RESEARCH ARTICLE OPEN ACCESS

A Novel High Performance Implementation of 64 Bit MAC Units and Their Delay Comparison

Kandimalla Rajaneesh, M.Bharathi
M.Tech (VLSI) Student, Sree Vidyanikethan Engineering College (Autonomous), A. Rangampet, Tirupati, India.
Assistant Professor, Sree Vidyanikethan Engineering College (Autonomous), A. Rangampet, Tirupati, India.

Abstract
A novel high performance 64 bit Multiplier-and-Accumulator (MAC) is implemented in this paper.MAC plays a vital role in most of the digital signal processing (DSP).The MAC unit is designed using vedic,braun,dadda multiplier and carry save adder hence, compared with performance of MAC unit using Wallace multiplier and carry save adder.In gate level Verilog hdl used for coding digital circuits using tool Xilinx ISE 10.1i and target family Spartan 3E,Device- XC3S500,speed -5,package: FG320.The synthesized for the proposed digital circuits.
Keywords-Carry save adder,Digital signal processing (DSP),multiplier and accumulator (MAC),Verilog-HDL.

I. INTRODUCTION

A MAC unit consists of a multiplier and an accumulator containing the sum of the previous successive products. The MAC inputs are obtained from the memory location and given to the multiplier block. 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. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, a whole spectrum of multipliers with different area-speed constraints have been designed with fully parallel.Hence,with the suitable choice of the type of the multiplier the performance of the MAC unit can be made better.

Hence, in the construction of the MAC unit multiplier plays a vital role so, we have selected those multipliers which exhibit better performance than the previous one implemented in the MAC unit like Vedic Braun, Dadda multiplier and compared the performance with Wallace multiplier which was implemented before in the base paper.

This paper is divided into eight sections. In the first section the introduction is discussed. In the second section discuss about MAC operation. The third section details about vedic multiplier is done and fourth section deals with the operation of Dadda multiplier and In the fifth section describes bruan multiplier hence, carry save adder is explained in sixth section and finally obtained results is discussed in seventh and the conclusion is made in the eight section.

II. MAC OPERATION

The Multiplier-Accumulator (MAC) operation is the key operation not only in DSP applications but also in multimedia information processing and various other applications. As we know, MAC unit mainly consist of multiplier, adder and accumulator. In this paper, we proposed a MAC unit consisting of adder and accumulator in a same block. By this proposed method by building both adder and the accumulator in the same block the delay can be decreased and other better performance can be seen. This will be useful in 64 bit digital signal processor. The input which is being fed from the memory location is 64 bit. When the input is given to the multiplier it starts computing value for the given 64 bit input and hence the output will be 128 bits. The multiplier output is given as the input to adder and accumulator block shown below in figure 1.

The function of the MAC unit is given by the following equation [1]:

$$F=\sum Pi\ Qi \quad (1)$$

The output adder and accumulator block is 129 bit i.e. one bit is for the carry (128bits+ 1 bit). Then, the output is fed back to the same adder and accumulator block. The figure 1 shows the new architecture of MAC unit.

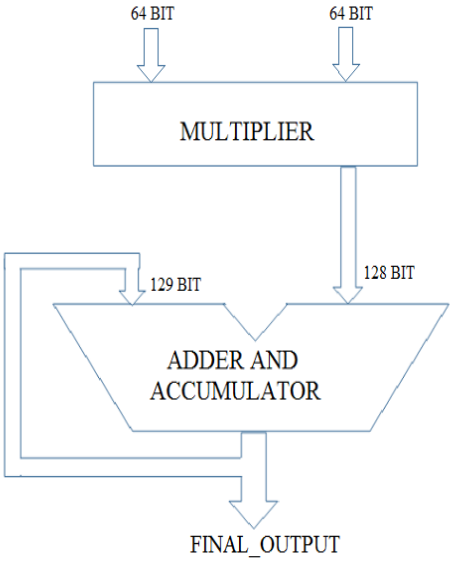


Figure 1 : New architecture of MAC unit.

III. VEDIC MULTIPLIER

The hardware architecture of 2X2, 4x4 and 8x8 bit Vedic multiplier module are displayed in the below sections. Here, "Urdhva-Tiryagbhyam" (Vertically and Crosswise) sutra is used to propose such architecture for the multiplication of two binary numbers. The beauty of Vedic multiplier is that here partial product generation and additions are done concurrently. Hence, it is well adapted to parallel processing. The feature makes it more attractive for binary multiplications. This in turn reduces delay, which is the primary motivation behind this work.

