European Journal of Information Technologies and Computer Science
www.ej-compute.org

Hardware Implementation of a High Efficiency and High-Speed Squaring Architecture

Shiva Maleki Varnosfaderani, Bahram Rashidi, and Mohammad Alhawari

ABSTRACT

This paper presents the design of a high speed and simple squaring structure based on half adder and full adder. The proposed architecture consists of three main steps; simplification of the squaring structure, calculation and transferring of the carry bit to the next part, and finally, applying the modified Wallace Tree Adder to calculate the summation of the products. The proposed squaring architecture is formed only by 13 half adders and 20 full adders for 8-bit squaring which has the lowest complexity compared to other works. The proposed structure is modeled by using Field-Programmable Gate Array (FPGA) and has been successfully synthesized and implemented with Xilinx Spartan-6 and Virtex-4 FPGA. Simulation results show that the proposed structure has high speed and excellent performance with low power consumption while having the lowest propagation delay (3.25 ns) and acceptable hardware utilization compared to existing squaring models. The gate-level design of 8-bit squaring is implemented using Cadence layout tools in 65 nm CMOS technology which has a total area of 0.095 mm^2.

Keywords: Field-Programmable Gate Array (FPGA), full adder (FA), gate-level design, half adder (HA), Wallace Tree Adder, Xilinx Spartan-6, Xilinx Virtex-4, 65 nm CMOS technology, 8-bit squaring.

Published Online: August 4, 2022

ISSN: 2736-5492

DOI :10.24018/ejcompute.2022.2.4.71

S. M. Varnosfaderani *
Department of Electrical and Computer Engineering, Wayne State University, Detroit, USA.
(e-mail: gy4030@wayne.edu)
B. Rashidi
Department of Electrical and Computer Engineering, University of Ayatollah Ozma Broujerdi, Iran.
M. Alhawari
Department of Electrical and Computer Engineering, Wayne State University, Detroit, USA.

**Corresponding Author*

I. Introduction

The drastic growth in digital technology and the expansion of digital signal processing (DSP) applications increase the demand for high-speed processing, which is possible by the high throughput arithmetic operations [1]. Arithmetic Logic Units (ALUs) are the vital blocks of processors that perform various arithmetic operations such as addition, subtraction, division, multiplication, and squaring [2].

The squaring algorithm is one of the fundamental and essential functions which is applied in digital algorithms. Accordingly, presenting the high-speed structure with low computations and high performance can have a profound impact on the hardware implementation of the digital signal and image processing algorithms. The evaluation of applications increases the requests for using the processors along with square and cube functions with higher speed [3]. Generally, because of resource sharing, generalization and simply convenience, the standard multiplier structure is used to implement squaring. However, due to generating a large number of duplicated partial products, the implementation of a specialized squaring method is more efficient than applying the multiplication techniques for squaring [4].

In general, the dedicated squaring can have a higher speed with lower consumption and smaller structure compared to a multiplier. Therefore, they have better performance in fixed-point function evaluations and various floating-point arithmetic computations [5]–[8].

In this research, the squaring structure consisting of three main steps is proposed which is based on HA and FA made by XOR–MUX that has fewer transistors, lower average delay, and higher speed compared to other FA structures [9]. For calculating the summation of the computed partial products, modified Wallace Tree Adder is used which has a higher speed with lower computations compared to the conventional Wallace Tree structure [10], [11]. The block diagram of the proposed structure for squaring is shown in Fig. 1.

In the following sections, the main squaring structures are reviewed in section II. The proposed structure for 8- bit squaring is explained in section III. The simulation and comparison results are presented in section IV and finally, in section V, the conclusion is provided.

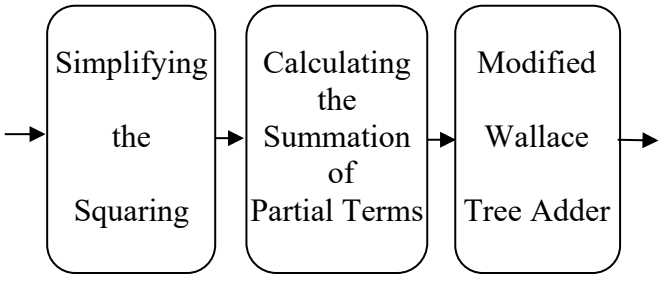

Fig. 1. The block diagram of the proposed structure.

