International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:7, No:1, 2013

Optimized Multiplier based upon 6-Input Luts and Vedic Mathematics

Zulhelmi Zakaria, and Shuja A. Abbasi

***Abstract*—**A new approach has been used for optimized design of multipliers based upon the concepts of Vedic mathematics. The design has been targeted to state-of-the art field-programmable gate arrays (FPGAs). The multiplier generates partial products using Vedic mathematics method by employing basic 4x4 multipliers designed by exploiting 6-input LUTs and multiplexers in the same slices resulting in drastic reduction in area. The multiplier is realized on Xilinx FPGAs using devices Virtex-5 and Virtex-6.Carry Chain Adder was employed to obtain final products. The performance of the proposed multiplier was examined and compared to well-known multipliers such as Booth, Carry Save, Carry ripple, and array multipliers. It is demonstrated that the proposed multiplier is superior in terms of speed as well as power consumption.

***Keywords*—**Multiplier, Vedic Mathematics, LUTs, FPGAs.

I. Introduction

A multiplier is a part of a processor that is widely used in digital devices such as computers, laptops, mobile phones, and so forth. It plays an important role in digital signal processing (DSPs) and image processing. Thus, the multipliers which have high speed and low power consumption are important.

For the last several decades, many attempts have been made by various researchers to develop algorithms for multipliers' design, which have high speed as well as low power consumption. However, to meet both these specifications simultaneously has been difficult. Some algorithms are suitable to design high speed multipliers, while others are appropriate for reducing area usage.

A number of multipliers such as Braun, parallel, column bypass, and carry ripple multipliers [1], [2] are slow multipliers. However, they use less area in ASIC or FPGAs implementation and power consumption is low. On the other hand, booth and Karatsuba-based Montgomery multipliers [3] are high speed multipliers with extra area used as a coding hardware. These multipliers are often used in applications requiring high speed. In the present work, an attempt is made to design a multiplier based on Vedic mathematics and exploiting 6-input LUTs of Xilinx FPGAs such that the two specifications, high speed and low power consumption are simultaneously met.

Besides speed and power issues in selecting an algorithm to design a multiplier, technology implementation is becoming the next issue.

There are two well-known technologies to realize hardware namely ASIC and FPGAs. ASIC is a full custom design, where a designer must arrange all modules on transistor level.

The ASIC technology provides high performance but has several disadvantages as given below:

- The integrated circuit costs are rising aggressively.
- ASIC complexity has lengthened development time.
- R&D resource and headcount are decreasing.
- Revenue losses for slow time-to-market are increasing.
- Financial constrains in a poor economy are driving low-cost technologies.

These trends make FPGAs a better alternative than ASIC for a large number of even higher-volume applications than they have been historically used for. Further, it is well established that full custom ASICs are the most expensive to manufacture and design and have the largest turn-around time. Thus they are preferred only when [4]:

1. There are no suitable existing cell libraries available which may be used for entire design, i.e. the cells are not small or fast enough or consume much power.
2. The ASIC technology is new or so specialized.

Therefore, FPGAs are selected for implementation of the optimized multiplier based on Vedic mathematics.

II. FPGA based Multipliers

As the rapid development on FPGA's technology, designing a low power and fast multiplier based upon FPGA is becoming realistic. A number of multipliers based on FPGAs have been realized.

Mangal et al.[5] introduced a technique of insertion of more zeros in the multiplicand to reduce switching activity and power consumption. Several advantages of this scenario are: first, average logic-switching-frequency is low; second, logic switching at each edge clock is decreased; finally, lowering capacitance of the routing network especially at high frequency. The design shows an improvement in power dissipation. However, the power reduction takes place only when a bit zero is available in the multiplicand. In addition to reducing power, a condition, where zero in the multiplicand is present in most of the time, is required. This situation is difficult to find in real life applications, where bits in the multiplicand are interchangeable between zeros and ones.