A. Vedic Multiplier for 2x2 bit Module

The method is explained below for two, 2 bit numbers *A* and *B* where *A* = *a*1*a*0 and *B* = *b*1*b*0 as shown in Fig. 1. Firstly, the least significant bits are multiplied which gives the least significant bit of the final product (vertical). Then, the LSB of the multiplicand is multiplied with the next higher bit of the multiplier and added with, the product of LSB of multiplier and next higher bit of the multiplicand (crosswise). The sum gives second bit of the final product and the carry is added with the partial product obtained by multiplying the most significant bits to give the sum and carry. The sum is the third corresponding bit and carry becomes the fourth bit Of the final product[10].

Figure 1 shows the block diagram of 2x2 bit Vedic Multiplier which is further used for the implementation of the 4x4 bit vedic multiplier and further 8x8 vedic multiplier is implemented.

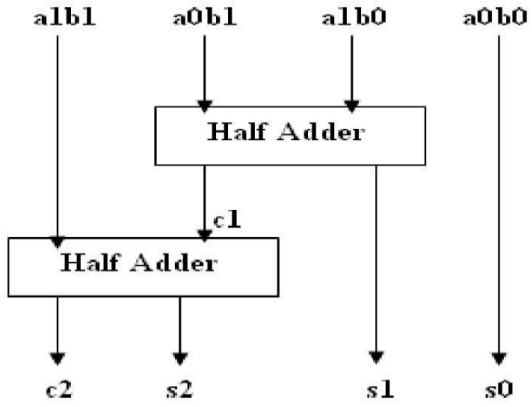


Figure 2[10]:Block Diagram of 2x2 bit Vedic Multiplier.

B. Vedic Multiplier for 4x4 bit Module

The proposed Vedic multiplier can be used to reduce delay. Early literature speaks about Vedic multipliers based on array multiplier structures. On the other hand, we proposed a new architecture, which is efficient in terms of speed. The arrangements of RC Adders shown in Fig. 2, helps us to reduce delay. Interestingly, 8x8 Vedic multiplier modules are implemented easily by using four 4x4 multiplier modules

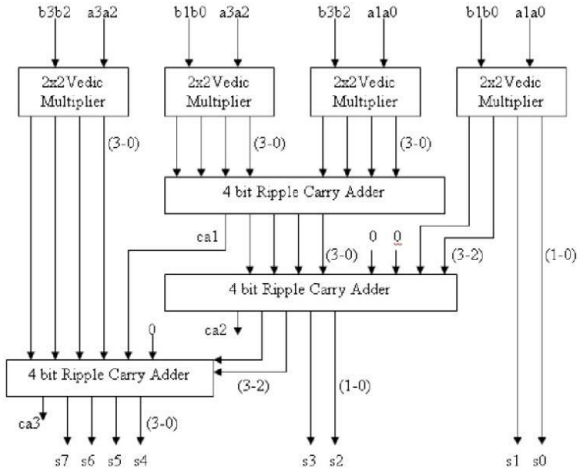


Figure 3[10]: Block Diagram of 4x4 bit Vedic Multiplier.

C. Vedic Multiplier for 8x8 bit Module

The 8x8 bit Vedic multiplier module as shown in the block diagram in Fig. 3 can be easily implemented by using four 4x4 bit Vedic multiplier modules as discussed in the previous section Let□ s analyze 8x8 multiplications, say A= A7 A6 A5 A4 A3 A2 A1 A0 and B= B7 B6 B5B4 B3 B2 B1B0. The output line for the multiplication result will be of 16 bits as – S15 S14 S13 S12 S11 S10 S9 S8 S7

S6bS5S4 S3 S2 S1 S0. Let□ s divide A and B into two parts, say the 8 bit multiplicand A can be decomposed into pair of 4 bits AH-AL. Similarly multiplicand B can be decomposed into BH-BL. The 16 bit product can be written as: Using the fundamental of Vedic multiplication, taking four bits at a time and using 4 bit multiplier block as discussed we can perform the multiplication. The outputs of 4x4 bit multipliers are added accordingly to obtain the final product. Here total three 8 bit Ripple-Carry Adders are required as shown in Fig.3.

Since the 64 bit vedic multiplier is difficult to represent, a typical 8-bit block diagram is shown in figure 4 for understanding.

