IJETST- Volume||01||Issue||05||Pages 589-596||July||ISSN 2348-9480 2014

International journal of Emerging Trends in Science and Technology

Simulation & Implementation of Complex Multiplier using Vedic Mathematics

Authors

Aruna. M, Usharani. G

PG Student (M. Tech), Dept. of ECE, REVA ITM Bangalore India
Email: *aruna.ym@gmail.com*
Sr. Assistant Professor, Dept. of ECE, REVA ITM Bangalore India
Email: *usharani@revainstitution.org*

Abstract
In VLSI technology speed optimization plays a vital role. So designing of high speed devices became necessary to fulfill the end user requirements. Generally the processor designing is mainly depending upon the MAC units. In that particularly multiplier architecture comes under crucial design. Vedic Mathematics is the ancient methodology of Indian mathematics which has a unique technique of calculations based on 16 Sutras (Formulae). In this paper the importance of Urdhva tiryakbhyam Sutra and Nikhilam Sutra are discussed. The design of complex multiplier designed using this sutra consists of Radix Selection Unit (RSU), Exponent Determinant (ED), Mean Determinant (MD) and Comparator. The multiplier shows the product of the provided inputs with reduced latency along with optimized power estimation. Transistor level implementation of Vedic Mathematics based 16-bit complex multiplier for high speed low power processor is reported in this paper. The functionality of these circuits was checked and performance parameters like propagation delay and dynamic power consumption were calculated by Xilinx sparten3E using standard 90nm CMOS technology for synthesis and simulation. The propagation delay of the resulting (16, 16) x (16, 16) complex multiplier was found to be 4ns and consume 81mW power. The implementation offered significant improvement in terms of delay and power from earlier reported ones.
Keywords—*Vedic Mathematics, Nikilam sutra, partial product, Exponent & Mean determinant, Xilinx.*

INTRODUCTION

Multiplication is an important fundamental function in arithmetic operations. Multiplication-based operations such as Multiply and Accumulate(MAC) and inner product are among some of the frequently used Computation Intensive Arithmetic Functions(CIAF) currently implemented in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform(FFT), filtering and in microprocessors in its arithmetic and logic unit [1]. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. Currently, multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip.

The demand for high speed processing has been increasing as a result of expanding computer and signal processing applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal and image processing applications [2]. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications [2, 3]**.** This paper presents complex number multiplier design based on Vedic Mathematics which is is one of the fast and low power multiplier**.**

Minimizing power consumption for digital systems involves optimization at all levels of the design. This optimization includes the technology used to implement the digital circuits, the circuit style and topology, the architecture for implementing the circuits and at the highest level the algorithms that are being implemented. Digital multipliers are the most commonly used components in any digital

circuit design. They are fast, reliable and efficient components that are utilized to implement any operation. Depending upon the arrangement of the components, there are different types of multipliers available. Particular multiplier architecture is chosen based on the application.

In many DSP algorithms, the multiplier lies in the critical delay path and ultimately determines the performance of algorithm. The speed of multiplication operation is of great importance in DSP as well as in general processor. In the past, multiplication was implemented generally with a sequence of addition, subtraction and shift operations. There have been many algorithms proposals in literature to perform multiplication, each offering different advantages and having tradeoff in terms of speed, circuit complexity, area and power consumption.

