ISSN: 2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

Vhdl Implementation and Comparison of Complex Multiplier Using Booth's and Vedic Algorithm

Rajashri K. Bhongade, Sharada G.Mungale, Karuna Bogawar
Priyadarshini college of Engineering

Abstract: For designing of complex number multiplier basic idea is adopted from designing of multiplier. An ancient Indian mathematics "Vedas" is used for designing the multiplier unit. There are 16 sutra in Vedas, from that the Urdhva Tiryakbhyam sutra (method) was selected for implementation complex multiplication and basically Urdhva Tiryakbhyam sutra applicable to all cases of multiplication. Any multi-bit multiplication can be reduced down to single bit multiplication and addition by using Urdhva Tiryakbhyam sutra is performed by vertically and crosswise. The partial products and sums are generated in single step which reduces the carry propagation from LSB to MSB by using these formulas. In this paper simulation result for 4bit complex no. multiplication using Booth's algorithm and using Vedic sutra are illustrated. The implementation of the Vedic mathematics and their application to the complex multiplier was checked parameter like propagation delay.

I INTRODUCTION

Important components of many high performance systems such as FIR filters, microprocessor, digital signal processor, etc. are multipliers. Complex number multiplication is important in digital single processing, especially in DIT-FFT twiddle factor multiplied with input is complex number. Four real number multiplication and two additions or subtractions are involve in complex number multiplication .Carry needs to be diffused from the least significant bit (LSB) to most significant bit (MSB) when binary partial products are added in real number multiplication. After binary multiplication, the overall speed is limit by the addition and subtraction [4] [5].

Vedic Mathematics is based on 16-sutras and 16-sub sutras invented in (1884-1960).In Vedic mathematics there are three sutras Nikhilam Navatascaraman Dasatah, Ekadhikena Purvena and Urdhva Tiryakbhyam used for multiplication [1]. The targeted Vedic sutra (algorithm) which is suitable for all cases of multiplication is Urdhva Tiryakbhyam. The Array multiplication and Booths algorithm are the most common multiplication algorithms. The Booth algorithm was invented by Andrew_Donald_Booth in 1950, that multiplies two binary numbers. These Booth algorithms are used for multi-bit and exponential operations that require large partial results and carry registers [3], and takes comparable computational time [10]. This paper presents which multiplication algorithm is efficient for digital signal processing. The framework of this multiplication algorithm is based on Urdhva Tiryakbhyam sutra of Vedic mathematics.

This paper is organized as follows. Design of complex number multiplication using Booth-Wallace algorithm, Urdhva Tiryakbhyam sutra of Vedic mathematics is explained with examples. Compare the result in both algorithms in term of delays. Results obtained for 4-bit complex multiplication are presented. Concluding remarks are presented.

II IMPLEMENTATION

For implementing complex number multiplication, require real part and imaginary part.

$$R + j\,I = (A + j\,B)\,(C + j\,D) \qquad (1)$$

(A + j B) is first complex number,
(C + j D) is second complex number,

Gauss's algorithm for complex number multiplication gives two separate final results to calculate real and imaginary part. From equation (1) the real part of the output can be computed using (AC - BD), and the imaginary part of the result can be computed using (BC + AD). Thus four separate multiplications and addition/subtraction are required to produce the real as well as imaginary part numbers [9] [10].

A. Complex number multiplication using conventional modified Booth Wallace multiplier

Multiplication process is the critical part for any complex number multiplier design. There are three major steps involved for multiplication. Partial products are generated in first step. In second step partial product reduction to one row of final sums and carries is done. Third and final stage adds the final sums and carries to give the result. Since the Radix-

4 modified Booth algorithm is capable to reduced the number of partial products by nearly equal to half, this algorithm is used in the first step of implementation of complex number multiplication using Booth's method. In the second step Wallace tree structure is used to rapidly reduce the partial product rows to the final two rows giving sums and carries. The multiplication design based on Radix-4 modified Booth algorithm consists of two main blocks known as MBE (Modified Booth Encoding) and partial product generator as shown in Fig. 2. Wallace Tree CSA structures have been used to sum the partial products in reduced time. In this regard, combining both algorithms in one multiplier, we can expect a significant reduction in computing multiplications [11].

