International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 9, September - 2013

Design of Two Variable Multiplier Using Vedic Mathematics and ROM Approach

M.Ganesh Kumar [1], I.V.Rameswar Reddy [2], K.Kameswara Reddy [3]
[1] PG Scholar, M.Tech VLSI, Department of ECE, AVR & SVR Engineering College, Kurnool
[2]Associate Professor, Department of ECE, AVR & SVR Engineering College, Kurnool
[3]Assistant Professor, Department of ECE, AVR & SVR Engineering College, Kurnool

ABSTRACT:- This article presents the design of a new high-speed multiplier architecture using Urdhava Tiryagbhyam Sutra of Vedic mathematics [1]. The architecture of this multiplier uses the concept of Constant Coefficient Multiplier (KCM). For KCM one input is fixed, while the proposed multiplier can multiply two variables. In this proposed method, the squares of numbers are stored in ROM which makes the multiplication faster. The multiplier circuit is synthesised and simulated using Xilinx ISE 12.2 software and the results are presented. The proposed multiplier is compared with Array Multiplier for 16 bit, 32 bit and 64 bit cases. In this article a layout has been developed for 4x4 Urdhava Tiryagbhyam Multiplier. The proposed multiplier is 1.5 times faster than the other multipliers.

KEYWORDS:- Array Multiplier, KCM, Urdhava Multiplier, Vedic Mathematics;

1. INTRODUCTION

Multiplication is a complex arithmetic operation, which is rejected in its relatively high signal propagation delay, high power dissipation and large area requirement. When choosing a multiplier for a digital system, the bitwidth of the multiplier is required to be at least as wide as the largest operand of the applications that are to be run on that digital system. The bitwidth of the multiplier is, therefore, often much larger than its operands, which leads to excessive power dissipation and long delay. This could partially be remedied by having several multipliers, each with a specific bitwidth, and using the particular multiplier with the smallest bitwidth that is large enough for the current multiplication.

A system's performance is generally determined by the performance of the multiplier, because the multiplier is generally the slowest element in the system. Ever increasing performance requirements make it challenging to implement multipliers that are efficient in terms of throughput, delay, power, and area for a wide range of bitwidths.

2. ARRAY MULTIPLIER

In Array multiplier [2], AND gates are used for generation of the bit-products and adders for accumulation of generated bit products. All bit-products are generated in parallel and collected through an array of full adders or any other type of adders. Since the array multiplier is having a regular structure, wiring and the layout are done in a much simplified manner. Therefore, among other multiplier structures, array multiplier takes up the least amount of area.

Example 1 describes the multiplication process using array multiplier and Figure 2.l depicts the structure of the same. A partial product enters at the top, the Pij values are added, and a new partial product emerges at the bottom, in carry-save form. In an actual layout, successive rows would probably be shifted to the right so that the adder cells would form a perfect rectangular array.

Example 1: (1011 x 1101) = 10001111

1 0 1 1
1 1 0 1 x
1 0 1 1
0 0 0 0 → Left Shift by 1 bit
1 0 1 1 → Left Shift by 2 bit
1 0 1 1 →Left Shift by 3 bit
1 0 0 0 1 1 1 1

Here from the above example it is inferred that partial products are generated sequentially, which reduces the speed of the multiplier. However the

structure of the multiplier is regular.

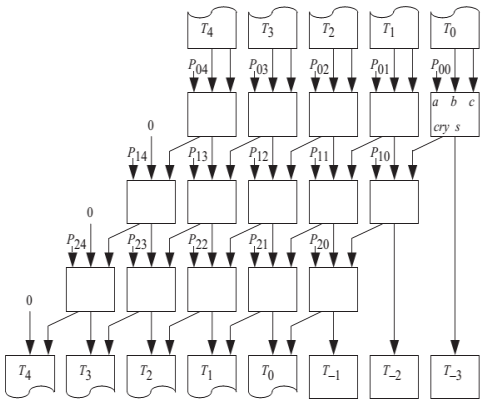

Figure 2.1: The multiplier array of adder cells.

