An Improved Squaring Circuit for Binary Numbers

Kabiraj Sethi
Department of Electronics and
Telecommunication Engineering,
VSS University of Technology, Burla
Formerly UCE, Burla-768018. (India)

Rutuparna Panda
Department of Electronics and
Telecommunication Engineering,
VSS University of Technology, Burla
Formerly UCE, Burla-768018. (India)

***Abstract*—In this paper, a high speed squaring circuit for binary numbers is proposed. High speed Vedic multiplier is used for design of the proposed squaring circuit. The key to our success is that only one Vedic multiplier is used instead of four multipliers reported in the literature. In addition, one squaring circuit is used twice. Our proposed Squaring Circuit seems to have better performance in terms of speed.**

Keywords-Vedic mathematics; VLSI; binary multiplication; hardware design; VHDL.

I. INTRODUCTION

Multiplication and squaring are most common and important arithmetic operations having wide applications in different areas of engineering and technology. The performance of any circuit is evaluated mainly by estimating the silicon area and speed (delay). Hence, continuous efforts are being made to achieve the same. In order to calculate the square of a binary number, fast multipliers such as Braun Array, Baugh-Wooley methods of two's compliment, Booth's algorithm using recorded multiplier and Wallace trees are in use. Recursive decomposition and Booth's algorithm are the most successful algorithms used for multiplication. Other methods include Vedic multipliers based on 'Urdhva Tiryagbhyam' and the "Duplex" properties of 'Urdhva Tiryagbhyam'. Therefore, the main motivation behind this work is to investigate the VLSI Design and Implementation of Squaring Circuit architecture with reduced delay. Interestingly, only one multiplier is used here instead of four multipliers reported in the literature. Here, one squaring circuit is used twice to reduce delay.

Vedic mathematics was reconstructed from Vedas by Sri Bharati Krisna Tirthaji (1884-1960) after his eight years of research on Vedas [1-3]. According to him, Vedic mathematics is mainly focused on sixteen very important principles or word-formulae, which are otherwise known as Sutras. Note that the most important feature of the Vedic mathematics is its coherence. The entire system is wisely interrelated and unified. The general multiplication scheme can easily be reversed to achieve one-line divisions. Similarly, the simple Squaring Scheme can easily be reversed to produce one-line Square Roots. These methods are very easy to understand. This paper discusses a possible application of Vedic mathematics to design multipliers and squaring circuits. General idea for design of digital multipliers is described in [4, 5].

The idea of using Vedic mathematics for design of multipliers has been discussed in [6-13]. In [14], 'Urdhva Tiryagbhyam' Sutra is shown to be an efficient multiplication algorithm as compared to the conventional counterparts. Authors of [15] have also shown the effectiveness of this Sutra to reduce N×N multiplier structure into an efficient 4×4 multiplier structure. However, they have mentioned that 4×4 multiplier section can be implemented using any efficient multiplication algorithm. In [16], authors have presented an array multiplier architecture using Vedic Sutra. More or less the coding is done in VHDL and synthesis is done in Xilinx ISE series [17,18].

Recently, a squaring circuit has been reported in the literature [19]. This may be noted that designing Vedic multipliers using array multiplier structures as discussed in above references provide us less delay and, thus, they are treated as high speed multipliers as compared to Booth's algorithm using recorded multipliers and Wallace trees. However, we can reduce delay further using carry save adders (CSA). The idea of using CSA for design of digital multipliers is explained in [3,4]. This has motivated us to design Vedic multipliers based on CSA [20]. One more crucial issue with the earlier proposed methods is that they use four numbers of such Vedic multipliers to evaluate squaring of a n-bit binary number. Here, we have more focus on the issue and tried to use only one Vedic multiplier instead of four for evaluating square of a n-bit binary number.

We apply Vedic Sutras to binary multipliers using carry save adders. In particular, we develop an efficient binary multiplier architecture that performs partial product generations and additions in parallel. With proper modification of the Vedic multiplier algorithm, the squaring circuit is developed. Here, the computation time involved is less. The combinational delay and the device utilizations obtained after synthesis is compared. Our proposed Vedic multiplier based Squaring Circuit seems to have better performance in terms of speed. The hardware architecture of the squaring circuit is presented.

