Fast and Power Efficient 16×16 Array of Array Multiplier using Vedic Multiplication

Dr. K.S. Gurumurthy [a, b], M.S Prahalad [a, c]
[a] DOS in Electronics and Communication Engineering, UVCE Bangalore- 560 001, India
[b] Professor, drksgurumurthy@gmail.com
[c] PG student, msprahalad@live.com

***Abstract*- This paper discusses about "Array of Array" multiplier which is a derivative of Braun Array Multiplier. Braun array are much suitable for VLSI implementation because of its less space complexity though it shows larger time complexity, on the other hand tree multipliers have time complexity of O(log n) but are less suitable for VLSI implementation since, being less regular; they require larger total routing length, which leads to performance degradation; simply put, they show higher space complexity. The main advantage of "Array of Array" multipliers is its inherent ability to reduce both time and space complexity [7][8] with intermediate relative performance [7]. In this paper a 16×16 unsigned 'Array of Array' multiplier circuit is designed with hierarchical structuring, it has been optimized using Vedic Multiplication Sutra (Algorithm) "Urdhva Triyagbhyam" [1][6] and Karatsuba-Ofman algorithm[2]. The proposed algorithm is useful for math coprocessors in the field of computers. Algorithm is implemented on SPARTAN-3E FPGA (Field Programmable Gate Array). The proposed multiplier implementation shows large reduction in average power dissipation and in time delay as compared to Booth encoded radix-4 multiplier.**

***Keywords*-Vedic Mathematics, Urdhva Triyakbhyam Sutra, Karatsuba - Ofman algorithm, Booth encoded radix-4 multiplier, Braun array.**

I. INTRODUCTION

The single core processor is heart and brain of a computer, enhancing the ability of single core to handle the complex and challenging processes has led to design of MP (Multi Core Processor). The load on the processor is reduced by supplementing the main processor with Co-Processors. Co-processors design is application specific, they are used as add on with the main processor in field of applications like Signal Processing, Image Processing, Servers, Graphics etc.. Most DSP tasks require real-time processing; it must perform these tasks speedily while minimizing Cost and Power. Although multiplication techniques such as 'Booth Algorithm Multiplier' have been effective over conventional 'Array Multiplication' technique, their disadvantage of time consumption has not been completely removed. The multiplication algorithms differ in the means of 'partial product generation' and 'partial product addition [16]. The array multiplier has linear time complexity i.e. O(n) therefore delay degrades the performance for multipliers having larger operand sizes. Also it has poor space complexity $O(n^2)$, as it requires approximately n^2 cells to produce multiplication. Therefore as the operand size grows, the circuit takes larger area and power [8]. Vedic Mathematics shows its application in fast calculations, trigonometry, three dimensional coordinate geometry, solution of plane and spherical triangles, linear and non-linear differential equations, matrices and determinants, log and exponential. Vedic Mathematics provides unique solutions in several instances where trial and error method is available at present [1]. In this paper we present implementation of 16×16, multiplier design using Array of Array technique. The VLSI implementation of multiplier circuit is done using VHDL. Simulation results are compared with Booth encoded radix 4 multiplier [3]. Section II gives detailed explanation on Vedic multiplication sutra and optimization of the sutra using Karatsuba - Ofman algorithm. Section III explains the design of functional blocks of the Array of Array Multiplier. Physical implementation and results are described in Section IV. Chapter V concludes the paper.

II. URDHVA TRIYAGBHYAM AND KARATSUBA-OFMAN ALGORITHMS

A. Urdhva Triyagbhyam

Urdhva Triyagbhyam is a multiplication Sutra (Algorithm) in Vedic Mathematics. This is the general formula applicable to all cases of multiplication [1]. Urdhva means Vertical and Triyagbhyam means Crosswise.

Illustration:

10 * 11

1 0

1 1

1; 1+0; 0 = 110 (**result**)

Steps:

i. Multiply LSBs (Least Significant Bits) '0' and '1'; place product '0' as LSB of the result: **Vertical Multiplication**

ii. Multiply Crosswise '1' and '1'; '0' and '1'; add the product terms and place obtained sum as middle term of the result: **Crosswise Multiplication**

iii. Multiply the MSBs (Most Significant Bits) '1' and '1'; place product '1' as MSB of the result: **Vertical Multiplication**

B. Karatsuba – Ofman Algorithm

The Karatsuba- Ofman algorithm is a formula that allows us to compute the product of two large numbers x and y using three multiplications of smaller numbers, each with about half as many digits as x or y, plus some additions and digit shifts. Let x and y be represented as n-digit strings in some base B. For any positive integer m less than n, one can split the two given numbers as follows

$$x = x_1B^m + x_0 \quad \text{(Higher term + Lower Term)}$$
$$y = y_1B^m + y_0$$

Where, x_0 and y_0 are less than B^m. The product is then

$$xy = (x_1B^m + x_0)(y_1B^m + y_0)$$
$$= z_2\,B^{2m} + z_1\,B^m + z_0$$

Where, $z_2 = x_1y_1$; $z_1 = x_1y_0 + x_0y_1$; $z_0 = x_0y_0$

A close observation reveals that algorithm is pretty much similar to Urdhva Triyagbhyam Sutra. Equal merit is given to both algorithms as they have been developed independently.