Similar work was done by Bhattacharjee, Basak, and Chakrabarti [6], who evaluated a number of multipliers in order to find out the most appropriate arithmetic circuits for FPGA, based DSP realization. The work indicated that among the six tested multipliers; Carry Save Multiplier (CSM) was the most suitable circuit for FPGA based DSP implementation. However, there are several reasons, why the result is not acceptable for DSP realization: firstly, it is not enough to evaluate only six types of multipliers to find out the most

Zulhelmi Zakaria and Shuja A. Abbasi are with the Department of Electrical Engineering, College of Engineering, King Saud University, Riyadh, Saudi Arabia (e-mail: [zajie; abbasi] @ ksu.edu.sa).

appropriate one. Secondly, CSM is a standard multiplier which has moderate performance in the case of speed and area. Finally, it may have other techniques that can realize better multipliers rather than CSM. Likewise, Aparna and Thomas [7] proposed parallel multiplier with two scenarios. First, using booth radix-4 and compressor 3:2; second, employing booth radix-8 and compressor 4:2. Even though, the two multipliers provide an alternative way of multiplier design, two drawbacks of the design arise. Firstly, complicated circuit and routing process occur at booth encoding. Secondly, the implementation of Carry Look Ahead Adder (CLA) on modern FPGA sends up with low performance in terms of latency and area.

A number of interesting methods of realizing a multiplier has been introduced in the last several decades. One of them is based on Vedic mathematics. Some of the intelligent works were dedicated to achieve high speed and low power multipliers. For example, Akhter [8] introduced a way of developing a bigger modular multiplier from a smaller one using Vedic multiplication. This was followed by Gurumurthy and Prahalad [9] who implemented a Vedic multiplier with bit width 16x16, and evaluated with Booth radix-4 multiplier. Tiwari et al., [10] proposed Vedic Multiplier with carry look a head adder as a final addition circuit. Array and Booth multipliers were used for comparison. The comparison indicated that Vedic multiplier proposed by [8]-[10] had less delay and area compared to array and booth multipliers. These techniques demonstrate better performance in terms of area and speed; however, ignoring enhancement of sub-multiplication and final adder blocks results in decreased performance when the multipliers are extended to higher bit width. Further, comparison with Array and Booth multipliers is inadequate to show the design superiority.

Mehta and Gawali [11] evaluated a number of multipliers; five of them were designed based upon Vedic multiplication, and the rest ones were implemented using array and Xilinx Synthesis Tool's (XST) techniques. The data indicated that the multiplier designed by the XST's technique (employing the operator '*' in HDL code) offered better results in terms of delay and area. In other words, [11] demonstrated that the XST's approach has the best way of realizing a multiplier. Although in some cases, XST provides maximum optimization in designing multipliers, it is not always true that his method gives a good approach to yield a multiplier based on FPGA. In addition, there are other tools, e.g. from Altera and SYNOPSYS, which may also provide better results.

The latest works related to Vedic multiplier were presented by Kunchigi, and Kulkarni [12], who introduced a 4x4 Vedic multiplier with a pipeline scheme to increase speed. They demonstrated that their technique gave better results compared to Booth and Array multipliers. In the same year, Sriraman and Prabakar [13] offered a multiplier based on ROM approach, where the ROM used to store the squares of average and deviation of input numbers. With this approach, the speed of the multiplier increased significantly. However, Speed improvement over the approach depends on the reading process of ROM and final subtraction process to get the final result. Moreover, the required area is becoming huge for bit width higher than the 8 bit multiplier.

III. Proposed Multiplier

The present work is based upon the Vedic multiplier techniques. A brief account of this is given below:

A. Vedic Mathematics

The Vedic mathematics is a store of knowledge explaining a number of mathematical branches such as algebra, trigonometry, and geometry, etc. One of the formulas (generally referred to as "*sutras*" in Vedic mathematics) is known as "*Urdhva-tiryakbhyam*" meaning vertically and crosswise [12]. This formula is mainly used to perform multiplication with a scheme as depicted in Fig. 1.

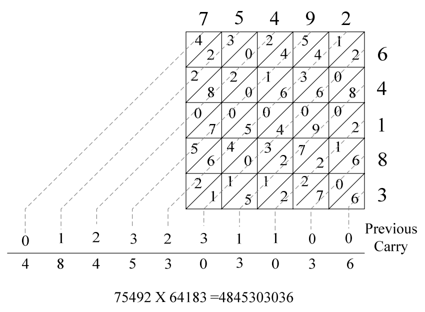


Fig. 1 Vertically and crosswise formula in multiplication

This figure describes multiplication of two numbers 75492 and 64183 (as an example). It may be seen that multiplication in decimal numbersis easy. Similarly, in binary, numbersthis operation may be done more easily since the cells in this case will only contain zeros or ones.