3. URDHAVA MULTIPLIER

Urdhava Tiryakbhyam [1][3] is a Sanskrit word which means vertically and crosswire in English. The method is a general multiplication formula applicable to all cases of multiplication. It is based on a novel concept through which all partial products are generated concurrently. Fig. 2 demonstrates a 4 x 4 binary multiplication using this method.

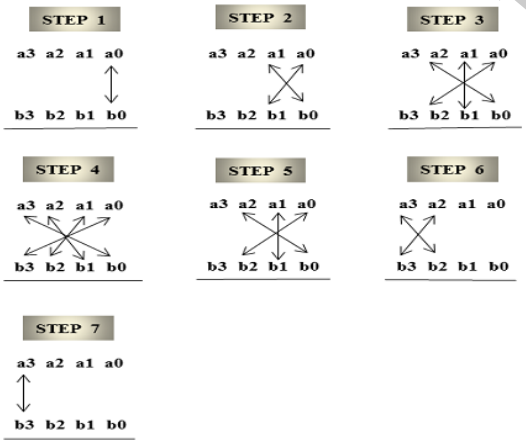

Fig. 2 Multiplication of two 4 bit numbers using Urdhava Tiryakbhyam method.[7]

The method can be generalized for any N x N bit multiplication. Due to its regular structure, it can be easily layout in a silicon chip and also consumes optimum area.

The line diagram in fig 2 illustrates the algorithm for multiplying two 4-bit binary numbers a3,a2,a1,a0 and b3,b2,b1,b0. The procedure is divided into 7 steps and each step generates partial products. Initially as shown in step 1 of fig2, the least significant bit (LSB) of the multiplier is multiplied with least significant bit of the multiplicand (vertical multiplication). 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. 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 multiplied according to Urdhava Tiryakbhyam (vertically and crosswire) method, we get,

AH AL
BH BL
_____
(AH x BH) + (AH x BL + BH x AL) + (AL x BL).

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.

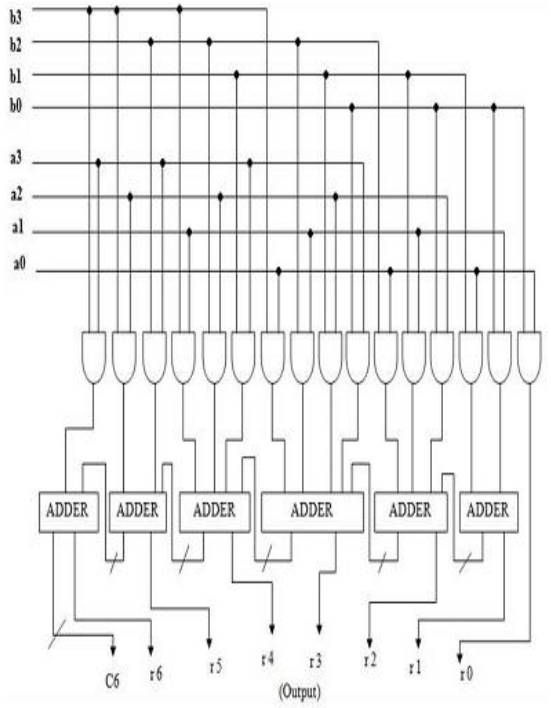

Fig. 3.Hardware architecture of 4 X 4 Urdhva Tiryakbhyam multiplier. [5]

4. PROPOSED METHOD

The proposed method is based on ROM approach however both the inputs for the multiplier can be variables. In this proposed method a ROM is used for storing the squares of numbers as compared to KCM where the multiples are stored.

Method: To find (a x b), first we have to find whether the difference between 'a' and 'b' is odd or even. Based on the difference, the product is calculated using (8) and (9).

i. In case of Even Difference
Result of Multiplication= $[Average]^2 - [Deviation]^2$ **……..** (8)

ii. In case of Odd Difference
Result of Multiplication = [Average x (Average + 1)] - [Deviation x (Deviation+ 1)]**……..** (9)
Where, Average = [(a+b)/2] and Deviation = [Average –smallest (a, b)]

