International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 8, Issue 6 (September 2013), PP.23-28

Design and Implementation of Vedic Multiplier

C.Sheshavali M.Tech[1], K.Niranjan kumar Asst.professor[2]
[1,2]*Department of ECE, PBR VITs, SPSR Nellore (district), JNTUA, Ananthapur, India.*

Abstract:- In this paper, a novel multiplier architecture based on ROM approach using Vedic Mathematics is proposed. This multiplier's architecture is similar to that of a Constant Coefficient Multiplier (KCM). However, for KCM one input is to be fixed, while the proposed multiplier can multiply two variables. The proposed multiplier is implemented on a Cyclone III FPGA, compared with Array Multiplier and Urdhava Multiplier for both 8 bit and 16 bit cases and the results are presented. The proposed multiplier is 1.5 times faster than the other multipliers for 16x16 case and consumes only 76% area for 8x8 multiplier and 42% area for 16x16 multiplier.

Keywords:- KCM; Urdhava; Vedic Maths; Array Multiplier; FPGA.

I. INTRODUCTION

Multiplication is one of the more silicon-intensive functions, especially when implemented in Programmable Logic. Multipliers are key components of many high performance systems such as FIR filters, Microprocessors, Digital Signal Processors, etc. A system's performance is generally determined by the performance of the multiplier, because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue.

Vedic mathematics [I] is the ancient Indian system of mathematics which mainly deals with Vedic mathematical formulae and their application to various branches of mathematics. The word 'Vedic' is derived from the word 'Veda' which means the store-house of all knowledge. Vedic mathematics was reconstructed from the ancient Indian scriptures (Vedas) by Sri Bharati Krshna Tirthaji (1884-1960), after his eight years of research on Vedas [1]. According to his research, Vedic mathematics is mainly based on sixteen principles or word-formulae which are termed as Sutras. This is a very interesting field and presents some effective algorithms which can be applied to various branches of Engineering such as Computing and Digital Signal Processing.

II. 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. But it is also the slowest with the latency proportional to O (Wct), where Wd is the word length of the operand. Example I describes the multiplication process using array multiplier and Fig.l depicts the structure of the same. Instead of Ripple Carry Adder (RCA), here Carry Save Adder (CSA) is used for adding each group of partial product terms, because RCA is the slowest adder among all other types of adders available. In case of multiplier with CSA [5], partial product addition is carried out in Carry save form and RCA is used only in final addition.

Example 1: (1101 x 1110) = 1 0 1 1 0 1 1 0

```
1 1 0 1
1 1 1 0 X
-----------------------------------
          0 0 0 0
        1 1 0 1     --- Left Shift by I bit
      1 1 0 1       --- Left Shift by 2 bit
    1 1 0 1         --- Left Shift by 3 bit
-----------------------------------------
  1 0 1 1 0 1 1 0
```

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.

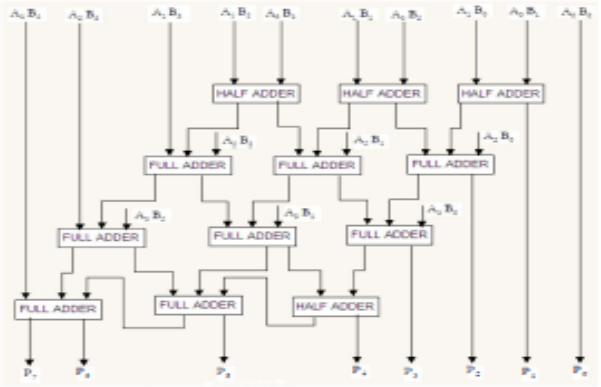

Fig. 1: Array Multiplier using CSA Hardware Architecture

III. URDHAVA MULTIPLIER

Urdhava Tiryakbhyam [1] [3] (Vertically and Crosswise), is one of Sixteen Vedic Sutras and deals with the multiplication of numbers. The sutra is illustrated in Example 2 and the hardware architecture is depicted in Fig.3. In this example two decimal numbers (31 x 35) are multiplied. Line diagram for the multiplication of two, three and four digit numbers is shown in Fig. 2 using Urdhava Method. The digits on the two ends of the line are multiplied and the result is added with the previous carry. When three or more lines are present, all the results are added to the previous carry. The least significant digit of the number thus obtained acts as one of the result digit and the rest act as the carry for the next step. Initially the carry is taken to be zero.