Fig. 2 illustrates as to how 8x8 binary multiplication is performed. Two 8-bit terms, *multiplier* and *multiplicand*, are split into four 4-bit terms. Each 4-bit term in *multiplier* is multiplied with two 4-bit terms in multiplicand yielding partial products as indicated in the Figure. These partial products, consisting of three rows, are then reduced with Carry Save Adder (CSA) to two rows. Moreover, the last partial products are ready to be added to yield final products.

The multiplication, based upon this technique may be done using FPGAs. The multiplier will basically require three sub-circuits. One of them will do sub-group multiplication. The second one will be used for reduction of partial products. Finally, the third sub circuit will do the addition to give the result.

The performance of the multiplier primarily depends on the sub-group multiplication circuit and the final addition circuit. Careful optimization of these two circuits is therefore essential. The subgroup multiplication circuit must be optimized for speed and area. For final addition, the adders available on the FPGAs are used. These adders are basically carry ripple/chain adders and have been optimized during the fabrication stage of the FPGAs. This optimization is done to minimize delay and simplify routing.

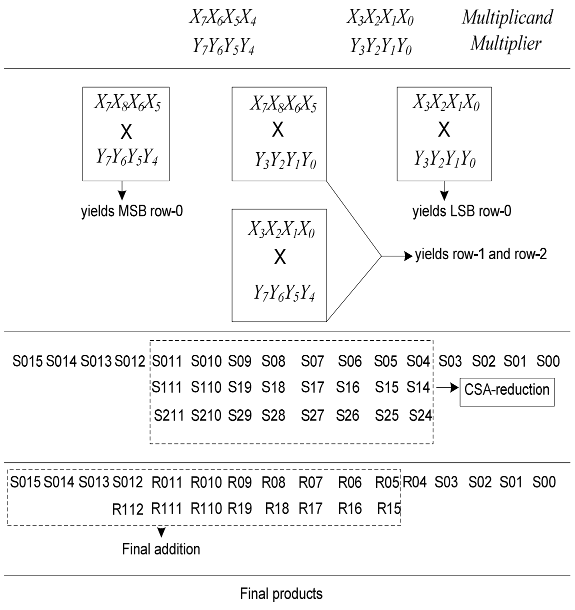


Fig. 2 8x8 proposed multiplier

B. Subgroup Multiplier

Arrangement of a block circuit or a sub block circuit within the same slice or CLBs is very important in FPGAs design. A number of advantages can be achieved with this scenario: first, the routing process becomes faster; second, time delay is minimized; finally, the performance of the circuit is enhanced.

Fig. 3 depicts Configurable Logic Blocks (CLBs) and slices of Virtex-5 Xilinx FPGA. Each CLB contains two slices. Each slice consists of four 6-input LUTs, four flip-flops, and supported logic circuitry such as MUXF7, MUXF8, and dedicated XOR gates (not shown in the Fig).

One of the new features of Virtex-5 is the availability of 6-input LUTs where designers are able to design a circuit having six inputs with one level logic.

Previous generations of FPGAs with 4-input LUTs takes two level logics to design 6 input circuits, for example. In addition, Virtex-5 and the latest generations of FPGAs are fabricated with new technology yielding better performance compared to the old generations. Hence, Virtex-5 devices are used for the present design.

The proposed subgroup multiplier is depicted in Fig. 4. This sub multiplier involves two 4-bit operands (eight *inputs*- X_0,X_1,X_2,X_3, and Y_0,Y_1,Y_2,Y_3), and eight outputs ($P_0,P_1,P_2,\ldots,P_7$).

The first three outputs are simply attained by exploiting only LUTs, while other five outputs require Multiplexers, MUXF7 and MUXF8 as indicated in the Fig. 4.