III. DESIGN OF THE 16X16 MULTIPLIER

*A. The Fundamental Block (*2×2 block*)*

In the design of the proposed Array of Array multiplier a 2×2 block is a *fundamental block* (Basic block).The truth table for a 2×2 combinational multiplier is shown in table I. The truth table can be solved using K- map, which generates the equations (1). A 2×2 block, combinational circuit can be realized using these equations.

$$\left.\begin{aligned} P_0 &= (A_0 \textit{ and } B_0)\\ P_1 &= (A_0 \textit{ and } B_1) \quad \textit{xor } (B_0 \textit{ and } A_1)\\ P_2 &= (A_1 \textit{and } B_1) \textit{ and } \overline{(A_0 \textit{ and } B_0)}\\ P_3 &= (A_1 \textit{and } B_1) \end{aligned}\right\} \quad \ldots\ldots (1)$$

TABLE I- TRUTH TABLE OF 2×2MULTIPLIER

A_1	A_0	B_1	B_0	P_3	P_2	P_1	P_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

B. Design of 4×4 *block*

The design of 4×4 block is a simple arrangement of 2×2 blocks in an effective and realizable manner. The first step in the design of 4×4 block will be finding the different combinations of input bit pairs that are derived in terms of 2×2 block. Each input bit-pair is handled by a separate 2×2 combinational multiplier to produce four partial product rows. These partial products rows are then added optimally to generate final product bits. The design procedure for 4×4 block is shown in table II, while figure 1 shows the schematic of a 4×4 block designed using 2×2 blocks. These partial products rows are then added optimally using 4-bit full adder cells. The partial products represent the Urdhva vertical and cross product terms. The 8×8 block is structurally similar to a 4×4 block with nibble (4 bits) inputs. The structure uses four 4×4 blocks and the orientation is similar to 4×4 block.

TABLE II- TABLE OF VERTICAL AND CROSS PRODUCT TERMS

Product terms	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7		
Vertical	A_0 B_0	A_1 B_1	A_2 B_2	A_3 B_3						
Cross	A_1 B_0	A_0 B_1	A_2 B_1	B_2 A_1	A_3 B_2	B_3 A_2	A_2 B_0	B_2 A_0	A_3 B_0	A_0 B_3

$P_0 = A_0B_0$

$P_1 = A_1B_0 + A_0B_1$

$P_2 = B_2A_0 + A_2B_0 + A_1B_1 + C_1(1)$

$P_3 = B_3A_0 + B_0A_3 + B_2A_1 + B_1A_2 + C_2(2 \text{ to } 1)$

$P_4 = B_3A_1 + B_2A_2 + B_1A_3 + C_3(2 \text{ to } 1)$

$P_5 = B_3A_2 + B_2A_3 + C_4(2 \text{ to } 1)$

$P_6 = B_3A_3 + C_5(2 \text{ to } 1)$

$P_7 = C_6(1)$

Product = P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0 (Concatenated)

Where, $C_1(1)$, $C_2(2 \text{ to } 1)$, $C_3(2 \text{ to } 1)$, $C_4(2 \text{ to } 1)$, $C_5(2 \text{ to } 1)$, $C_6(1)$ represent the carries generated in the previous stage. Now let us define the meaning of cross product and vertical product in case of 4×4 multiplier

Let us consider the multiplicand and multiplier,

$X = A_3A_2\ A_1A_0$

$Y = B_3B_2\ B_1B_0$

The cross product is defined as,

$PP_1 = (A_3A_2\ B_1B_0)\ X\ (B_3B_2\ A_1A_0)$

The vertical product is defined as,

$PP_2 = (A_3A_2\ B_3B_2)\ X\ (B_1B_0\ A_1A_0)$

Where, PP_1, PP_2 represents the partial output products. These two partial products are logically and optimally added using a 4-bit Adder as shown in the figure 1.