The multiplier is a fairly large block of a computing system. The amount of circuitry involved is directly proportional to the square of its resolution i.e. A multiplier of size n bits has n^2 gates. For multiplication algorithms performed in DSP applications latency and throughput are the two major concerns from delay perspective. Latency is the real delay of computing a function, a measure of how long the inputs to a device are stable is the final result available on outputs. Throughput is the measure of how many multiplications can be performed in a given period of time; multiplier is not only a high delay block but also a major source of power dissipation. That's why if one also aims to minimize power consumption, it is of great interest to reduce the delay by using various delay optimizations.
Digital multipliers are the core components of all the digital signal processors (DSPs) and the speed of the DSP is largely determined by the speed of its multipliers. Two most common multiplication algorithms followed in the digital hardware are array multiplication algorithm and Booth multiplication algorithm. The computation time taken by the array multiplier is comparatively less because the partial products are calculated independently in parallel. The delay associated with the array multiplier is the time taken by the signals to propagate through the gates that form the multiplication array. Booth multiplication is another important multiplication algorithm. Large booth arrays are required for high speed multiplication and exponential operations which in turn require large partial sum and partial carry registers. Multiplication of two *n*-bit operands using a radix-4 booth recording multiplier requires approximately $n / (2m)$ clock cycles to generate the least significant half of the final product, where *m* is the number of Booth recorder adder stages. Thus, a large propagation delay is associated with this case. Due to the importance of digital multipliers in DSP, it has always been an active area of research and a number of interesting multiplication algorithms have been reported in the literature [4].
In this paper, Urdhva tiryakbhyam Sutra is first applied to the decimal number system and is used to develop digital multiplier architecture. This is shown to be very similar to the popular array multiplier architecture. Nikhilam Sutra is then discussed and is shown to be much more efficient in the multiplication of large numbers as it reduces the multiplication of two large numbers to that of two smaller ones.
The proposed multiplication algorithm is then illustrated to show its computational efficiency by taking an example of 16X16-bit complex number multiplication. This paper presents a systematic design methodology for fast and area efficient 16 bit complex multiplier based on Vedic mathematics. The Multiplier Architecture is based on the Vertical and Crosswise algorithm of ancient Indian Vedic Mathematics [5].

INTRODUCTION TO VEDIC SUTRAS

A. Urdhva Tiryakbhyam Sutra

The multiplier used in this paper is based on Urdhva Tiryakbhyam (Vertical & Crosswise) sutra, of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and crosswise". It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products. The parallelism in generation of partial products and their summation is obtained using Urdhava Triyakbhyam. The algorithm can be generalized for n x n bit number. Since the partial products and their sums are calculated in parallel, the multiplier is independent of the clock frequency of the processor. Thus the multiplier will require the same amount of time to calculate the product and hence is independent of the clock frequency.
The net advantage is that it reduces the need of microprocessors to operate at increasingly high clock frequencies. While a higher clock frequency generally results in increased processing power, its disadvantage is that it also increases power dissipation which results in higher device operating temperatures. By adopting the Vedic multiplier, microprocessors designers can easily circumvent these problems to avoid catastrophic device failures. The processing power of multiplier can easily be increased by increasing the input and output data bus widths since it has a quite a regular structure. Due to

its regular structure, it can be easily layout in a silicon chip. To illustrate this multiplication scheme, let us consider the multiplication of two decimal numbers (325 * 738). Line diagram for the multiplication is shown in Fig.1. The digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and hence the process goes on. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and all other bits act as carry for the next step. Initially the carry is taken to be zero.

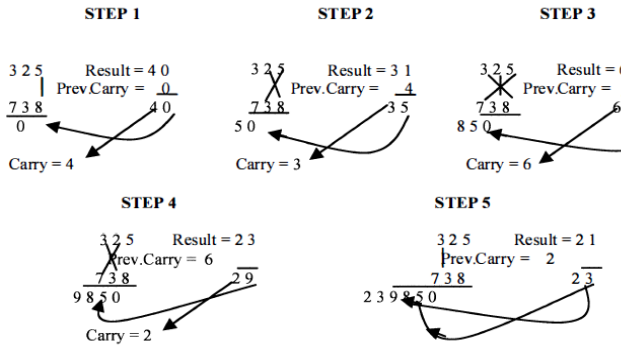

Figure: 1 Multiplication of two decimal numbers by Urdhva Tiryakbhyam

The 4 x 4 multiplication has been done in a single line in Urdhva method, whereas in shift and add method (Conventional) four partial products have to be added to get the result. This implies the increase in speed. Urdhva Tiryakbhyam (Vertically and Crosswise), deals with the multiplication of numbers [8], [9], [12].