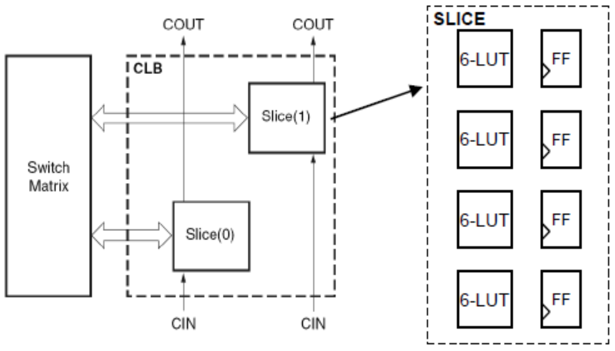


Fig. 3 Virtex-5 CLB and Slice [14]

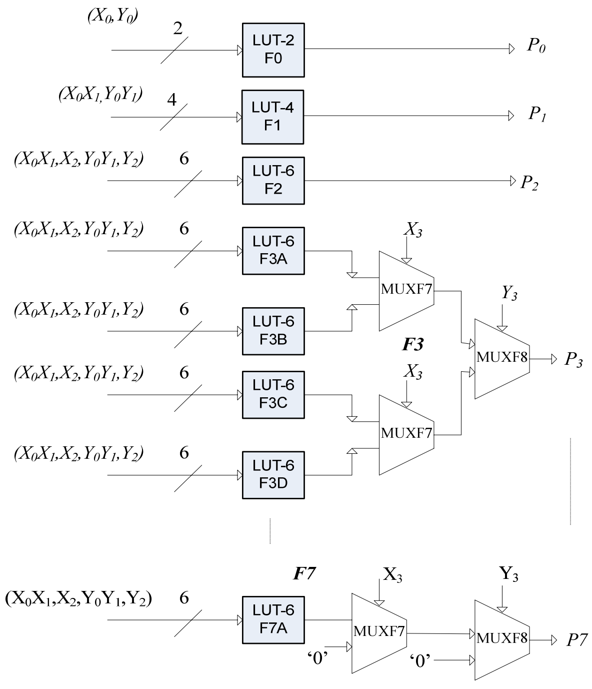


Fig. 4 4x4 multiplier based LUTs

IV. RESULTS AND DISCUSSION

The proposed Vedic multiplier and its corresponding blocks are described using structural Verilog-HDL and synthesized employing Xilinx Synthesis Tool (XST), WebPACK version 13.3.The implementation was targeted to Xilinx Virtex-5, device XC5VLX30. From the implementation results as indicated in Table I, it is found that the proposed 8x8 multiplier has a delay which is roughly half of the other six multipliers reported in [6]. In the case of the occupied area, the proposed design comprises about 23 slices, which is slightly larger than CSM design.

Moving to higher multipliers, Fig. 5 shows the histogram stating the proposed 16x16 multiplier (PM) that offers high speed with delay less than half of the other multipliers except CSM and BSM III. In terms of the occupied slice, it shows a small increase compared to CSM and BSM II.

TABLE I
OCCUPIED SLICES AND MAXIMUM DELAYS

Multiplier width	Multiplier type	Delay (ns)	No. of slices
8 x 8	Basic Multiplier	15.67	30
	Carry Save Multiplier	8.90	21
	Carry Ripple Multiplier	14.08	23
	Boot Sign Multiplier I	16.64	35
	Boot Sign Multiplier II	11.56	23
	Boot Sign Multiplier III	11.52	26
	Proposed Multiplier	**6.59**	**23**

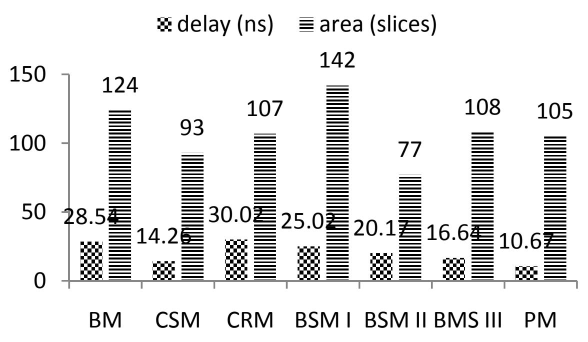


Fig. 5 Delay and area comparison of 16x16 multipliers

A sharp increase in delays occurs at 32x32 multipliers as described in [6]. It is almost double compared to 16x16 multipliers. In contrast, with the proposed Multiplier, only a small, insignificant increase in delay is found which is about 4.8 ns as can be seen in Fig. 6. In the case of area occupation, PM places middle position among of the multipliers. It is fair to sacrifice small slices to achieve the best performance.