Example 2: 40x45 =1800

4 0 0 4 0 4

4 5X 5 X 4 5 4X

0	20 + 0 = 20	16+2=18

Carry to next stage

Answer: 40x45 =1800

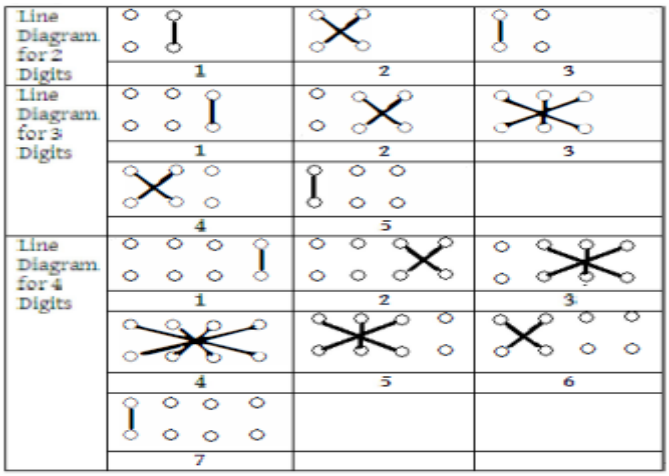

Fig. 2: Line Diagram for Urdhava Multiplication of 2, 3 and 4 digits

From the Example 2, it is observed that all the partial products are generated in parallel. So the speed of the multiplier is higher compared to array multiplier.

The above discussions can now be extended to multiplication of binary number system with the preliminary knowledge that the multiplication of two bits a_0 and b_0 is just an AND operation and can be implemented using simple AND gate. To illustrate this multiplication scheme in binary number system, consider the multiplication of two binary numbers $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$. As the result of this multiplication would be more than 4 bits, the product is expressed as r7r6r5r4r3r2rlr0. Least significant bit r_0 is obtained by multiplying the

least significant bits of the multiplicand and the multiplier as shown in the Fig.2. The digits on 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 (r_0) and a carry (C_n). This carry is added in the next step and thus 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 the other entire bits act as carry.

For example, if in some intermediate step, we get 110, then 0 will act as result bit and 11 as the carry (referred to as Cn in this text). It should be clearly noted that C n may be a multi-bit number. Thus the following expressions (1) to (7) are derived:

$r_0 = a_0b_0$ ... (1)

$c_1r_1 = a_1b_0 + a_ob_1$ ... (2)

$c_2r_2 = c_1 + a_2b_0 + a_1b_1 + a_ob_2$ ... (3)

$c_3r_3 = c_2 + a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3$ ... (4)

$c_4r_4 = c_3 + a_3b_1 + a_2b_2 + a_1b_3$ ... (5)

$c_5r_5 = c_4 + a_3b_2 + a_2b_3$ ... (6)

$c_6r_6 = c_5 + a_3b_3$ ... (7)

with $c_6r_6r_5r_4r_3r_2r_1r_O$ being the final product. Partial products are calculated in parallel and hence the delay involved is just the time it takes for the signal to propagate through the gates.

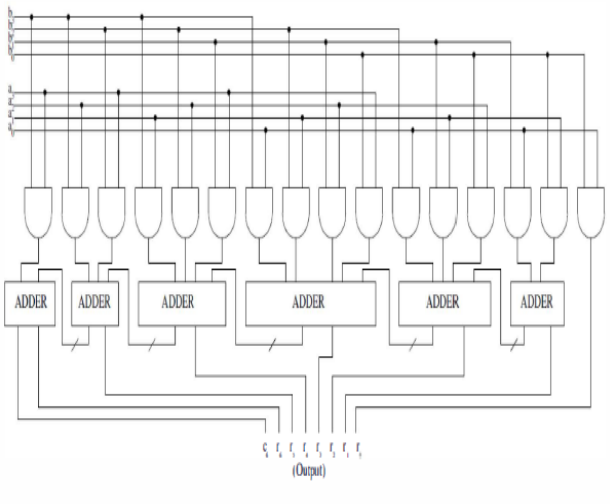

Fig.3 Urdhava Multiplier Hardware Architecture

The main advantage of the Vedic Multiplication algorithm (Urdhava Tiryakbhyam Sutra) stems from the fact that it can be easily implemented in FPGA due to its simplicity and regularity [3]. The digital hardware realization of a 4-bit multiplier using this Sutra is shown in Fig. 3. This hardware design is very similar to that of the array multiplier where an array of adders is required to arrive at the final product. Here in Urdhava, all the partial products are calculated in parallel and the delay associated is mainly the time taken by the carry to propagate through the adders.

IV. 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= $[\text{Average}]^2 - [\text{Deviation}]^2$... (8)

II. In case of Odd Difference

Result of Multiplication = [Average x (Average + 1)]-[Deviation x (Deviation+ I)] ... (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 variable 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+ I)] 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 1. TABLE 1: ROM CONTENTS Address Memory Content (Square)

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 Maths. Square of both Average and Deviation is read out simultaneously by using a two port memory to reduce memory access time.