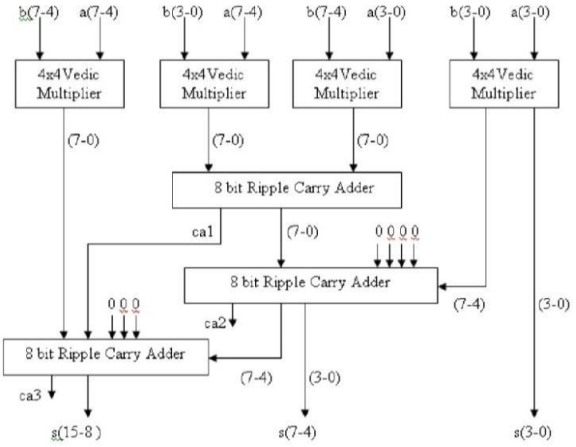


Figure 4[10]: Block Diagram of 8x8 bit Vedic Multiplier.

IV. DADDA MULTIPLIER

Dadda multipliers are the refinement of parallel multipliers first presented by Wallace in 1964. In contrast to the Wallace reduction Dadda multiplier perform the least reduction at each stage[2]. The maximum height of each stage is determined by working back from final stage which consists of two rows of partial products. The height of each stage should be in the order 2,3,4,6,9,13,19,28,42,63 etc. An 8 bit Dadda multiplier reduction is shown in figure 5.

For Dadda multipliers the number of full adders and half adders required depends on the value of N.

No of Full Adders = $\mathbf{N^2 - 4N+3}$
No of Half Adders = **N-1**

Since the 64 bit dadda multiplier is difficult to represent, a typical 8-bit by 8-bit reduction shown in figure 5 for understanding.

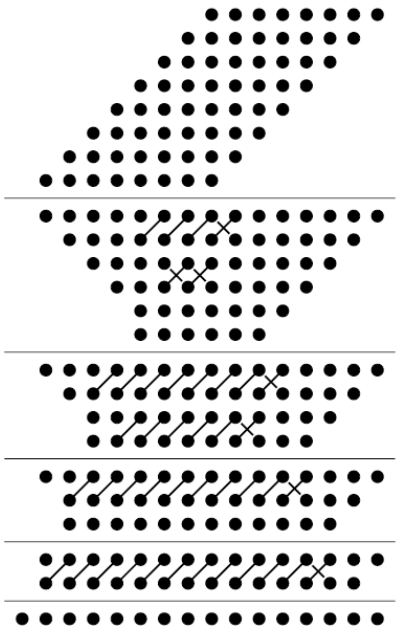


Figure 5[2]: Dadda Multiplier for N=8

V. BRAUN'S MULTIPLIER

Braun's multiplier is an n × m bit parallel multiplier and generally known as carry save multiplier and is constructed with m × (n-1) address and m × n AND gates. The Braun's multiplier has a glitching problem which is due to the ripple carry adder in the last stage of the multiplier.

A. Mathematical Basics

Consider a generic m by n multiplication of two unsigned n-bit numbers Y = Ym-1 …. Y0 and X = Xn-1…X0

$$Y = \sum_{i=0}^{m-1} Y_i 2^i \quad (2)$$

$$X = \sum_{i=0}^{n-1} X_i 2^i \quad (3)$$

The product P = P2n-1 …P1P0, which results from multiplying the multiplicand Y by the multiplier X, can be written as follows:

$$P = XY = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (X_i . Y_j) 2^{i+j} \quad (4)$$

An n*n bit Braun multiplier is constructed with n (n-1) adders and n^2 AND gates as shown in the fig.6, where,
X : 4 bit Multiplicand
Y : 4 bit Multiplier
P : 8 bit Product of X & Y
Pn : Xi Yi is a Product bit

The internal structure of the full adder can be realized using FPGA. Each products can be generated in parallel with the AND gates. Each partial product can be added with the sum of partial product which has previously produced by using the row of adders. The carry out will be shifted one bit to the left or right and then it will be added to the sum which is generated by the first adder and the newly generated partial product.