II. THE MULTIPLIER ARCHITECTURE

Booth's multipliers [5] are normally used for squaring of binary numbers. The modified Booth multiplier considered uses four components. Booth Encoder, Partial Product Generator, Wallace Tree and Binary Adders are used for Booth multiplier architecture. Booth multiplier uses two main ideas to increase the speed of the multiplication process. First attempt is to reduce the number of partial products.

Then the second attempt is to increase the speed at which the partial products are added. The partial products are reduced

using Booth encoder. Time for partial product additions is reduced using Wallace Tree. Further, Binary adder is used for addition of final sum vector and carry vector.

In this section, we propose an efficient multiplier architecture using Vedic mathematics. The 'Urdhva Tiryagbhyam' (Vertically and Crosswise) sutra [2] has been traditionally used for the multiplication of two numbers in the decimal number system. In this paper, we apply this Sutra to the binary number system. The motivation behind the extension to binary number system is to make it compatible with the digital hardware circuits. This Sutra is illustrated with the help of a numerical example, where two decimal numbers are multiplied.

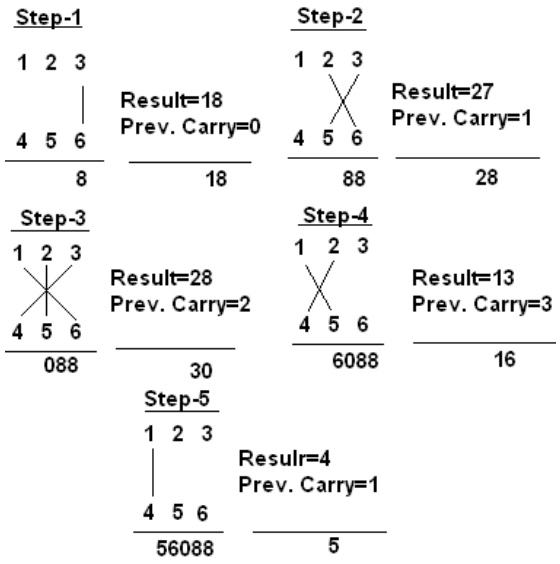

Figure 1. Line diagram for multiplication.

For more clarity, line diagram for the multiplication of two decimal numbers (123×456) is displayed in Fig. 1. Note that the digits on two ends of the line are multiplied and then results are added with the previous carry, as shown in the figure. When we find more lines in one step as shown in steps 2 to 4, all results are added to the previous carry. Interestingly enough, the least significant digit of the number, thus, obtained acts as one of the result digits. The rest digit acts as the carry for the next step. Note that here the initial carry is zero.

We now extend the above idea (for multiplication) to binary number system. Computer arithmetic [3,4] usually deals with binary number systems. Thus, there is a strong need to develop efficient schemes for multiplication of binary numbers. This may be noted that multiplication of two bits A0 and B0 is nothing but an AND operation.

Interestingly, this operation can easily be implemented using a two input AND gate used in digital circuits. Such types of multipliers using digital circuits similar to array multipliers are discussed in [21,22]. In order to illustrate this coveted multiplication scheme in binary number system, here we consider the multiplication of two binary numbers (4 bits) A3A2A1A0 and B3B2B1B0. As the result of this multiplication would be more than 4 bits, we express it as ……R3R2R1R0. As an illustration, line diagram for multiplication of two 4-bit binary numbers is shown in Fig. 2. The same idea can be extended to higher bits. It is noteworthy to mention here that Fig.2 is simply the mapping of Fig.1 in binary system. For the sake of simplicity, each bit is represented by a cross enclosed by a circle. Least Significant Bit (LSB) R0 is obtained by multiplying the LSBs of the multiplicand and the multiplier. Here, the multiplication process is carried out according to steps displayed in Fig. 2. Further, digits on both sides of the line are multiplied and added with the carry from the previous step. All seven steps shown in Fig.2 are important. This generates one of the bits of the result (Rn) and a carry (Cn). This carry is added in the next step and, hence, the process goes on. If more than one line are there in one step, all results are added to the previous carry. In each step, least significant bit acts as the result bit and other bits act as carry. To be more specific, if in some intermediate step, the sum is '110', then '0' acts as the result bit and '11' as the carry (which is denoted as Cn in this paper). It is noteworthy to mention here that Cn may be a multi-bit number. Thus, we get the following expressions:

$$R_0 = A_0B_0 \quad (1)$$
$$C_1R_1 = A_1B_0 + A_0B_1 \quad (2)$$
$$C_2R_2 = C_1 + A_2B_0 + A_1B_1 + A_0B_2 \quad (3)$$
$$C_3R_3 = C_2 + A_3B_0 + A_2B_1 + A_1B_2 + A_0B_3 \quad (4)$$
$$C_4R_4 = C_3 + A_3B_1 + A_2B_2 + A_1B_3 \quad (5)$$
$$C_5R_5 = C_4 + A_3B_2 + A_2B_3 \quad (6)$$
$$C_6R_6 = C_5 + A_3B_3 \quad (7)$$

with C6R6R5R4R3R2R1R0 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 [8].

The multiplier architecture is explained below. Both 2X2 Vedic multiplier module and 4X4 Vedic multiplier architecture are displayed below. The motivation is to reduce delay. Multiplier design ideas are well explained in [3,4]. Here, 'Urdhva Tiryagbhyam' (Vertically and Crosswise) sutra [2] is used to propose such an architecture for the multiplication of two binary numbers.

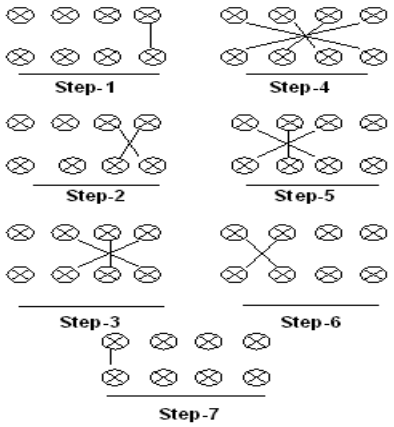

Figure 2. Line Diagram for multiplication of two 4-bit binary numbers.

A. 2x2 Vedic Multiplier Module For Binary Numbers

Here, an efficient Vedic multiplier using carry save adder is presented. The 2X2 Vedic multiplier module is implemented

using two half-adder modules and is displayed in Fig. 3. Very precisely we can state that the total delay is only 2-half adder delays, after final bit products are generated.

It is wise to write Implementation Equations of 2X2 Vedic multiplier module for simulation. The implementation equations are written as:

R_0 (1-bit) = $A_0.B_0$ (8)
R_1 (1-bit) = $A_1.B_0 + A_0.B_1$ (9)
R_2 (2-bits) = $A_1.B_1 + R_1$ (1) (10)
Product = R_2 & R_1 & R_0 (11)

where '&' denotes concatenation operation. Note that final result (product) is obtained using Eq.(11).

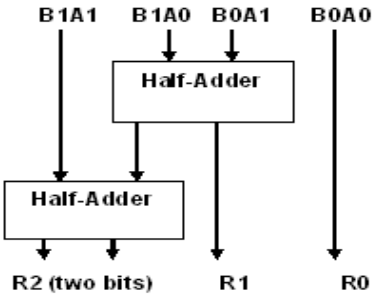

Figure 3. Architecture of 2X2 multiplier.

B. 4x4 Vedic Multiplier Module

The 4X4 Vedic multiplier architecture is displayed in Fig.4. This is implemented using four 2X2 Vedic multiplier modules as discussed in Fig. 3. 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.