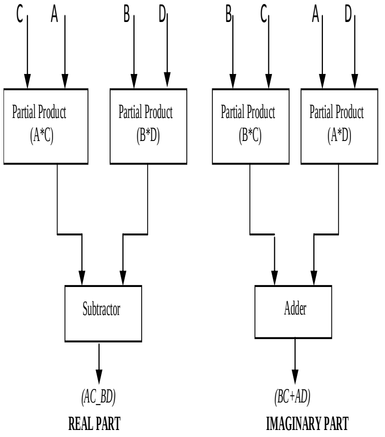


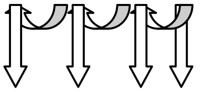
Fig.1. Block diagram of complex number multiplier

Example 1: multiplication of 3 and 4 using Booth's Method.

Step1: convert numbers into binary form i.e. 3=0011(multiplicand) and 4=0100(multiplier).
Step 2: According to Booth concentrate on multiplier

a) Copy LSB as it is
b) Go from right to left
i) If 00/11 then answer is 0,
ii) If go from 0 to 1 then answer is -1,
iii) If go from 1 to 0 then answer is 1.

0 1 0 0



1 -1 0 0

Step 3: Now multiply (0011) and (1-100)
Step 4: Multiplication is done by simple way, but by multiplying by -1 to any multiplicand take 2's complement of
That multiplicand i.e.by multiplying "-1" with 0011 then take 2's complement of 0011 as a result.

Final Result of multiplication is 00001100=12.

B. Vedic multiplication method

The Vedic Mathematics is an ancient mathematic invented by Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja. The proposed complex number multiplier is based on the Urdhva Tiryakbhyam sutra. For the multiplication of two numbers in decimal number system, these sutras have been traditionally used. In this work the same ideas are applied to binary number system, to make the proposed algorithm compatible with the digital hardware.

C. Urdhva Tiryakbhyam sutra

The basic meaning of Sanskrit word is "Vertically and crosswise". It is applicable to all multiplication. The vertical and crosswise multiplication can be implemented starting either from left hand side or from right hand side [7][8].

Meaning of vertically is straight above multiplication and meaning of crosswise is diagonal multiplication and adds them.General multiplication procedure using "Urdhva Tiryakbhyam sutra" is illustrated below.

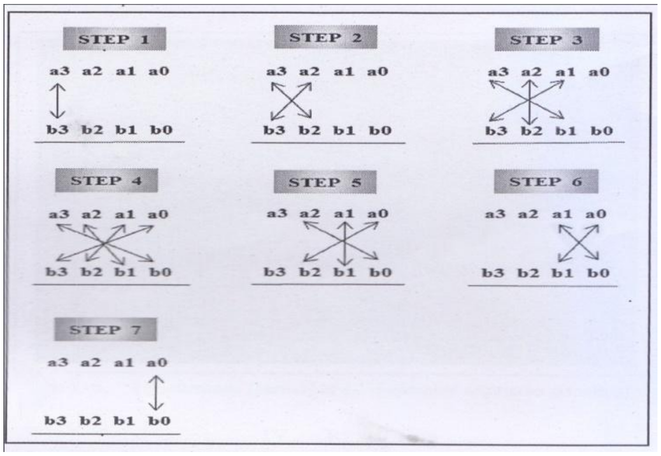


Fig2: Vertically Crosswise multiplication of binary numbers [2]

The above figure shows multiplication of 4 bit number, performance is starting from right hand side. Corresponding expression as follows:

R0=a3b3 (1)
C1R1=a2b3+a3b2 (2)
C2R2=C1+a3b1+a2b2+a1b3 (3)
C3R3=C2+a3b0+a2b1+a1b2+a0b3 (4)
C4R4=C3+a2b0+a1b1+a0b2 (5)
C5R5=C4+a1b0+a0b1 (6)
C6R6=C5+a0b0 (7)
Final product is being with C6R6R5R4R3R1R0 [3][4][7].

Example 2: Multiplication of 423 and 114

Step1: Start working from left to right, the first thing we will do is multiply both the left-hand side vertically.
1 x 4 = 4….write '4' down.

4 2 3
1 1 4

4

Step 2: Next crosswise multiply and add these result i.e. (1 x 4) + (1 x 2) = 6.

4 2 3
1 1 4

4 6

Step3: We are going to cross multiply (4 x 4) and (1 x 3), and multiply the middle term vertically (2 x 1). Add these three numbers together…16 + 3 + 2 = 21. We put down the ’1′, and carry the ’2′ over. The second term now becomes ’8′… (6 + 2).

