater than the addition time, the total processing time for maximum processors primarily depends upon the number of multiplications. Hence, we can conclude that the proposed Vedic multiplier proves to be a better option over conventional multipliers used in several expeditious and complex VLSI circuits. Upon comparison based on speed, with two other popular multipliers, Vedic multiplier proves to be a better option. Hence, it can be considered as a potential alternative of the existing conventional multipliers.

## REFERENCE


[1] Charles. Roth Jr. "Digital Systems Design using VHDL," Thomson Brooks/Cole, 7th reprint, 2005.

[2] Jagadguru Swami Sri Bharati KrisnaTirthaji Maharaja, Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda, Delhi (1965).

[3] P. D. Chidgupkar and M. T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", Global J. of Engg. Edu, vol.8, no.2, 2004.

[4] Shamim Akhter," VHDL Implementation of Fast NXN Multiplier Based on Vedic Mathematics", Jaypee Institute of Information Technology University, Noida, 201307 UP, INDIA, 2007 IEEE.

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

[6] Wallace, C.S., "A suggestion for a fast multiplier," IEEE Trans. Elec. Computer., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[7] Booth, A.D., "A signed binary multiplication technique," Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236– 240, 1951.

[8] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur and Girish V A, "Implementation of Vedic Multiplier for Digital Signal Processing", International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011.

[9] G.Ganesh Kumar, V.Charishma, "Design of High Speed Vedic Multiplier using Vedic Mathematics Techniques", International Journal of Scientific and Research Publications, Volume 2, Issue 3, March 2012.

[10] M. Uma MaheswaraSainath, B.Sekhar, "High Speed Vedic Multiplier", International Journal of Engineering Research, Volume No., 22 March 2014.

[11] Kaustubh M. Gaikwad, Mahesh S. Chavan, "Vedic Mathematics for Digital Signal Processing Operations: A Review", International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 18, March 2015.

[12] AsmitaHavelia, 2012 "FPGA implementation of a vedic convolution algorithm", International Journal of Engineering research and applications, Vol.2, issue 1, Jan-Feb 2012, 678-884.

[13] Akanksha Kant, Shobha Sharma, "Applications of Vedic Multiplier Designs-A Review", IEEE.

[14] Honey Durga, Tiwari, GanzorigGankhuyag, Chan Mo Kim, Yong Beom Cho 2008, "Multiplier design based on ancient Indian Vedic Mathematics", International SoC Design Conference, pp 65-68.

[15] Eppili Jaya, K. Chidambara Rao, "Power, Area and Delay Comparisons of Different Multipliers", International Journal of Science, Engineering and Technology Research (IJSETR) Volume 5, Issue 6, June 2016.