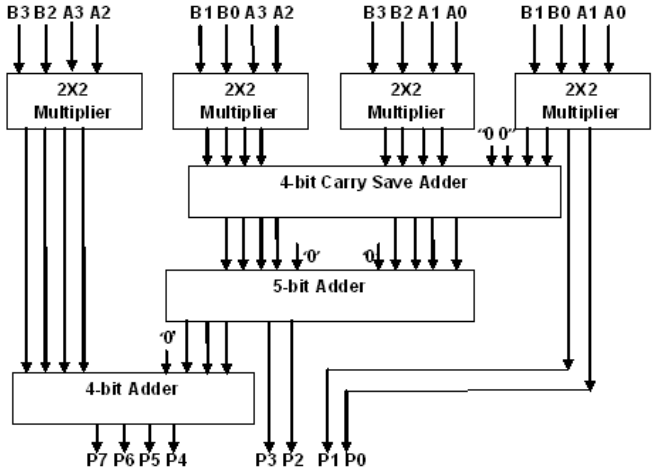

Figure 4. Architecture of 4X4 multiplier

In this section, we describe the architecture of 4X4 multiplier using Vedic method discussed above (Eqns.8-11). To get final product (P7P6P5P4P3P2P1P0), one 4-bit carry save adder, one 5-bit binary adder and one 4-bit binary adder are used. In this proposal, the 4-bit carry save adder (CSA) is used to add three 4-bit operands, i.e. concatenated 4-bit ("00" & most significant two output bits of right hand most of 2X2 multiplier module as shown in Fig.3) and two 4-bit operands we get from the output of two middle 2X2 multiplier modules. It may be noted that the outputs of the CSA (sum and carry) are fed into a 5-bit binary adder to generate 5-bit sum, as desired. Many more interesting ideas can be revoked here.

It may be reiterated the fact that the middle part (P3P2) denotes the least significant two bits of 5-bit sum obtained from the 5-bit binary adder. Finally, as shown in Fig.4, the 4-bit output of the left most 2X2 multiplier module and concatenated 4-bits ('0' & the most significant three bits of 5-bit sum) are fed into a 4-bit binary adder. In this architecture, the P7P6P5P4 express the sum.

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 CSA and binary adders shown help us to reduce dely. Interestingly, 8X8 and 16X16 Vedic multiplier modules are implemented easily by using four 4X4 and four 8X8 multiplier modules, respectively. Further, the proposed 4X4 Vedic multiplier can also be used for squaring of a 4-bit binary number.

C. 2-Bit Squaring Circuit

The 2X2 Vedic multiplier architecture is modified as shown in Fig. 5 to realise the 2-bit squaring circuit. Here, one half adder and one AND gate are utilized instead of two half-adders as shown in Fig. 3.

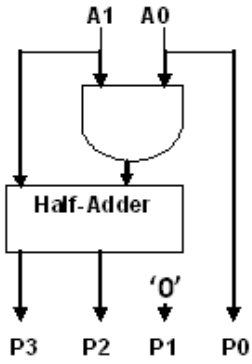

Figure 5. Block Diagram of 2-bit Squaring Circuit.

This is the basic module. Note that a 4-bit squaring circuit is implemented using two 2-bit squaring circuits (as shown in Fig.5) and one 2X2 Vedic Multiplier (as displayed in Fig.3) instead of four 2X2 Vedic Multiplier modules used in Fig.4.

In the same manner, 8-bit squaring circuit and 16-bit squaring circuits are implemented using Vedic multiplier module and squaring circuits of 4-bit and 8-bit, respectively. Likewise, n-bit squaring circuit can be implemented taking one (n/2)-bit Vedic multiplier module and two (n/2)-bit squaring circuits.

D. n-Bit Squaring Circuit

Taking the architectural concept of 4X4 Vedic multiplier module, general block diagram of the newly proposed n-bit squaring circuit is shown in Fig. 6.

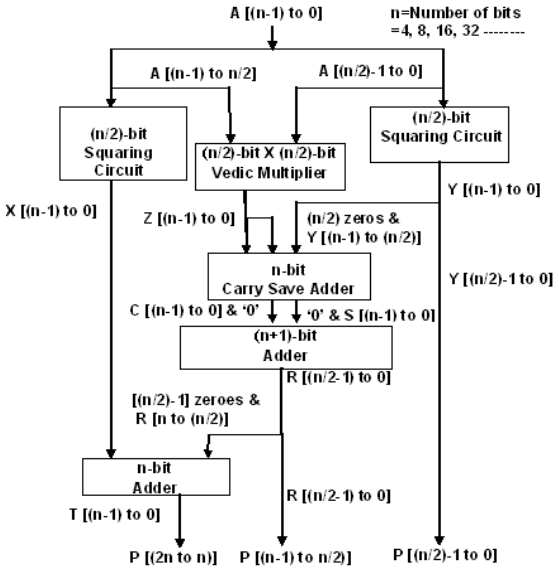

Figure 6. Architecture of n-bit Squaring Circuit.