Example 3 (Even difference) and Example 4 (Odd difference) depict the multiplication process. Thus the two variables multiplication is performed by averaging, squaring and subtraction. To find the average [(a+b)/2], which involves division by 2 is performed by right shifting the sum by one bit. If the squares of the numbers are stored in a ROM, the result can be instantaneously calculated. However, in case of Odd difference, the process is different as the average is a floating point number. In order to handle floating point arithmetic, Ekadikena Purvena - the Vedic Sutra which is used to find the square of numbers end with 5 is applied. Example 5 illustrates this. In this case, instead of squaring the average and deviation, [Average x (Average + 1)] - [Deviation x (Deviation+ 1)] is used. However, instead of performing the multiplications, the same ROM is used and using equation (10) the result of multiplication is obtained.
$n(n+l) = (n^2+n)$ **…….** (10)
Here n^2 is obtained from the ROM and is added with the address which is equal to n(n+l). The sample ROM contents are given in Table 4.1.

Table 4.1: Rom Contents

Address	Memory Content (Square)
1	1
2	4
3	9
4	16
…	…

Thus, division and multiplication operations are effectively converted to subtraction and addition operations using Vedic Mathematics. Square of both Average and Deviation is read out simultaneously by using a two port memory to reduce memory access time.

Example 3: 14 x 10 = 140
1) Find the difference between (14-10) = 4 → Even Number
2) For Even Difference, Product = $[Average]^2 - [Deviation]^2$

i. Average = [(a+b)/2] = [(14+10)/2] = [24/2] = 12
ii. Smallest (a,b) = smallest(l4,10) =10
iii. Deviation = Average - Smallest (a,b) = 12 -10 =2
3) Product = $12^2 - 2^2$= 144 - 4 = 140

Example 4: 17 x 14 = 238
1) Find the difference between (17-14) = 3 →Odd Number
2) For Odd Number Difference, Product = [Average x (Average + 1)] - [Deviation x (Deviation+ 1)].
i. Average = [(a+b)/2] = [(17+14)/2] = 15.5
ii. Deviation = [Average – smallest (a, b)] = [15.5 – smallest (l7, 14)] = [15.5 - 14] = 1.5
3) Product = (l5x16) - (lx2) = 240 - 2 =238

Example 5: $25^2 = 625$
1) To find the square of 25, first find the square of 5 which is 25 and put 2 in the tens place and 5 in the ones place of the answer respectively.
2) To find the number in the hundreds place, multiply 2 by its immediate next number, 3, which is equal to (2x3) = 6
3) Answer $25^2 = 625$

5. RESULTS

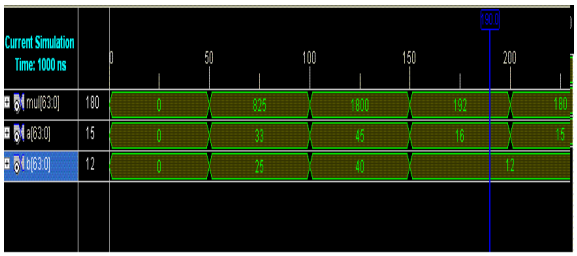

Figure 5.1: 64 bit proposed multiplier

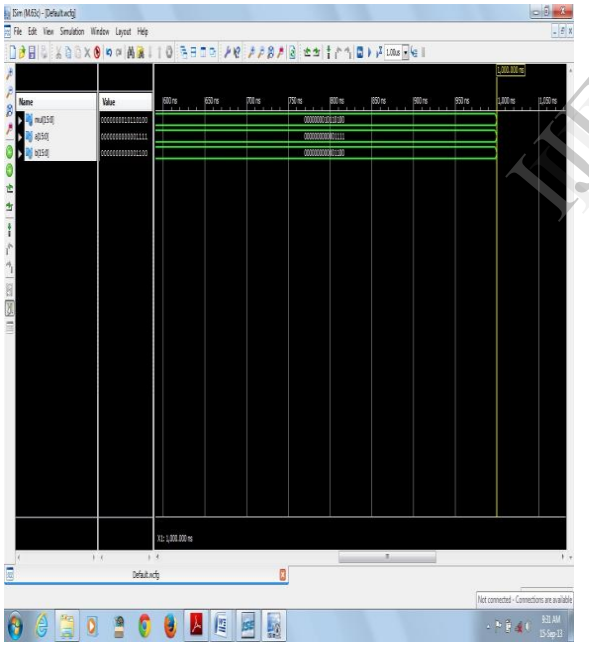

Fig 5.2: 16 bit binary multiplication using proposed method.