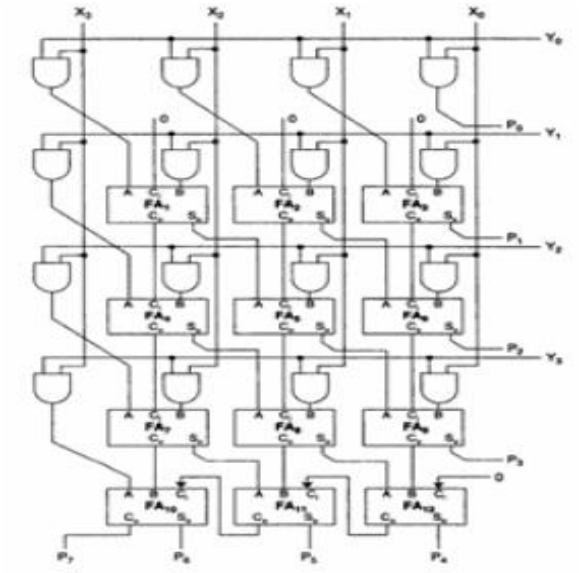


Figure 6[3]: Braun multiplier for n=4

Since the 64 bit braun multiplier is difficult to represent, a typical 4-bit architecture is shown in the above figure 6.

VI. CARRY SAVE ADDER

The carry-save unit consists of n full_adders, each of which computes a single sum and carry bit based solely on the corresponding bits of the three input numbers. Given the three n - bit numbers a, b, and c, it produces a partial sum **PS** and a shift-carry **SC.**

$$PS_i = a_i \wedge b_i \wedge c_i \quad (5)$$

$$SC_i = (a_i \wedge b_i) \square (a_i \wedge c_i) \square (b_i \wedge c_i) \quad (6)$$

Since the representation of 128 bit carry save adder is infeasible , hence a typical 8 bit carry save adder is shown in the figure 7[1].Here we are computing the sum of two 128 bit binary numbers, then 128 half adders at the first stage instead of 128 full adder. Therefore , carry save unit comprises of 128 half adders, each of which computes single sum and carry bit based only on the corresponding bits of the two input numbers[1].

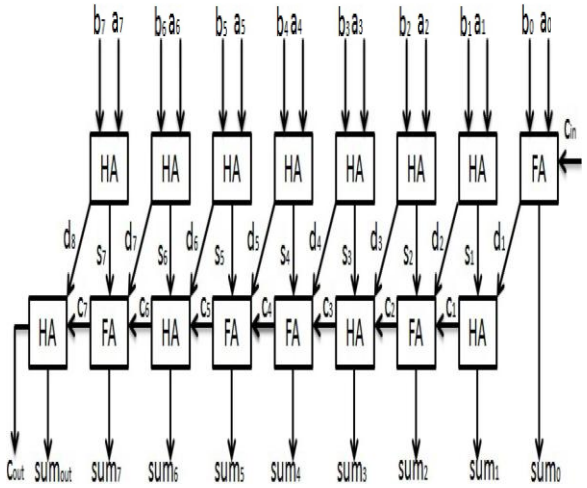


Figure 7[12]: Carry save adder for n=8

VII. RESULTS

The design is done using Verilog-HDL by using tool Xilinx ISE 10.1i and target family Spartan 3E,Device- XC3S500,speed -5,package: FG320.As a previous work,64 bit MAC is constructed using Wallace multiplier but here different MAC units are constructed and compared the performance with the earlier is done here the multipliers designed are below (i) Vedic multiplier (ii) Dada multiplier (iv) Braun multiplier (v) Wallace multiplier. The delay table and the corresponding graph for different multipliers is shown below.Hence,form the the below table it is clear that the delay for the vedic,dadda.,braun multipliers is less than the Wallace multipliers if we choose the best multiplier to implement it in MAC unit it may results in better performance and better results.

MULTIPLIER	Maximum Combinational path Delay(ns)			
	8 BIT	16 BIT	32 BIT	64 BIT
VEDIC	18.353	24.541	31.835	33.882
DADDA	20.871	28.037	35.372	37.57
BRAUN	21.049	28.155	35.444	37.642
WALLACE	26.248	33.354	40.643	42.842

TABLE I. DELAY OF DIFFERENT MULTIPLIERS

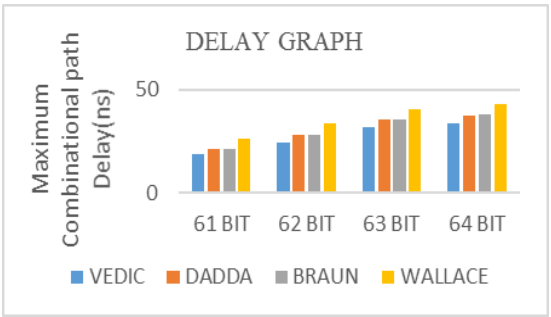


Figure 8 :Delay comparison of different multipliers.