B. Nikhilam Navatascaramam Dasatah Sutra

Nikhilam Sutra literally means "all from 9 and last from 10". Although it is applicable to all cases of multiplication, it is more efficient when the numbers involved are large. It finds out the compliment of the large number from its nearest base to perform the multiplication operation on it, hence larger the original number, lesser the complexity of the multiplication. We first illustrate this Sutra by considering the multiplication of two decimal numbers (97× 94) where the chosen base is 100 which is nearest to and greater than both these two number.

As shown in Fig 2 below, we write the multiplier and the multiplicand in two rows followed by the differences of each of them from the chosen base, i.e., their compliments. We can now write two columns of numbers, one consisting of the numbers to be multiplied (Column 1) and the other consisting of their compliments Column 2). The product also consists of two parts which are demarcated by a vertical line for the purpose of illustration. The right hand side (RHS) of the product can be obtained by simply multiplying the numbers of the Column 2 (3×6 = 18). The left hand side (LHS) of the product can be found by cross subtracting the second number of Column 2 from the first number of Column 1 or vice versa, i.e., 94 - 3 = 91 or 97 - 6 = 91. The final result is obtained by concatenating RHS and LHS (Answer = 9118).

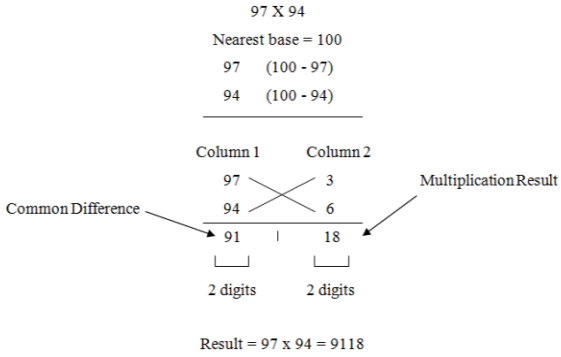

Figure 2: Multiplication using Nikhilam Navatascaramam Dasatah Sutra

MATHEMATICAL REPRESENTATION OF URDHAV TRIYAMBHYAM SUTRA

Assume that X and Y are two numbers, to be multiplied. Mathematically X and Y can be represented as:

$$A = \sum_{i=0}^{n-1} A_i 10^i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (11)$$

$$B = \sum_{j=0}^{n-1} B_j 10^j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (12)$$

Assume that, their product is equal to Z. Then Z can be represented as:

$$Z = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} A_i B_j 10^{i+j} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (13)$$

Where (Ai, Bj) ∈ (0, 1, 2... ... . 9) and 'n' may be any number. From the above expression, equation no (13), it can observed that each digit is multiplied consecutively and shifted towards the proper positions for partial product generation. Finally the partial products are added with the previous carry to produce the final results.

MATHEMATICAL REPRESENTATION OF NIKHILAM SUTRA

Assuming A and B are two n-bit numbers to be multiplied and their product is equals to P.

$A = \sum_{i=0}^{n-1} A_i 10^i$ where $A_i \in \{0, 1 \ldots 9\}$..... (1)

$B = \sum_{i=0}^{n-1} B_i 10^i$ where $B_i \in \{0, 1 \ldots .9\}$ ..... (2)

Multiplication Rule

P=AB..... (3)

Equation (3) can be reformulated as by adding and subtracting the $10^{2n}+ 10^n (A +B)$ in the right hand side.

$P = AB + 10^{2n} + 10^n (A + B) – 10^{2n} – 10^n (A+ B)$… ..... (4)

$P= \{10^n (A + B) – 10^{2n}\} + 10^{2n} – 10^n (A+ B) + AB$ ..... (5)

Equation no (5) can be derived for both the numbers if the number is greater than the base or less than the base.