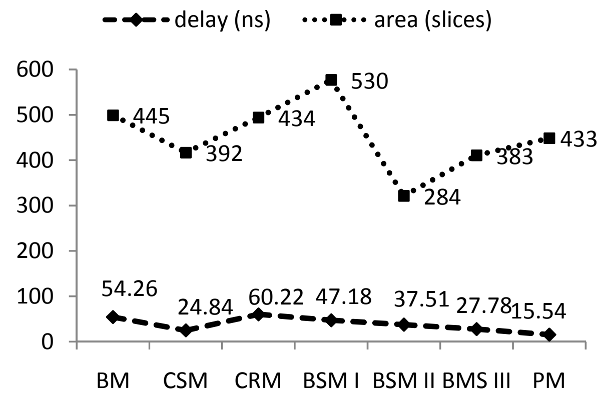
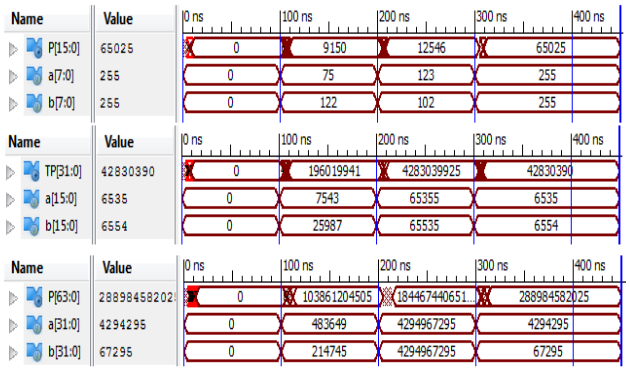


Fig. 6 The association of area and delay in 32x32 multipliers

To ensure that the hardware implementation works properly, simulation test was performed using ISim (O.76.xd). Fig. 7 indicates the post-route simulation result of the proposed multiplier. The test was done for different bit width, starting with 8x8, 16x16, and 32x32 bits. For convenience, the decimal number is used for displaying digit inputs and outputs of the hardware.



Name	Value	0 ns	100 ns	200 ns	300 ns
P[15:0]	65025	0	9150	12546	65025
a[7:0]	255	0	75	123	255
b[7:0]	255	0	122	102	255

Name	Value	0 ns	100 ns	200 ns	300 ns
TP[31:0]	42830390	0	196019941	4283039925	42830390
a[15:0]	6535	0	7543	65355	6535
b[15:0]	6554	0	25987	65535	6554

Name	Value	0 ns	100 ns	200 ns	300 ns
P[63:0]	28898458202	0	103861204505	184467440651...	288984582025
a[31:0]	4294295	0	483649	4294967295	4294295
b[31:0]	67295	0	214745	4294967295	67295

Fig. 7 Simulation result of the proposed multiplier

Another implementation was also done for Virtex-6, device XC6VLX75T. Although the FPGA's generations and the device categories are diverse, a comparison of the proposed multiplier with Virtex-5 implementation is described to get a brief view of the different technology's realization. Fig. 8 illustrates the variation of delays when PM was implemented on Virtex-6 and Virtex-5. From the histogram, it can be seen that the delays are slightly lower in Virtex-6 compared to Virtex-5. In addition, the area occupation of both implementations is shown in Table II. Even though the area occupation looks closed to each other, the physical area is much more on Virtex-5 compared to Virtex-6. This is understandable since Virtex-5 is fabricated using 65nm technology where asVirtex-6'sfabricated with 40nm technology [15], [16].

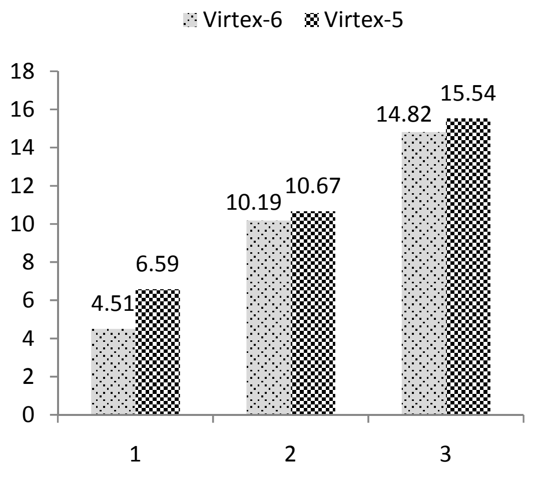


Fig. 8 Maximum delays on Virtex-5 and Virtex-6

TABLE II
OCCUPIED SLICES ON VIRTEX-5 AND VIRTEX-6

Multiplier width	Multiplier type	Virtex-5	Virtex-6
8 x 8	Proposed Multiplier (PM)	24	25
16 x 16		105	108
32 x 32		433	439