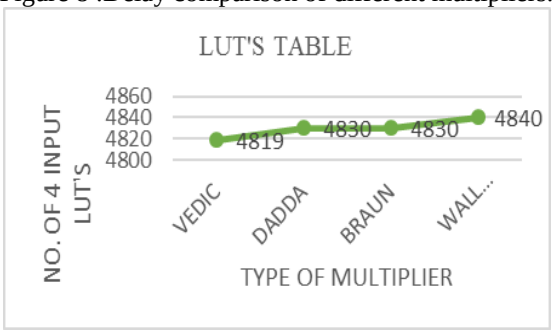
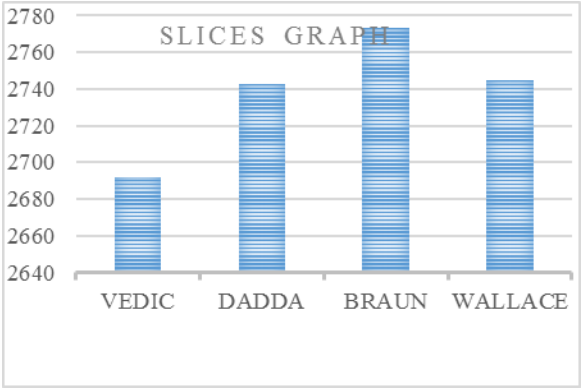


Figure 9 : No.of 4 input LUT comparison of different MAC units.

TABLE II . DELAY OF MAC UNITS.

SI. NO	MAC NAME	LOGIC DELAY (ns)	ROUTE DELAY (ns)	TOTAL DELAY (ns)
1	VEDIC MULTIPLIER CARRY SAVE ADDER	245.03	70.73	325.76
2	DADDA MULTIPLIER CARRY SAVE ADDER	251.73	79.02	330.76
3	BRAUN MULTIPLIER CARRY SAVE ADDER	260.12	83.03	343.15
4	WALLACE MULTIPLIER CARRY SAVE ADDER	263.57	86.43	350.01



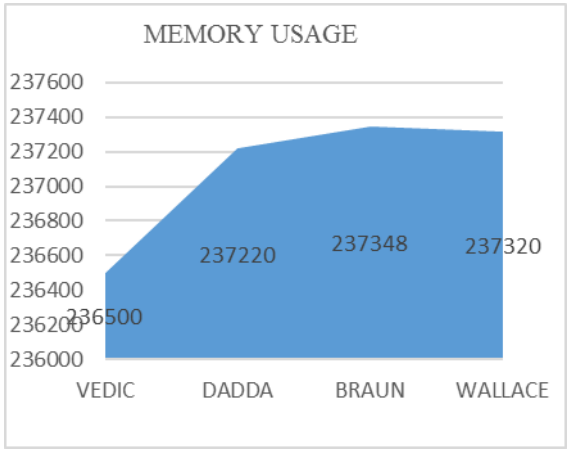


Figure 10: Slice usage of different MAC unit.
Figure 11 : Memory usage of different MAC units.

The figure 8 and 9 shows the delay and no. of 4 input LUT for different types of MAC units. The figure 10 shows the no. of slices for the above four MAC units and figure 11 represents the amount of memory taken by different MAC units. From the above all the results it is clear that the vedic,dadda,braun MAC units has least delay and less memory required than Wallace MAC unit.

The simulation results of 64 bit different types of MAC units is shown below. Most of the simulation result it is observed that there is a delay by one clock cycle, this is because it takes some time to compute since the multiplier used here is 64 bit. The overall simulated output is shown in below figures.

The below figures from figure 12 to figure 14 shows the simulation results of the different types of 64 bit MAC units.

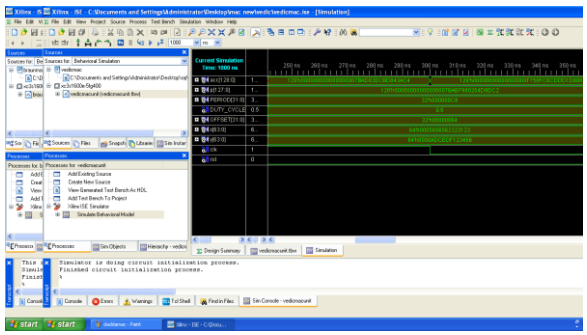


Figure 12: Simulation result of Vedic MAC unit.