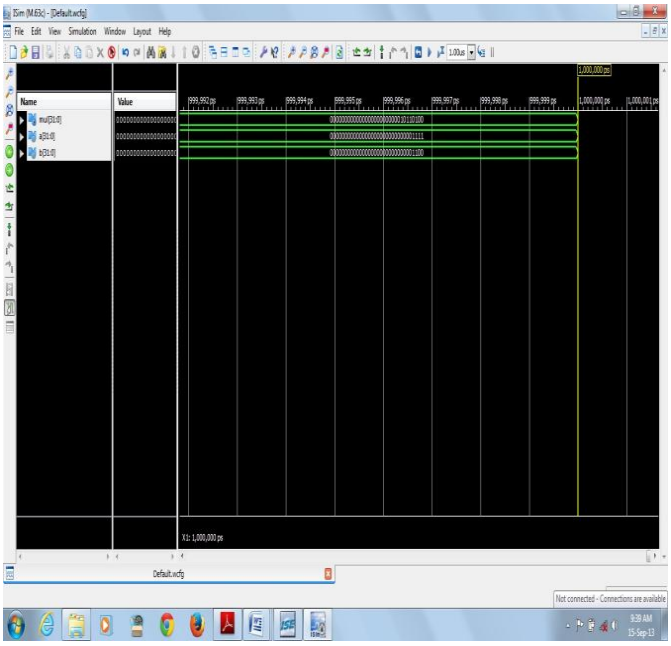

Fig 5.3: 32 bit binary multiplication using proposed method.

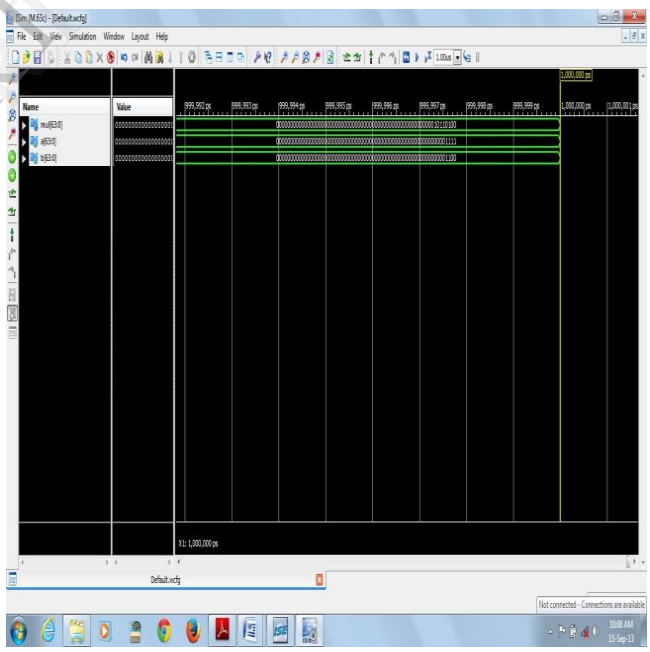

Fig 5.4: 64 bit binary multiplication using proposed method.

SYNTHESIS RESULTS:

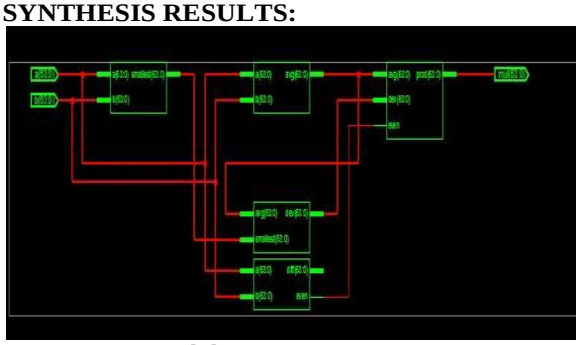

Fig 5.5: Top Module I

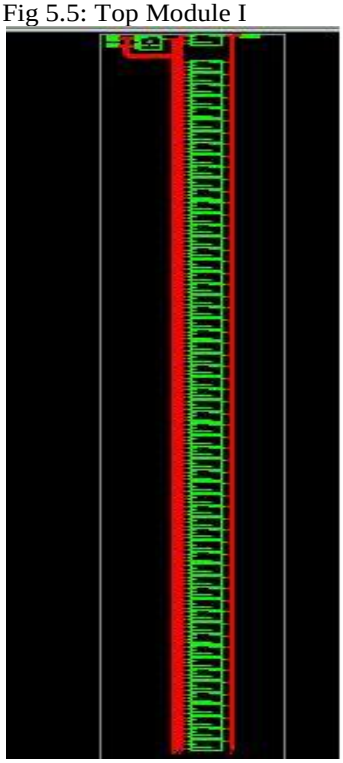

Fig 5.6: Top Module II

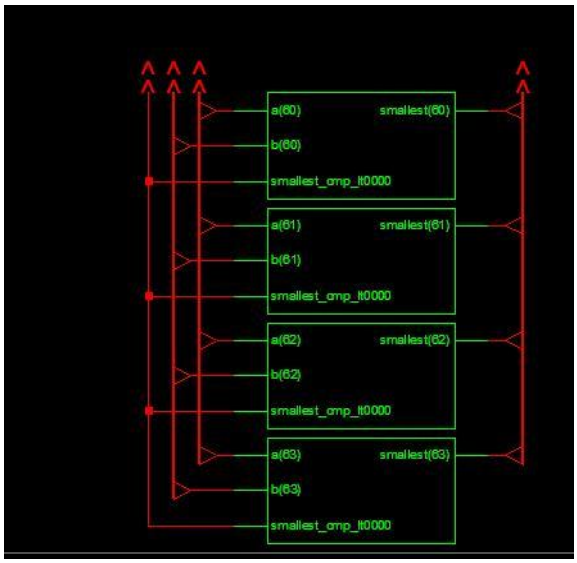

Fig: 5.7 Top Module III

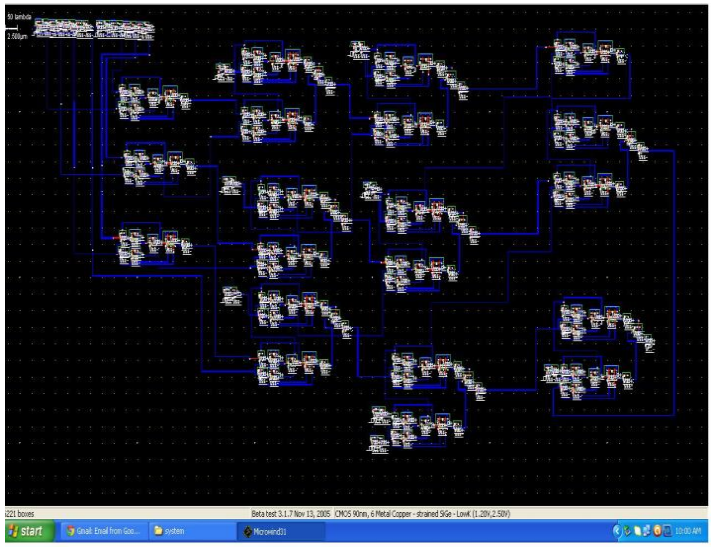

Fig 5.8: Layout

6. CONCLUSIONS

Thus the proposed multiplier provides higher performance for higher order bit multiplication. In the proposed multiplier for higher order bit multiplication i.e. for 16x16 and more, the multiplier is realized by instantiating the lower order bit multipliers like 8x8. This is mainly due to memory constraints. Effective memory implementation and deployment of memory compression algorithms can yield even better results.

Ancient Indian system of mathematics, known as Vedic mathematics, can be applied to various branches of engineering to have a deeper insight into the working of various formulae. The algorithms based on conventional mathematics can be simplified and even optimized by the use of Vedic Sutras. One such possible application of Vedic mathematics to digital signal processing has been discussed. A simple Vedic multiplier architecture based on the Urdhava Tiryagbhyam (Vertically and Cross wise) Sutra of Vedic mathematics has been presented. The hardware architecture of the Vedic multiplier is also depicted and is found to be very similar to that of the so called array multiplier. This is just one of the many possible applications of the Vedic Mathematics to Engineering and some serious efforts are required to fully utilize the potential of this interesting field for the advancement of Engineering and Technology. Although, Vedic mathematics provides many interesting Sutras, but their application to the field of engineering is not yet fully studied. Hence, the vast potential of this interesting field should be exploited to solve the real world problems efficiently.