Let us describe the proposed architecture of n-bit Squaring Circuit in a tabular form. Table-I explains the idea.

TABLE I. N-BIT SQUARING CIRCUIT DESCRIPTION

Bit size	Description	Numbers used
2 bit	AND Gate XOR Gate	2 1
4 bit	2X2 Vedic Multiplier 2 bit squaring circuit CSA Binary Adder	1 2 1 2
8 bit	4X4 Vedic Multiplier 4 bit squaring circuit CSA Binary Adder	1 2 1 2
16 bit	8X8 Vedic Multiplier 8 bit squaring circuit CSA Binary Adder	1 2 1 2
32 bit	16X16 Vedic Multiplier 16 bit squaring circuit CSA Binary Adder	1 2 1 2
64 bit	32X32 Vedic Multiplier 32 bit squaring circuit CSA Binary Adder	1 2 1 2

III. VERIFICATION AND IMPLEMENTATION

In this work, 4-bit, 8-bit, 16-bit, 32-bit and 64-bit squaring circuits are implemented in VHDL [17]. Logic synthesis and simulation are done in Xilinx - Project Navigator and Modelsim simulator [18].

We compare our synthesis results with the method recently presented by Prabha et al [19]. The results are displayed in Table 2 and Table 3 for squaring circuits of different bit size. These Tables show the difference in combinational delays and the device utilization. Squaring circuits of different bit size are considered for simulation. Comparison of combinational delay in nano seconds (ns) is displayed in Table-II.

TABLE II. COMPARISON OF COMBINATIONAL DELAY (NS)

Device: Vertex4vlx 15sf363-12	Modified Booth Multiplier [19]	Prabha et al [19]	Ours
4 bit	8.154	4.993	4.993
8 bit	15.718	14.256	12.781
16 bit	36.657	33.391	15.994
32 bit	74.432	68.125	18.272
64 bit	141.982	129.867	22.905

TABLE III.
COMPARISON OF DEVICE UTILISATION (4 INPUT LUTS)

Device: Vertex4vlx 15sf363-12	Modified Booth Multiplier [19]	Prabha et al [19]	Ours
4 bit	32	6	6
8 bit	186	35	64
16 bit	880	294	366
32 bit	2760	1034	1267
64 bit	6854	4535	5361

Comparison of device utilization (4 input LUTs) is shown in Table-III. The performance of all squaring circuits are evaluated on the same device Vertex4vlx15sf363 with a speed grade of -12.The results suggest that the proposed architecture is faster than "Modified Booth Multiplier" and "the method recently presented by Prabha et al" [19]. Here, we see significant speed improvement though there is a small increase in area. Simulation results obtained are shown in figures 7-9 for verification. Simulation results for 32-bit and 64-bit are not presented here to avoid consuming space. However, they are also verified and found correct.

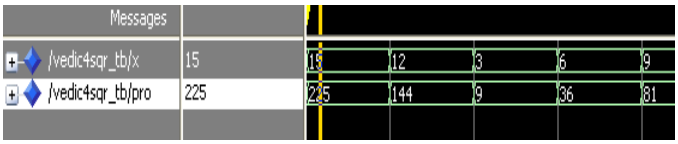

Figure 7. 4-bit Squaring Circuit.

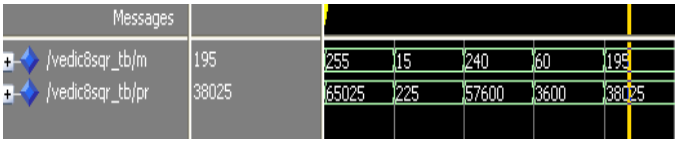

Figure 8. 8-bit Squaring Circuit.

Figure 9. 16-bit Squaring Circuit.

It is worthy to mention here that the results displayed in Table-II are quite expected. The delay is about 2.5 times when we go from 4-bit to 8-bit, in our case. However, the increase in combinational delay is less for higher bits, which is due to inherent parallelism.