4 2 3
1 1 4

4 8 1

Step 4: Now continue with the cross multiplication, this time multiplying (2 x 4) and (3 x 1). Add these two numbers together…8 + 3 = 11.Now write down the ’1′ as the fourth term, and carry the ’1′. Add this ’1′ to our third term (’1′)…but this now equals ’2′.

4 2 3
1 1 4

4 8 2 1

Step 5**:** Finally multiply the right-hand side vertically. (3 x 4) = 12. Write down the ’2′, and carry the ’1′. Add the ’1′ to our fourth term (’1′) to give us 2′, write that down in the fourth spot.

4 2 3
1 1 4

4 8 2 2 2

Result of multiplication 423 * 114 = 48222.

III COMPLEX MULTIPLIER

Complex multiplier design by using multiplier, adder and subtractors.Fig.3 shows the implementation of Complex number Multiplier.

Multiplication Algorithm: (Ar+jBi) and (Cr+jDi) =R+jI

<Input>:

(Ar+jBi) and (Cr+jDi) both are complex number, Ar and Cr represents real number input; and Bi and Di represents imaginary number input

<Output>:

R and I are the real and imaginary part of complex number.

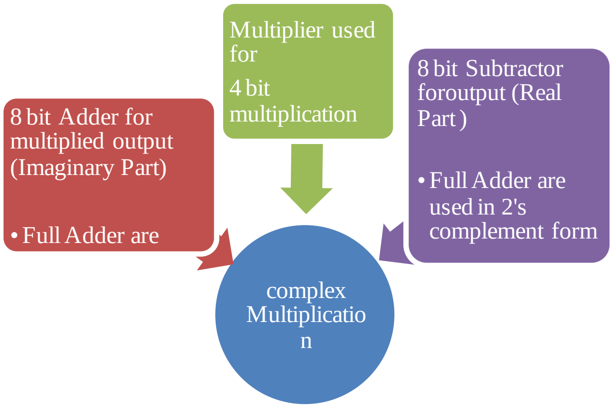

Multiplier used for
4 bit multiplication
8 bit Adder for multiplied output (Imaginary Part)
• Full Adder are
8 bit Subtractor foroutput (Real Part)
• Full Adder are used in 2's complement form
complex Multiplicatio n

Fig 3: Method for complex number Multiplication

IV DISCUSSION

By using VHDL, 4 bit complex number multiplier using Vedic mathematic (Urdhva Tiryakbham) and Booth’s algorithm is implemented.

The functional verification of the code through simulation is carried out by Xilinx ISE simulator. The complete code is synthesized using Xilinx synthesis tool (XST). Table 1 indicates the device utilization summary, comparison between architecture in term of propagation delay of the Vedic complex and Booth’s complex multiplier. Figure 4 shows the comparative chart of device utilization and figure 5 & 7, figure 6 & 8 shows the RTL schematic and synthesis report of 4-bit complex multiplier using Booth’s and Vedic algorithm respectively. Since the methodology used for the implementation of complex multiplier in both case is quite same, but in case of Vedic algorithm is less compared to Booth’s algorithm.

TABLE I. DEVICE UTILIZATION SUMMARY

Algorithm	NO. of Slices	NO. of 4 input Slices	NO. Of IO’s	Delay in nS.
Vedic complex	84	147	33	18.41
Booth’s complex	100	174	33	19.66