Example 3: 18 x 14=252
I. Find the difference between (18-14) = 4 → Even Number
II. For Even Difference, Product = $[Average]^2$ - $[Deviation]^2$
i. Average = [(a+b)/2] = [(18+14)/2] = [32/2] = 16
ii. smallest(a, b) = smallest(l8,14) =14
iii. Deviation = Average - Smallest (a,b) = 16 -14 =2
III. Product = 16^2-2^2 = 256 - 4 = 252

Example 4: 16 x 13 = 208
I. Find the difference between (16-13) =3 →Odd Number
II. For Odd Number Difference find the Average and Deviation.
i. Average = [(a+b)/2] = [(16+13)/2] = 14.5
ii. Deviation=[Average - smallest(a,b)]= [14.5 - smallest(l6,13)] = [14.5 - 13] = 1.5
III. Product = (l4xI5) - (lx2) = 210 - 2 =208

Example 5: 25^2=625
I. 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.
II. To find the number in the hundreds place, multiply 2 by its immediate next number, 3, which is equal to (2x3) = 6
III. Answer 25^2=625

Fig.4 depicts the RTL view of the proposed multiplier for 4x4 as a sample case, implemented on a Cyclone II device. 8x8 multiplier is implemented using ROM approach, by storing the squares of the numbers in the memory starting from 0000 0000 to 1111 1111. The memory requirement for an 8x8 multiplication will be 8KB. But in the case of 16xl6 multiplier the memory requirement will be huge, 2^{16}x32=2MB. So, in order to reduce the memory requirements for higher order bit multiplication, (l6x16, 32x32, etc.) lower order (8x8) multiplier can be instantiated[1 7]. By this process the constraint of larger memory requirements can be overcome.

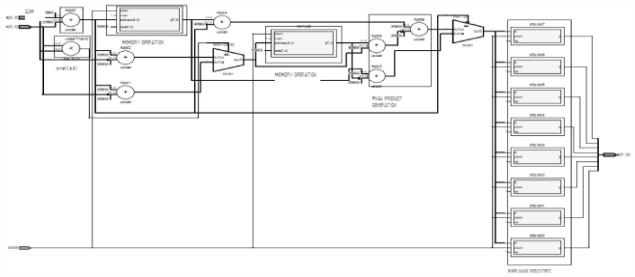

Fig. 4: RTL View of Proposed Multiplier (4x4)

V. EXPERIMENTAL RESULTS

From the Table 2 and Table 3, it is inferred that the proposed multiplier is best suited for the applications where the less area requires and speed is major considerations. This is achieved due to the feature of multiplier that will consume only fewer logic elements for its implementation.

	Array Multiplier	Urdhava Multiplier	Proposed Multiplier
16x16 Multiplier	510	810	145
8x8 Multiplier	126	180	311

Table: 2 Requirements of combinational logic functions

Array Multiplier	Urdhava Multiplier	Proposed Multiplier
61.277	50.952	23.87

Table: 3 Time delay in nanoseconds for 16x16 Multipliers

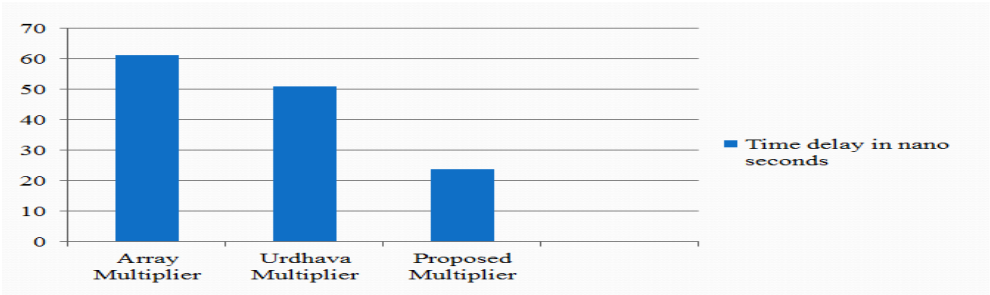

Fig:5 For 16x16 Multipliers it will shows the time delay comparison

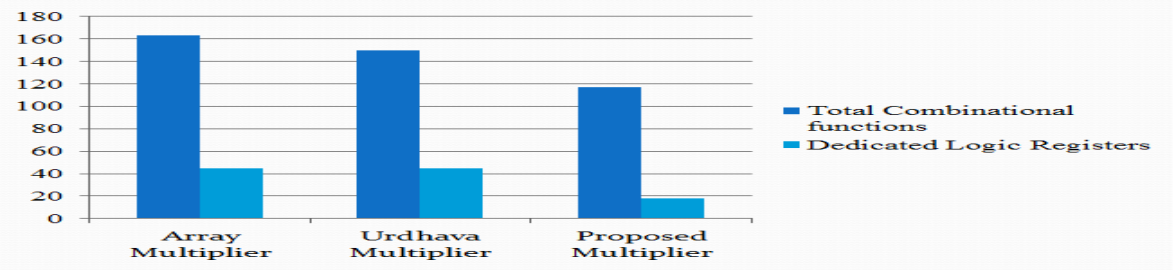

Fig:6 For 16x16 Multipliers Area comparison

From the observation of simulation results for 8x8 and 16x16 multipliers in the case of proposed multipliers it is clear that it is more efficient and comfortable for higher order multipliers i.e, greater than 8x8 multipliers

VI. CONCLUSION

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.

REFERENCES

[1]. Swami Bharati Krishna Tirthaji, Vedic Mathematics. Delhi: Motilal Banarsidass Publishers, 1965.
[2]. K.K.Parhi "VLSI Digital Signal Processing 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, 2008.
[4]. Asmita Haveliya "A Novel Design for High Speed Multiplier .for 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 for Hardware implementation of 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 Conference, 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 of 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),