If the number is greater than the base:

$P= \{10^n (A + B) – 10^n \} + (10^n – A) (10^n – B)$ …. (6)

If the number is less than the base:

$P= \{10^n (A + B) – 10^n\} + (A – 10^n) (B – 10^n)$..... (7)

$P= \{10^n \{A – (10^n – B)\} + (10^n – A) (10^n – B)$…. (8)

$P= 10^n (A – \bar{B}) + A\bar{B}$ ..... (9)

Where $\bar{A}$ and $\bar{B}$ are the $10^{n's}$ complement of A and B.

Subtraction Rule:

$\bar{A}= 10^n – \sum_{i=0}^{n-1} A_i 10^i$ ..... (10)

$\bar{A}= \sum_{i=0}^{n-1} (9-A_i) 10^i + (10 – A_0)$..... (11)

The serious drawback of Nikhilam sutra can be summarized as:

(i) both the multiplier and multiplicand must be less or greater than the base.

(ii) Multiplier and multiplicand must be nearer to the base.

Hardware Implementation Of Vedic Multiplier

Consider two n bit numbers X and Y, kl and k2 are the exponent of X and Y respectively. X and Y can be represented as:

$X = 2^{k1} \pm Z1$..... (12)

$Y = 2^{k2} \pm Z2$..... (13)

Assuming product of the number is equals to P.

$P=X*Y= (2^{k1} \pm Z_1) (2^{k2} \pm Z_2)$..... (14)

For the fast multiplication using Nikhilam sutra the bases of the multiplicand and the multiplier should have different base, thus the equation no 14 can be rewritten as:

$Y \times 2^{k1-k2} = 2^{k1} \pm Z_2 2^{k1-k2}$..... (15)

$X \times Y \times 2^{k1-k2} = (2^{k1} \pm Z1) (2^{k1} \pm Z_2 2^{k1-k2})$

$= 2^{2k1} \pm Z_1 2^{k1} \pm Z_2 2^{2k1-k2} \pm Z_1 Z_2 2^{k1-k2}$

$= 2^{k1}(2^{k1} \pm Z_1 \pm Z_2 2^{k1-k2}) \pm Z_1 Z_2 2^{k1-k2}$

$=2^{k1}(X \pm Z_2 2^{k1-k2}) \pm Z_1 Z_2 2^{k1-k2}$ ..... (16)

$P = XY = 2^{k2} (X \pm Z_2 2^{k1-k2}) \pm Z_1 Z_2$ ..... (17)

From equation no 17 it is observed that a large number multiplication can easily decomposed into a small number multiplication, addition/subtraction and shifting, leading towards the reduction of hardware cost, propagation delay and power consumption.

The mathematical expression for the proposed algorithm is shown in equation no 17. Hardware implementation of this mathematics is shown in Fig. 3. It consists of five major segments such as:- (i) RSU, (ii) Subtractor, (iii) Adder/Subtractor, (iv) Multiplier and (v) Shifter. Since the radix is same for both the inputs so the radix generated from RSU corresponding to input X is fed to the sub-tractor. The other input to the sub-tractor is Y. The output generated is the residue corresponding to Y. Both the residues are multiplied through the multiplier. The residue corresponding to Y is added with or

subtracted from X by the first adder/subtractor block. The result is shifted left based on the exponent generated from RSU. The multiplier result is added with or subtracted from the shifted result by the second adder/subtractor.