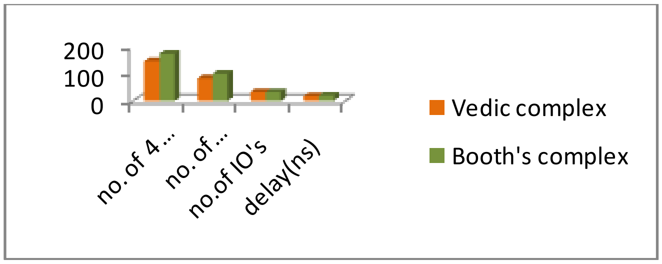


Fig.4. Comparative chart of device utilization

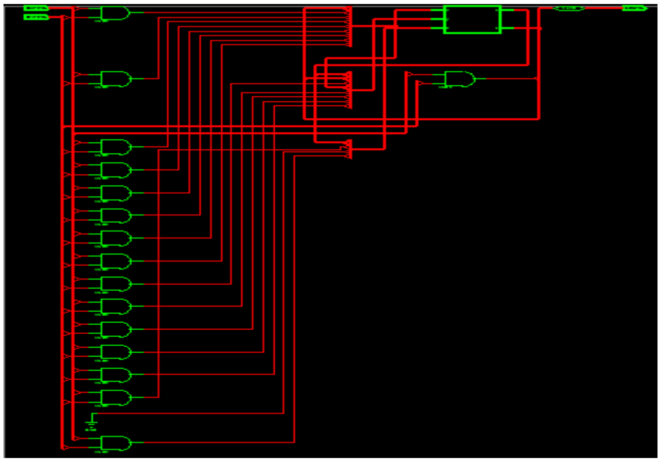
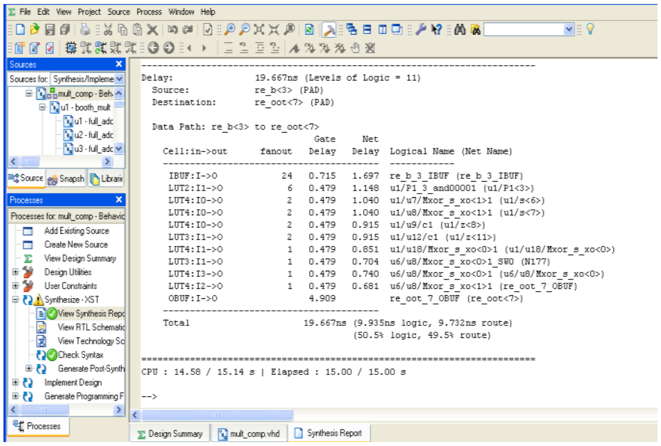


Fig 5:RTL View of 4 bit complex multiplier(Booth's Algorithm)



```
--------------------------------------------------------------------------
Delay:               19.667ns (Levels of Logic = 11)
  Source:            re_b<3> (PAD)
  Destination:       re_oot<7> (PAD)

  Data Path: re_b<3> to re_oot<7>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            24   0.715   1.697  re_b_3_IBUF (re_b_3_IBUF)
     LUT2:I1->O            6   0.479   1.148  u1/P1_3_and00001 (u1/P1<3>)
     LUT4:I0->O            2   0.479   1.040  u1/u7/Mxor_s_xo<1>1 (u1/s<6>)
     LUT4:I0->O            2   0.479   1.040  u1/u8/Mxor_s_xo<1>1 (u1/s<7>)
     LUT4:I0->O            2   0.479   0.915  u1/u9/c1 (u1/z<8>)
     LUT3:I1->O            2   0.479   0.915  u1/u12/c1 (u1/z<11>)
     LUT4:I1->O            1   0.479   0.851  u1/u18/Mxor_s_xo<0>1 (u1/u18/Mxor_s_xo<0>)
     LUT3:I1->O            1   0.479   0.704  u6/u8/Mxor_s_xo<0>1_SW0 (N177)
     LUT4:I3->O            1   0.479   0.740  u6/u8/Mxor_s_xo<0>1 (u6/u8/Mxor_s_xo<0>)
     LUT4:I2->O            1   0.479   0.681  u6/u8/Mxor_s_xo<1>1 (re_oot_7_OBUF)
     OBUF:I->O                 4.909          re_oot_7_OBUF (re_oot<7>)
    ----------------------------------------
    Total                     19.667ns (9.935ns logic, 9.732ns route)
                                       (50.5% logic, 49.5% route)

==========================================================================
CPU : 14.58 / 15.14 s | Elapsed : 15.00 / 15.00 s

-->
```

Fig.6.Synthesis report of 4-bit complex multiplier (Booth's)