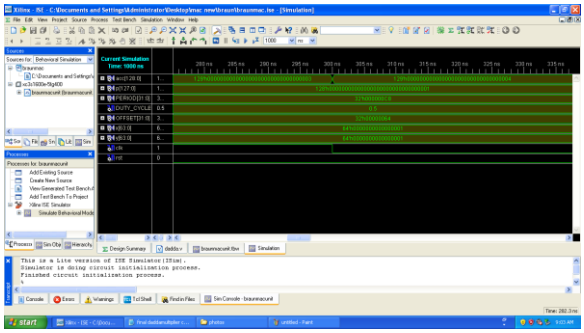


Figure 13: Simulation result of Braun MAC unit.

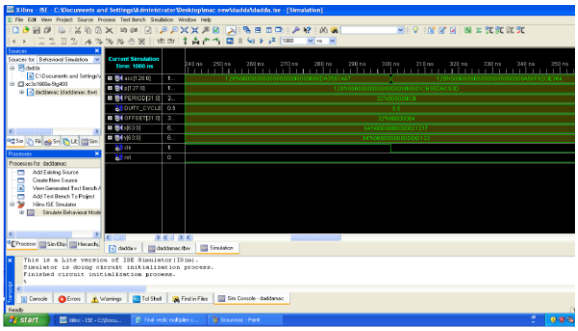


Figure 14: Simulation result of dadda MAC unit.

VIII. CONCLUSION

In this paper implementation 64 bit MAC is done using the new architecture of MAC unit and different parameters like delay, no. of LUTs, memory usage is found out. Since, multipliers are used in most of the applications we can select the best multiplier with the above parameters and get better results. Hence, using this new architecture of MAC unit with the suitable multiplier in most of the DSP applications leads to better results and performance.The Performance analysis, simulation result and comparison are reported above.

IX. ACKNOWLEDGMENT

The author would like to thank to Assistant.Prof. M.Bharathi, Assistant.Prof.K.Neelima of the VLSI Division, SVNE College for their contribution of this work.

REFERENCES

[1] P.jagadeesh**"***Design ofHigh Performance 64 bit MAC Unit*" 2013 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2013].

[2] W.J. Townsend, E.E. Swartzlander Jr., and J.A. Abraham, "*A Comparison of Dadda and Wallace Multiplier Delays*, "Proc. SPIE, Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, pp. 552-560, 2003.

[3] M.H. Rais, M.H. Al Mijalli, "*Braun's multipliers: Spartan-3AN based design and implementation*", J. Comput. Sci., vo. 7, no. (11), pp.1629-1632, 2011.

[4] L. Dadda, "*Some Schemes for Parallel Multipliers*," Alta Frequenza,vol. 34, pp. 349-356, 1965.

[5] C.S. Wallace, "*A Suggestion for a Fast Multiplier*," IEEE Trans.Electronic Computers, vol. 13, no. 1, pp. 14-17, Feb. 1964.

[6] Ron S. Waters and Earl E. Swartzlander, Jr., "*A Reduced Complexity Wallace Multiplier Reduction*, " IEEE Transactions On Computers, vol. 59, no. 8, Aug 20 10.

[7] Rais, M.H., 2009a. *FPGA design and implementation of fixed width standard and truncated 6×6-bitmultipliers: A comparative study*. Proceedings of the 4th IEEE International Design and TestWorkshop, Nov. 15-17, IEEE Xplore Press,Riyadh, Saudi Arabia, pp: 1-4.DOI:10.1109/IDT.2009.5404081.

[8] Rais, M.H., 2010a. *Hardware implementation of truncated multipliers using Spartan 3AN, Virtex-4and Virtex-5 devices*. Am. J. Eng. Applied Sci., 3:201-206. DOI: 10.3844/ajeassp.2010.201.206.

[9] Pushpalata Verma, K. K. Mehta, " *Implementation of an Efficient Multiplier based on Vedic Mathematics Using EDA Tool*," International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Volume-1, Issue-5, June 2012.

[10] Manoranjan Pradhan, Rutuparna Panda, Sushanta Kumar Sahu, "*Speed Comparison of 16x16 Vedic Multipliers*," International Journal of Computer Applications (0975 – 8887),Volume 21– No.6, May 2011.

[11] Naveen K Gahlan, Prabhat, Jasbir Kaur " *Implementation of WallaceTree Multiplier Using Compressor*" International Journal of Computer & Technology.

[12] W.J. Townsend, E.E. Swartzlander Jr., and J.A. Abraham, "*A Comparison of Dadda and Wallace Multiplier Delays*,"Proc.-SPIE, Advanced Signal Processing Algorithms, Architectures ,and Implementations XIII, pp. 552-560, 2003.