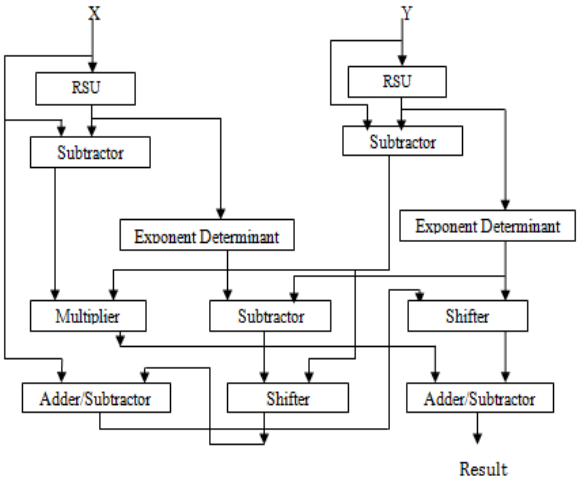

Figure 3: Hardware implementation of Nikilam sutra

C. Mathematical Representation of RSU

Consider an 'n' bit binary number X, and it can be represented as

$X = \sum_{i=0}^{n-1} X_i 2^i$ where $X_i \in \{0, 1\}$

Then the values of X must lie in the range $2^{n-1} \leq X < 2^n$.

Consider the mean of the range is equals to A.

$$A = 1 + \frac{2^{n-1}+2^n}{2}$$
$$= 2^{n-2} \times 3$$

If X >A Then radix = 2^n
If Then X <A radix = 2^{n-1}

The Hardware representation of RSU is shown in Fig. 4. The maximum power of X is extracted at the output which is again fed to shifter and the adder block. The second input to the shifter is the (n+1) bit representation of decimal '1 '. If the maximum power of X from the ED unit is (n-1) then the output of the shifter is $2^{(n-1)}$. The adder unit is needed to increment the value of the maximum power of X by '1'. The second shifter is needed to generate the value of 2^n. Here n is the incremented value taken from the adder block. The Mean Determinant unit is required to compute the mean of $(2^{n-1} + 2^n)$. The comparator compares the actual input with the mean value of $(2^{n-1} + 2^n)$. If the input is greater than the mean then 2^n is selected as the required radix. If the input is less than the mean then 2^{n-1} is selected as the radix. The select input to the multiplexer block is taken from the output of the comparator.

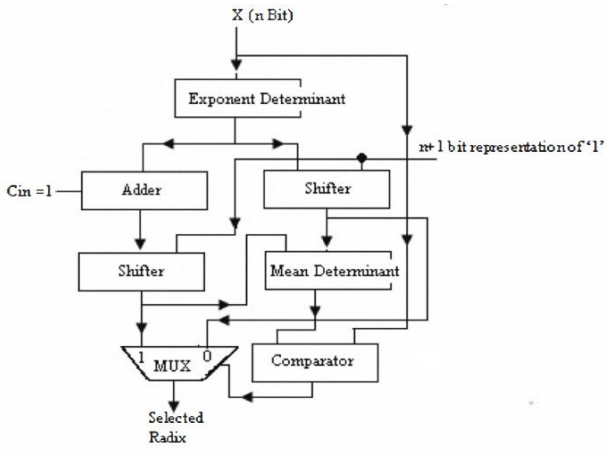

Figure 4: Hardware representation of RSU

D. Exponent Determinant

The integer part or exponent of the number from the binary fixed point number can be obtained by the maximum power of the radix. For the non-zero 16 bit input, A parallel searching procedure has been implemented here using a priority encoder to search the first '1' starting from the MSB. When the searched bit is '1' the priority encoder shows the integer part (exponent) of the number as at which position among 16 bits '1' is present from MSB to LSB.

COMPLEX MULTIPLIER

Complex multiplier design by using parallel adders and subtractors has been designed by Saha[9]. Fig. 5 shows the direct method for implementation of the complex multiplier design. In this paper complex multiplier design is done using Vedic Multipliers and Vedic subtractors.

Multiplication Algorithm

<Input>

A and B: Multiplicand and multiplier respectively. Both are complex numbers. A=Ar +jA; and B =Br +jB; (here all are N Bit unsigned numbers).

Step1. Select the appropriate base using RSU.

Step2. Multiply the numbers and then add or subtract them to obtain the real and the imaginary part of the result.