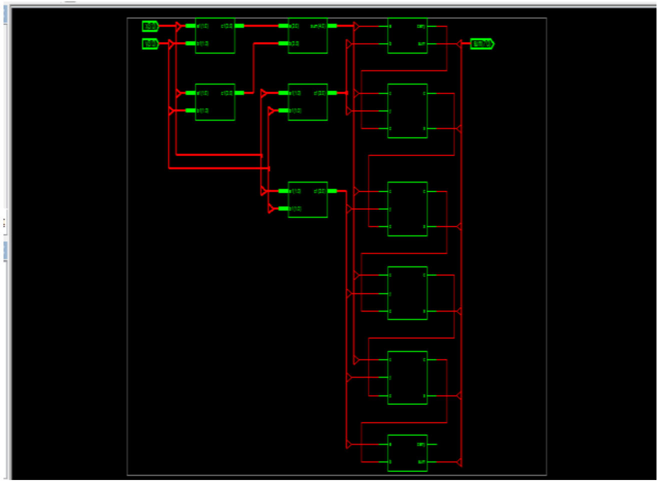
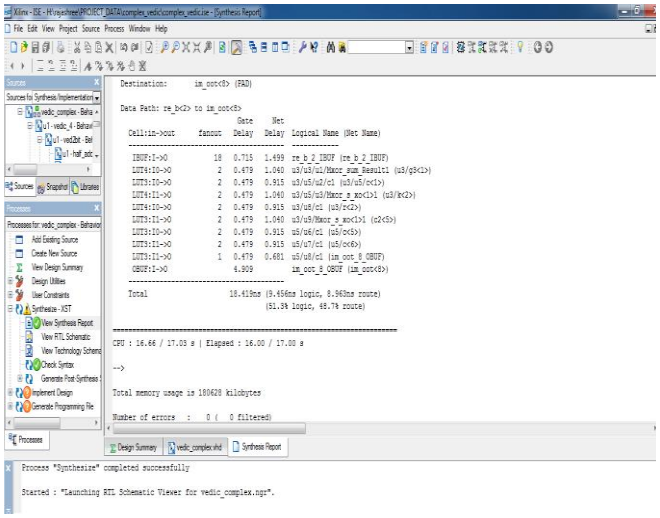


Fig 7:RTL View of 4 bit complex multiplier(Vedic)



```
Destination:       im_oot<8> (PAD)

Data Path: re_b<2> to im_oot<8>
                                Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O            18   0.715   1.499  re_b_2_IBUF (re_b_2_IBUF)
   LUT4:I0->O            2   0.479   1.040  u3/u3/u1/Mxor_sum_Result1 (u3/g3<1>)
   LUT3:I0->O            2   0.479   0.915  u3/u5/u2/c1 (u3/u5/c<1>)
   LUT4:I1->O            2   0.479   1.040  u3/u5/u3/Mxor_s_xo<1>1 (u3/k<2>)
   LUT4:I0->O            2   0.479   0.915  u3/u8/c1 (u3/r<2>)
   LUT3:I1->O            2   0.479   1.040  u3/u9/Mxor_s_xo<1>1 (c2<5>)
   LUT3:I0->O            2   0.479   0.915  u5/u6/c1 (u5/c<5>)
   LUT3:I1->O            2   0.479   0.915  u5/u7/c1 (u5/c<6>)
   LUT3:I1->O            1   0.479   0.681  u5/u8/c1 (im_oot_8_OBUF)
   OBUF:I->O                 4.909          im_oot_8_OBUF (im_oot<8>)
  ----------------------------------------
  Total                     18.419ns (9.456ns logic, 8.963ns route)
                                     (51.3% logic, 48.7% route)

==========================================================================
CPU : 16.66 / 17.03 s | Elapsed : 16.00 / 17.00 s

-->

Total memory usage is 180628 kilobytes

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