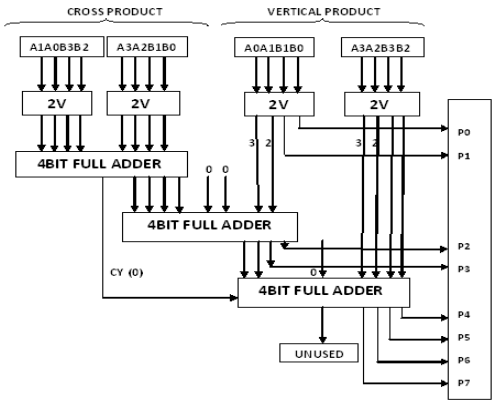

Figure 1- Structure of 4×4 block

C. *DESIGN OF A* 16×16 *MULTIPLIER*

The structural characteristics of the 16×16 are inherited from 4×4 structure. Blocks of 8 bits (byte) is considered as the input and Urdhva Triyagbhyam multiplication sutra is applied as discussed in the previous sub-section B. The first step in design will be finding the different combinations of input bytes that generate vertical and cross products and are derived in terms of 8×8 block. Each input bit-pair is handled by a separate 4×4 block which has been already designed using 2X2 multiplier as explained in section III B. These separate 8×8 block produce four partial product rows. These partial products rows are then added optimally to generate final product bits of 16×16 multiplier as shown in figure 2.

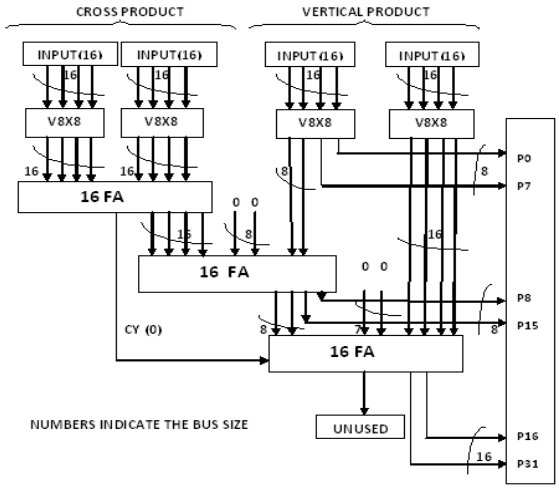

Figure 2- Structure of 16×16 Multiplier

IV. PHYSICAL IMPLEMENTATION

The Booth Radix-4 and proposed "Array of Array" multiplier (discussed in section III) are simulated using Modelsim VHDL [13][15] simulator. The synthesis and implementation is done by using Xilinx ISE [14]. The algorithms are implemented on Xilinx FPGA SPARTAN-3E device xc3s100e-5vq100. The efficient Booth radix-4 multiplier VHDL simulation waveforms are as shown in the figure 3 and found to be correct. The multiplier is designed for two 8 bit numbers. The generated result of placement and routing for the selected FPGA when Booth Multiplier was implemented is shown in figure 4. One can easily see from the placement and routing result in the figure 4 that, in spite of its efficiency, this multiplier consumes space and hardware.

The proposed Array of Array multiplier VHDL simulation waveforms are as shown in the figure 5. Figure 6 is the generated result of placement and routing for the selected FPGA when proposed "Array of Array" Multiplier is implemented. It is clear from figures 4 and 6 that the proposed multiplier uses less resources are utilized from FPGA as compared to Booth Radix 4 multiplier.