V. CONCLUSION

This paper presents a new approach for design and hardware realization of multipliers based upon the concept of Vedic mathematics. The technique is applied on sub-multiplication blocks, which are the fundamental blocks of the Vedic multiplier. The sub-multiplication blocks are optimized by employing 6-input LUTs and multiplexers within the same slices or CLBs. By setting placement and design goal strategies, the results of the proposed multiplier show better performance in terms of speed compared to the multipliers reported in literature. In terms of the device utilization also, in most of the cases studied, better results are obtained. Finally, it may be mentioned that in the present work hardware realization is based on FPGAs. It is suggested that the proposed multiplier technique may be considered for transistor level optimization using full-custom implementation in order to get the best performance.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from National Plan for Science, Technology and Innovation (NPST), Saudi Arabia under project no. 09-ELE854-02.

REFERENCES

[1] J.Senthil Kumar , G.Sriram, G.Lakshminarayanan , B.Venkataramani, " Design and Implementation of FPGA based Fast Multipliers with Optimum Placement & Routing Using Structure Organizer", National Conference on VLSI design & Testing, PSG College of Technology, Coimbatore, 21-22 February, 2003

[2] T. V. More and R. V. Kshirsagar, "Design of low power column bypass multiplier using FPGA," in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011, pp. 431-435.

[3] G. C. T. Chow, K. Eguro, W. Luk, and P. Leong, "A Karatsuba-Based Montgomery Multiplier," in Field Programmable Logic and Applications (FPL), 2010 International Conference on, 2010, pp. 434-437.

[4] Michael John Sebastian Smith, “Application-specific integrated circuits”, Addison-Wesley, (1997).

[5] S. K. Mangal, R. B. Deshmukh, R. M. Badghare, and R. M. Patrikar, "FPGA Implementation of Low Power Parallel Multiplier," in VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on, 2007, pp. 115-120.

[6] S. Bhattacharjee, S. Sil, B. Basak, and A. Chakrabarti, "Evaluation of power efficient adder and multiplier circuits for FPGA based DSP applications," in Communication and Industrial Application (ICCIA), 2011 International Conference on, 2011, pp. 1-5.

[7] P. R. Aparna and N. Thomas, "Design and implementation of a high performance multiplier using HDL," in *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, 2012, pp. 1-5.

[8] S. Akhter, "VHDL implementation of fast NxN multiplier based on vedic mathematic," in *Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on*, 2007, pp. 472-475.

[9] K. S. Gurumurthy and M. S. Prahalad, "Fast and power efficient 16 X16 Array of Array multiplier using Vedic Multiplication," in *Microsystems Packaging Assembly and Circuits Technology Conference (IMPACT), 2010 5th International*, 2010, pp. 1-4.

[10] H. D. Tiwari, G. Gankhuyag, K. Chan Mo, and C. Yong Beom, "Multiplier design based on ancient Indian Vedic Mathematics," in *SoC Design Conference, 2008. ISOCC '08. International*, 2008, pp. II-65-II-68.

[11] P. Mehta and D. Gawali, "Conventional versus Vedic Mathematical Method for Hardware Implementation of a Multiplier," in *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09. International Conference on*, 2009, pp. 640-642.

[12] V. Kunchigi, L. Kulkarni, and S. Kulkarni, "High speed and area efficient Vedic multiplier," in Devices, Circuits and Systems (ICDCS), 2012 International Conference on, 2012, pp. 360-364.

[13] L. Sriraman and T. N. Prabakar, "Design and implementation of two variable multiplier using KCM and Vedic Mathematics," in *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on*, 2012, pp. 782-787.

[14] Ahmed et. al, “Architecture-Specific Packing for Virtex-5 FPGAs,” *FPGA’08*, February 24-26, 2008, Monterey, California, USA.

[15] Xilinx, “ds100, Virtex-5 Family Overview,” February 6, 2009, Xilinx. Inc.

[16] Xilinx, “ds150, Virtex-6 Family Overview,” March 24, 2011, Xilinx. Inc.