To be very precise, the implementation equations (Eqs.8-11) are well adapted to parallel processing. For 8-bit squaring, we are using two 4-bit squaring circuits and one 4X4 Vedic multiplier followed by one CSA and two binary adders as shown in Table-I. Hence, increase in delay is more when we move from 4-bit to 8-bit. But we could exploit the benefit of parallelism while implementing squaring circuits of higher bit size, i.e. 16-bit, 32-bit and 64-bit as displayed in Table-II.

Thus, our method outperforms other methods in terms of speed. The proposed squaring circuit may be useful for the design of hardware for computer arithmetic.

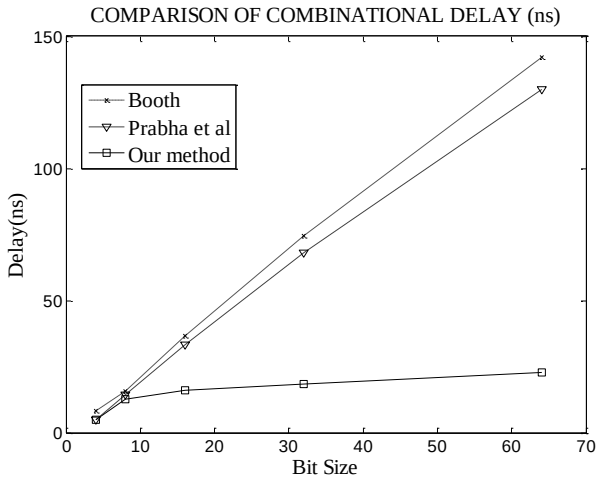

Figure 10. Comparison of Combinational Delay (ns).

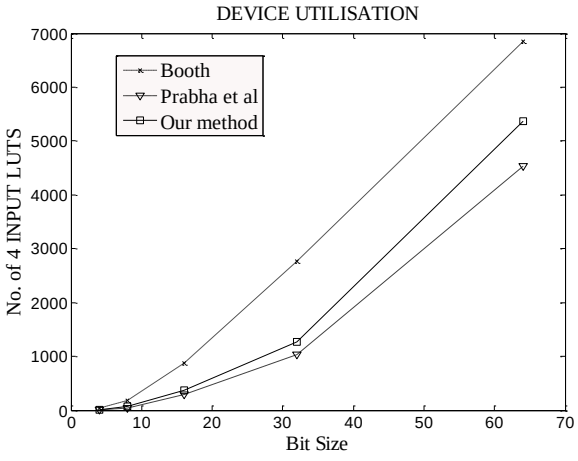

Figure 11. Comparison of Device Utilisation (4 Input LUTs).

To make things explicitly clear, delays for different bit size are displayed in Fig.10. It is observed that the delay is significantly less in our method, particularly for bit size more than or equal to 16. Therefor there is a significant speed improvement in our case.

The reason may be due to the fact that only one multiplier is used instead of four multipliers reported in [19]. However, an engineering tradeoff is observed between Fig.10 and Fig.11. Device utilisation curves are displayed in Fig.11 for a comparison. Space requirement is slightly more in our case as compared to the scheme proposed by Prabha et al [19]. The reason may be due to the fact that we use two squaring circuits of size (n/2)-bits and one CSA.

IV. CONCLUSION

The performance of the proposed squaring circuit using Vedic Mathematics proved to be efficient in terms of speed. Due to its regular and parallel structure, it can be realised easily on silicon as well. Squaring of binary numbers of bit size other than powers of 2 can also be realized easily. For example, squaring of a 24-bit binary number can be found by using 32-bit squaring circuit with 8 MSBs (of inputs) as zero. The idea proposed here may set path for future research in this direction. Future scope of research is to reduce area requirements.

REFERENCES

[1] Maharaja, J.S.S.B.K.T., "Vedic mathematics," Motilal Banarsidass Publishers Pvt. Ltd, Delhi, 2009.

[2] Swami Bharati Krisna Tirtha, "Vedic Mathematics," Motilal Banarsidass Publishers, Delhi, 1965.

[3] Vedic Mathematics [Online]. Available: http://www.hinduism.co.za/vedic.htm.

[4] B. Parhami, "Computer Arithmetic Algorithms and Hardware Architectures," 2nd ed, Oxford University Press, New York, 2010.