```
Process "Synthesize" completed successfully

Started : "Launching RTL Schematic Viewer for vedic_complex.ngr".
```

Fig.8.Synthesis report of 4-bit complex multiplier (Vedic)

V RESULT

The work presented in this paper was implemented using VHDL and logic simulation was done using Xilinx ISE simulator. The obtained results are presented in table I, and waveforms for 4-bit complex multiplication using Urdhva Tiryakbhyam and Booth's algorithm is shown in figure 9 & 10 respectively. The device utilization in case of Vedic complex is less (No. of Slices: 84 out of 1920 - 4%, Number of 4 input LUTs: 147 out of 3840 - 3%, compared to Booth's complex (No. of Slices: 100 out of 1920 - 5%, Number of 4 input LUTs: 174 out of 3840 - 4 %,). The delay required by Vedic complex multiplier is 18.39 ns, while it is 19.96 ns for Booth's complex.

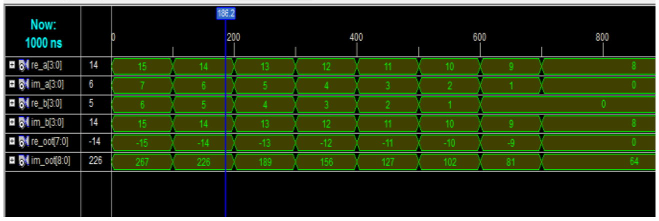

186.2
Now:
1000 ns
0
200
400
600
800
re_a[3:0] 14 15 14 13 12 11 10 9 8
im_a[3:0] 6 7 6 5 4 3 2 1 0
re_b[3:0] 5 6 5 4 3 2 1 0
im_b[3:0] 14 15 14 13 12 11 10 9 8
re_oot[7:0] -14 -15 -14 -13 -12 -11 -10 -9 0
im_oot[8:0] 226 267 226 189 156 127 102 81 64

Fig. 9. Simulation waveforms for 4-bit complex multiplication (Vedic)

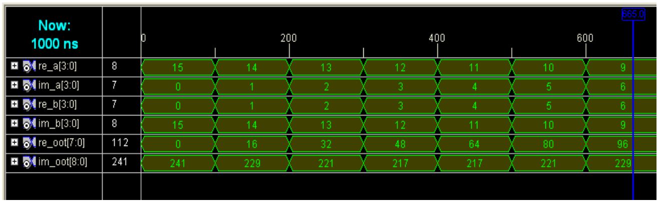

665.0
Now:
1000 ns
0
200
400
600
re_a[3:0] 8 15 14 13 12 11 10 9
im_a[3:0] 7 0 1 2 3 4 5 6
re_b[3:0] 7 0 1 2 3 4 5 6
im_b[3:0] 8 15 14 13 12 11 10 9
re_oot[7:0] 112 0 16 32 48 64 80 96
im_oot[8:0] 241 241 229 221 217 217 221 229

Fig. 10: Simulation waveforms for 4-bit complex multiplication (Booth's)

VI CONCLUSION

In many DSP applications numbers of complex multiplications are involved, in which high speed performance is a prime target. However, achieving this may be done at the expense of area, power dissipation, propagation delay and accuracy. So efforts have to be made by decrease the number of multipliers and by increase their speed. A high speed complex number multiplier design using ancient Indian Vedic Mathematics (Urdhva Tiryakbhyam sutra) is implemented using VHDL. This sutra is applicable to all cases of multiplication. The results show that Urdhva Tiryakbhyam sutra used to implement high speed complex multiplier efficiently in digital signal processing algorithms by decreasing propagation delay (ns).

VII REFERENCES

[1] Jagadguru Swami Sri Bharati Krishna Teerthaji Maharaja, "Vedic Mathematics," Motilal Banarsidas Publishers Pvt. Ltd, 2001.

[2] L. Sriraman, T. N. Prabakar, "Design and Implementation of Two Variable Multiplier Using KCM and Vedic Mathematics," 1st International Conf. on Recent Advances in Information Technology, RAIT, 2012.

[3] Sandesh S. Saokar, R. M. Banakar, and Saroja Siddamal, "High Speed Signed Multiplier for Digital Signal Processing Applications," 2012 IEEE.

[4] S.S. Kerur, Prakash Narchi Jayashree C .N., Harish M. Kittur "Implementation of Vedic multiplier for digital signal processing," international journal of computer application, 2011, vol. 16, pp 1-5.

[5] Devika Jaina, Kabiraj Sethi, and Rutuparna Pamda, "Vedic Mathematics Based Multiply Accumulate Unit," International conference on computational intelligence and communication system, 2011, pp 754-757.

[6] V Jayaprakasan, S Vjayakumar, and V S Kanchana Bhaaskaran, "Evaluation of the Conventional vs. Ancient Computation methodology for Energy Efficient Arithmetic Architecture," 2011 IEEE.

[7] Thakre L. P., et al,"Performance Evaluation and Synthesis of Multiplier used in FFT operation using Conventional and Vedic algorithms," Third International Conference on Emerging Trends in Engineering and Technology, ICETET.2010, pp 614-619

[8] Rudagi J. M.,et al, "Design And Implementation of Efficient Multiplier Using Vedic Mathematics ," International Conference on Advances in Recent Technologies in Communication and Computing, 2011,pp 162-166.

[9] H. S. Dhillon, et al, "A Reduced Bit Multiplication Algorithm for Digital Arithmetic," International Journal of Computational and Mathematical Sciences, 2008, pp 64-69.

[10] Man Yan Kong, J.M. Pierre, and Dhamin Al-Khalili, "Efficient FPGA Implimentation of Complex Multipliers Using the Logarithmic Number System," 2008 IEEE. Pp 3154-3157.

[11] Langlois Rizalafande Che Ismail and Razaidi Hussin, " High Performance Complex Number Multiplier Using Booth-Wallace Algorithm," *ICSE2006 Proc. 2006, Kuala Lumpur, Malaysia, pp* 786-790.

[12] Deena Dayalan, S.Deborah Priya , "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques," ACTEA IEEE July 15-17, 2009 Zouk Mosbeh, Lebanon, PP 600-603.