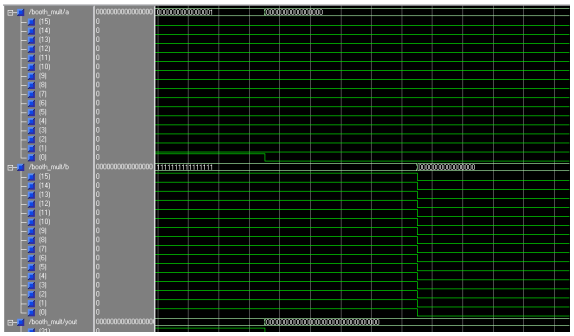

Figure-3 VHDL Simulation Results of Booth Radix-4 Multiplier

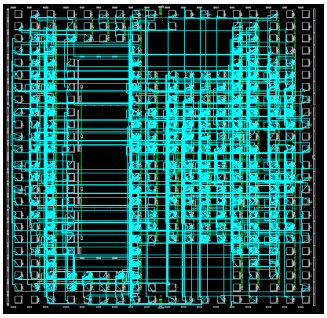

Figure 4- Placement and Routing Results of Booth Radix-4 Multiplier

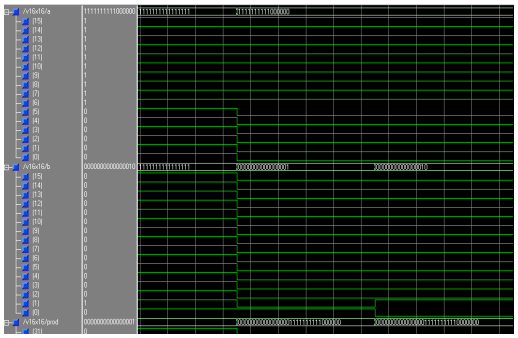

Figure-5 VHDL Simulation Results of proposed Array of Array Multiplier

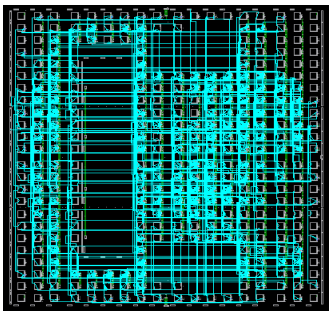

Figure 6- Placement and Routing Results of proposed Array of Array Multiplier

V. CONCLUSION

TABLE II- COMPARISION OF SYNTHESIS RESULTS

Algorithm	**Proposed**	**Booth Radix4**
Delay(nS)	37.668 (22.635 logic, 15.033 route)	46.740 (28.351 logic, 18.389 route)
Power Dissipation (mW)	29.34	151.34
Slices	404 (out of 960) (42%)	575 (out of 960) (59%)
4 Input Luts	716 (out of 1920) (37%)	900 (out of 1920) (46%)
Total Equivalent Gate Count For Design	4299	7296

Table II indicates that the delay in multiplication is reduced by 9.072nS in comparison with Booth Radix-4, this is primarily due to the hierarchical structuring and effective addition of the partial product terms. Further, the proposed multiplier is space efficient too as compared to Booth multiplier implementation. The proposed multiplier dissipates approximately 5 times lesser power than Booth Radix-4 multiplier. Due to factors such as timing efficiency, speed and lesser area and low power dissipation the proposed Multiplier can be implemented in Arithmetic and Logical Units replacing traditional multipliers. The speed, power and area improvements are gained due to the hierarchical structuring and effective addition of the partial product terms and such a design must enable large saving of resources when used in FPGA or ASIC for digital signal processing, General purpose processing, image processing and any other application.

REFERENCES

[1] Jagadguru Swami, Sri Bharati Krsna Tirthji Maharaja," Vedic Mathematics", Motilal Banarsidas, Varanasi, India, 1986.

[2] A. Karatsuba, Yu. Ofman(1962). "Multiplication of Many-Digital Numbers by Automatic Computers". *Proceedings of the USSR Academy of Sciences* 145: 293–294.

[3]Mrs. M. Ramalatha, Prof. D. Sridharan, "VLSI Based High Speed Karatsuba Multiplier for Cryptographic Applications Using Vedic Mathematics", IJSCI, 2007

[4] Thapliyal H. and Srinivas M.B. "High Speed Efficient N x N Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics", Transactions on Engineering, Computing and Technology, 2004, Vol.2.

[5] Prashant Nair, Darshan Paranji, S. S. Rathod," VLSI Implementation of Matrix-Diagonal Method of Binary Multiplication". Proceedings of SPIT-IEEE, Vol. 2, 55, 2007

[6] M. Ramalatha, K. Deena Dayalan, P. Dharani, S. Deborah Priya," High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques ", ACTEA 2009

[7] Abhijit Asati and Chandrashekhar "A High-Speed, Hierarchical 16×16 Array of Array Multiplier Design", IMPACT 2009.

[8] Kevin Biswas, "Multiplexer Based Array Multipliers," A Ph.D. Dissertation, University of Windsor, Electrical and Computer Engineering, Apr. 2005.

[9] Himanshu Thapliyal and Hamid R. Arabnia, "A time area power efficient multiplier and square architecture based on ancient Indian Vedic mathematics," www.vedicmathsindia.org.

[10] Vishal Verma and Himanshu Thapliyal , "High Speed Efficient N X N Bit Multiplier Based On Ancient Indian Vedic Mathematics", Proceedings International Conference On VLSI, Las Vegas, June 2003.

[11] A.P. Nicholas, K.R Williams, J. Pickles, "Application of Urdhava Sutra", Spiritual Study Group, Roorkee (India), 1984

[12] Stephen Brown, Zvonko Vranesic, "Fundamentals of digital logic with VHDL design," 1st Ed. New York: McGraw-Hill, 2007

[13] "Active-HDL 7.2User Manual", Active HDL Inc, USA, 2007.

[14] "Xilinx ISE 9.2i User Manual", Xilinx Inc, USA, 2007

[15] "VHDL Reference Manual", IEEE standard, 1993