[5] Kai Hwang, "Computer Arithmetic: Principles, Architecture and Design," New York: John Wiley & Sons, 1979.

[6] S. Akhter,, "VHDL implementation of Fast NxN Multiplier Based on Vedic Mathematics," Proc. of IEEE Conference, pp.472-475, 2007.

[7] P.D. Chidgupkar, and M.T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing," Global Journal of Engng. Educ., vol.8 , pp.153-158, 2004.

[8] H.S. Dhillon, and A. Mitra, "A Reduced-Bit Multiplication Algorithm for Digital Arithmetic," International Journal of Computational and Mathematical Sciences, pp.64-69, 2008,

[9] P. Mehta, and D. Gawali, "Conventional versus Vedic Mathematical Method for Hardware Implementation of a Multiplier," Proc. Int Conf. on Advances in Computing, Control, and Telecommunication Technologies, Trivandrum, Kerala, India, pp.640-642, 2009.

[10] P. Nair, D. Paranji, and S.S. Rathod, "VLSI Implementation of Matrix-Diagonal Method of Binary Multiplication," Proc. of SPIT-IEEE Colloquium and Int Conf., Mumbai, India, pp.55-58, 2008.

[11] M. Ramalatha, K.D. Dayalan, P. Dharani, ,and S.D. Priya, "High Speed Energy Efficient ALU Design using Vedic Multiplication Technique," Lebanon , pp. 600-603, 2009.

[12] H. Thapliyal, and M.B. Srinivas, "VLSI Implementation of RSA Encryption System Using Ancient Indian Vedic Mathematics," Proc. of SPIE VLSI Circuits and Systems II, pp.888-892, 2005.

[13] H.D. Tiwari, G. Gankhuyag, C.M. Kim, and Y.B. Cho, "Multiplier Design Based on Ancient Indian Vedic Mathematics", Proc. Int SoC Design Conf., pp.65-68. 2008.

[14] P.D. Chidgupkar, and M.T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing," Global J. of Engg. Edu., vol.8, no. 2, 2004.

[15] H. Thapliyal and M. B. Srinivas, "High Speed Efficient N×N Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics," Enformatika Trans., vol. 2, pp. 225–228, Dec. 2004.

[16] R. Pushpangadan, V. Sukumaran, R.Innocent, D. Sasikumar, and V. Sundar, "High Speed Vedic Multiplier for Digital Signal Processors," IETE Journal of Research, vol.55, pp.282-286, 2009.

[17] V.A. Pedroni, "Circuit Design with VHDL," 2008.

[18] 'Xilinx ISE User Manual', Xilinx Inc, USA, 2007

[19] Prabha S., Kasliwal, B.P. Patil and D.K. Gautam, "Performance Evaluation of Squaring Operation by Vedic Mathematics", IETE Journal of Research, vol.57, Issue 1, Jan-Feb 2011.

[20] Devika, K. Sethi and R.Panda, "Vedic Mathematics Based Multiply Accumulate Unit," 2011 International Conference on Computational Intelligence and Communication Systems, CICN 2011, pp.754-757, Nov. 2011.

[21] http://www.scribd.com/doc/29045484/A digital multiplier architecture using Urdva Tiryagbhyam sutra of Vedic mathematics

[22] http;//webspace.utexas.edu/hd3496/www/Downloads/ DhiMit_IC3_2007.pdf

AUTHORS PROFILE

Mr. Kabiraj Sethi is presently a Senior Faculty in the Department of Electronics and Telecommunication Engineering, VSS University of Technology Burla. His area of research interests includes – VLSI and Digital signal processing.

Dr. Rutuparna Panda was born in 1963. He received B.Sc Engg. and M. Sc. Engg. degrees from UCE Burla in 1985 and 1988, respectively. He obtained Ph.D.(Engg) degree from IIT,Kharagpur in 1998. He is currently a Professor in the Department of Electronics and Telecommunication Engineering, VSS University of Technology Burla. He has guided 24 M.Tech Theses and 4 Ph.D. Theses. He has over 70 papers in International/National Journals and conferences. His area of research interests includes – Bioinformatics, Biomedical Image Fusion, Digital signal/image processing, VLSI Signal Processing.