7. REFERENCES

[1] Swami Bharati Krshna Tirthaji, Vedic Mathematics. Delhi: Motilal Banarsidass Publishers, 1965.
[2] K.K.Parhi "VLSI Digital Signal Processiong Systems -Design and Implementation" John Wiley & Sons, 1999.
[3]Harpreet Singh Dhillon and Abhijit Mitra "A Digital Multiplier Architecture using Urdhava Tiryakbhyam Sutra oj Vedic Mathematics" IEEE Conference Proceedings,200S.
[4] Asmita Haveliya "A Novel Design ./i)r High Speed Multiplier .fi)r Digital Signal Processing Applications (Ancient Indian Vedic mathematics approach)" International Journal of Technology And Engineering System(IJTES):Jan - March 2011- Vo12 .Nol
[5] Raminder Preet Pal Singh, Parveen Kumar, Balwinder Singh "Perfimnance Analysis of'32-Bit Array Multiplier with a Carry Save Adder and with a Carry-Look-Ahead Adder" International Journal of Recent Trends in Engineering, Vol 2, No.6, November 2009
[6] Parth Mehta, Dhanashri Gawali "Conventional versus Vedic mathematical method Jor Hardware implementation oj a multiplier" 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies
[7] Prabir Saha, Arindam Banerjee, Partha Bhattacharyya, Anup Dandapat ""High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics" Proceeding of the 2011 IEEE Students' Technology Symposium 14-16 January, 20 II, liT Kharagpur
[8] H. D. Tiwari, G. Gankhuyag, C. M. Kim, and Y. B. Cho, "Multiplier design based on ancient Indian Vedic Mathematics," in Proceedings IEEE International SoC Design Cotiference, Busan, Nov. 24-25, 200S,pp.65-6S
[9] H. Thapliyal, M. B. Srinivas and H. R. Arabnia , "Design And Analysis oj a VLSI Based High PerJormance Low Power Parallel quare Architecture", in Proc. Int. Conf. Algo. Math. Compo Sc. , Las Vegas, June 2005, pp. 72-76.
[10] P. D. Chidgupkar and M. T. Karad, "The Implementation oj Vedic Algorithms in Digital Signal Processing", Global J. oj /c'ngg. /c'du., vol. 8, no.2, pp. 153-158, 2004.
[11] H. Thapliyal and M. B. Srinivas, "High Speed Efficient N x N Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics", EnJormatika Trans., vol. 2, pp. 225-22S, Dec. 2004.
[12] Wakerly, J.F. "Digital Design-Principles and Practices", 2006, 4th Edition. Pearson Prentice Hall.
[13] J. Bhasker, "Verilog HDL Primer" BS P Publishers, 2003.
[14] Himanshu Thapliyal, S. Kotiyal and M.B. Srinivas, "Design and Analysis (Jf a Novel Parallel Square and Cube Architecture Based on Ancient Indian Vedic Mathematics", Proceedings on 48th II/c'/c'/c' International Midwest Symposium on Circuits and Systems (MWSCAS 2005)
[15] Himanshu Thapliyal and Hamid R. Arabania, "A Time- Area Power Efficient Multiplier and Square Architecture Based on Ancient Indian Vedic Mathematics", proceedings on VLSI04, Las Vegas, U. S. A, June 2004
[16] J M. Ramalatha, K. Deena Dayalan, P. Dharani, S. Deborah Priya, "High Speed Jc'nergy ElJzcient AL U Design using Vedic Multiplication Techniques", ACTEA 2009, Zouk Mosbeh, Lebanon
[17] Pau1.B.C., Fujita.F.S., Okajima.M., "ROM Based Logic (RBL)
Design: A Low-Power 16 Bit Multiplier', IEEE Jouna! of Solid State
Circuits, Volume 44, Issue II, Pg. 2935-2942, Nov 2009.