<Output>

Result: Cr and C; are the real and imaginary part of complex number

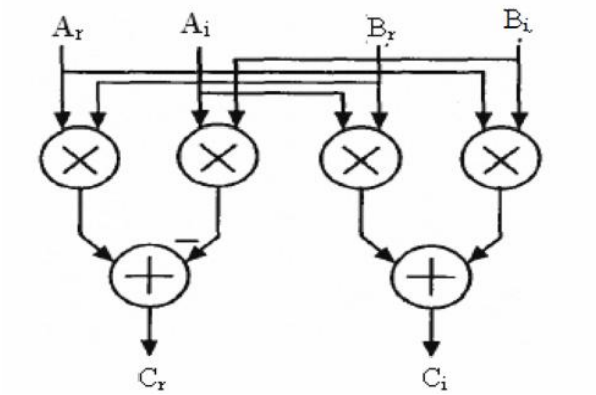

Figure: 5 Implementation method of Complex multiplier

SIMULATION RESULTS

The simulation results are shown in the respective figure along with the notations of important signals. The results obtained from simulation are validated using Microsoft Windows® Calculator version 6.1(Build 7601: Service Pack 1)

The Appropriate test cases have been identified in order to test this modeled Complex multiplier design. Based on the identified values, the simulation results which describes the operation of the Complex multiplier design has been achieved. This proves that the modeled design works properly as per the process.

The performance parameters analysis using Vedic multiplication simulation output is shown in Test case below. Input data is taken as a regular fashion for experimental purpose. We have kept our main concentration for reducing the propagation delay.

Testcase:

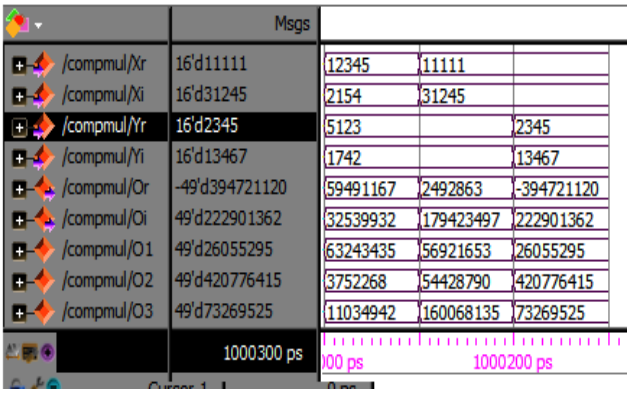

Figure 6: Simulation Results

Where Xr is real input of 1st number

Xj is the imaginary input of 1st number

Yr is real input of 2nd number

Yj is the imaginary input of 2nd number

Or is the real part output

Oi is the imaginary part output

Thus the simulated results are calculated and match correctly.

Timing summary

Speed Grade: -4

Minimum period: No path found

Minimum input arrival time before clock: 57.526ns

Maximum output required time after clock: 4.283ns

Maximum combinational path delay: No path found

Table 1: Comparison of delay obtained from synthesis results for different methods of complex multiplier algorithms for 16 bit maximum digit inputs.

Multiplier Type	Architecture used	Delay (ns)
16*16	Blahut[10]	6
16*16	Distributed Algorithm[11]	2.5
16*16	Saha[9]	5
16*16	Proposed Vedic Complex Multiplier	4

Table 1: Comparison of delay

Device Utilization Summary

Selected Device: 3s500efg320-5
Number of GCLKs: 1 out of 24 4%
Minimum period: 4.283ns
Total estimated power consumption: 81 mW

Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	5,176	9,312	55%
Number of occupied Slices	2,729	4,656	58%
Number of bonded IOBs	129	232	55%

Table: 2 Device Utilization Summary

CONCLUSION

In this paper we report on complex number multiplier simulation and implementation. There are various conventional and Vedic methods of multipliers available today. Many systems need complex multipliers for various applications. However, to achieve good performance in terms of speed, area occupied and minimum power consumption, appropriate algorithm must be implemented in hardware. In this paper, we have attempted for finding such an algorithm, and Vedic method is found suitable and shown that it can be implemented in hardware by obtaining synthesis results.

The computation delay for 16x16 bits Vedic multiplier was found to be 4ns. It is therefore seen that the Vedic multipliers are much faster than the conventional multipliers. The algorithms of Vedic mathematics are much more efficient than of conventional mathematics. Comparing their simulation and synthesis results, Vedic complex multiplier has found to be advantageous and is implementable. Hence "Simulation & Implementation of complex multiplier using Vedic Mathematics" is discussed and analyzed extensively in this paper. The techniques are simulated using hardware description language, Verilog and synthesized using Xilinx Sparten 3E Board.

REFERENCES

1. Asati and Chandrashekhar, "An Improved High Speed fully pipelined 500 MHz 8×8 Baugh Wooley Multiplier design using 0.6 µm CMOS TSPC Logic Design Style", Proc. IEEE, ICIINFS 2008, pp. 1-6, Dec. 8-10, 2008.
2. Chandrakasan and R. Brodersen, "Low-power CMOS digital design" , IEEE Journals on Solid-State Circuits, Vol. 27, No. 4, pp. 473–484, Apr. 1992.
3. N.-Y. Shen and O. T.-C. Chen, "Low-power multipliers by minimizing switching activities of partial products", Proc. IEEE, ISCAS 2002, vol. 4, pp. 93–96, May 2002.
4. O. T. Chen, S. Wang, and Y.-W. Wu, "Minimization of switching activities of partial products for designing low-power multipliers," IEEE Transc. on Very Large Scale Integration (VLSI) Syst., Vol. 11, No. 3, pp. 418–433, June 2003.
5. Parhami, Computer Arithmetic Algorithms and Hardware Designs, 1st ed. Oxford, U.K.: Oxford Univ. Press, 2000.
6. S Wallace, "A Suggestion for a Fast Multiplier," IEEE Trans. on Computers, Vol. EC13, pp. 14-17, December 1964.
7. J. Hu, L. Wang, and T. Xu, "A Low-Power Adiabatic Multiplier Based on Modified Booth Algorithm", Proc. IEEE, ISIC'07, pp. 489-492, Sept. 26-28, 2007.
8. R. Baugh, and B.A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm", IEEE transc. on Computers, Vol. C-22, No. 12, pp. 1045-1047, December 1973
9. K-C. Kuo, and C-W. Chou, "Low power and high speed multiplier design with row bypassing and parallel architecture, Journal of Microelectronics, 2010, doi:10.1016/j.mejo.2010.06.009
10. P. K. Saha, A. Banerjee, and A. Dandapat, "High Speed Low Power Complex Multiplier Design Using Parallel Adders and

Subtractors," *International Journal on Electronic and Electrical Engineering, (/EEE)*, vol 07, no. II, pp 38-46, Dec. 2009.

11. R. E. Blahut, *Fast Algorithms for Digital Signal Processing*, Reading, MA: Addison-Wesley, 1987.
12. S. He, and M. Torkelson, "A pipelined bit-serial complex multiplier using distributed arithmetic," in *proceedings IEEE International Symposium on Circuits and Systems*, Seattle, WA, April 30-May-03, 1995,pp. 2313-2316.
13. J. E. VoIder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput*, vol. EC-8, pp. 330-334, Sept. 1959.
14. R. Krishnan, G. A. Jullien, and W. C. Miller, "Complex digital“ Spartan-3E FPGA Starter Kit Board User Guide”, UG230 (v1.1) June 20, 2008.
15. Jagadguru Swami Sri Bharati Krsna Tirthaji Maharaja, “Vedic mathematics”, Motilal Banarsidass Publishers Pvt Ltd, Delhi