II. Related Work

In [12], the presented squaring and multiplication structures are based on ancient Indian Vedic Mathematics Sutras. In common Vedic multiplier architectures, the partial product terms are calculated in parallel form, and afterward, the calculated terms are added to get the final result. However, in the proposed structures in [12]–[19], all the partial products are adjusted by applying the concatenation operation. Then, instead of applying two adders at different steps, the carry- save adder (CSA) is used. In the n-bit squaring architecture, for computing the partial products, one multiplier of n/2 bits along with two squaring circuits of n/2 bits is applied. Finally, for calculating the final output, all the computed partial products are adjusted accordingly, and their summation is calculated using the single CSA.

In [20], the squaring structure is based on Anurupya Sutra of Vedic Mathematics. In this structure, the considered level base value is less than the decimal input number which the deficit is calculated using subtracting the level base value from the number.

In [21], the multiplier based on Yavadunam used the bit reduction technique to improve the area and speed of the structure. In another word, reducing the bit numbers feeding to the multiplier decreased the number of interconnects and part leading to decrease the delay and area, and subsequently, increasing the speed. The proposed squaring structure in [1] is based on modified Yavadunam which to optimize the area and speed, the method of weight reduction is employed to reduce the N-bit size to N-1 bits.

The presented squaring structure in [22] is inspired by the Peasants method which is the ancient Egyptian method of multiplication. In this method, two numbers are determined in the left and right sides. Then, the multiplication of each bit on the right side in a specific row and the last bit (LSB) of the number of the same row is calculated. If the number on the left side is even, the multiplication of this bit will be automatically canceled by making zero all elements of that row on the right side which simplifies the computations.

In [23], the presented squaring architecture is based on a novel multi-precision squaring method named Complete Hybrid Karatsuba Squaring (CHKS0. In this method, the subtractive Karatsuba is the basic part of the structure. Also, to make shorter the input, the adopt Vector-Like (VL) squaring is used which has a slight advantage compared with traditional product-scanning squaring.

III. Proposed Structure

The proposed squaring structure consists of three main steps; simplification the squaring, calculating the summation of partial products and transferring the carry bits to the next part, which is done in two steps, and applying the modified Wallace Tree Adder to calculate the summation of the provided products.

In the first step, some partial products are replaced by similar terms. This step makes the structure of squaring simpler. The second step consists of two main parts and the main goal of these parts is to calculate the carry bits according to two equations and then transfer them to the next part. In this step, many partial products are removed, which directly increases the speed of the calculation. Finally, for adding the partial products, modified Wallace Tree Adder is used, which is faster than the conventional Wallace Tree Adders.

A. Simplification of the Squaring

In this step, firstly, the expanded square of the 8-bit number is determined and then, the $a_n a_m$ terms are replaced by the $a_m a_n$ terms (n is bigger than m). Hence, similar partial products are specified and replaced. This step has a great effect on reducing the number of partial products in the next step.

B. Calculating the Summation of Partial Terms

This step consists of two main parts. In the first part, the carry bits of some partial terms are calculated according to (1a) and then transferred to the next stage [22]. In the second part, the summation of some terms is computed and simplified according to (1b). The results of the simplification of the squaring structure according to steps 1 and 2 are shown in Fig. 2. In this step, the number of partial products significantly reduces from 64 to 37. As a result, the number of FA and HA which will be applied at the next step for calculating the final output is reduced. Hence, this step plays a critical role in improving the speed of the proposed structure.

$$A = a^2 = a_7a_6a_5a_4a_3a_2a_1a_0 \times a_7a_6a_5a_4a_3a_2a_1a_0$$

$$\begin{array}{cccccccccccccccc}
 & & & & & & & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & \\
 & & & & & & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & \\
 & & & & & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & 0 & \\
 & & & & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & 0 & 0 & \\
 & & & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & 0 & 0 & 0 & \\
 & & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & 0 & 0 & 0 & 0 & \\
 & a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
a_7a_0 & a_6a_0 & a_5a_0 & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
\hline
a_7a_6 & a_7\overline{a_6} & a_6a_5 & a_6\overline{a_5} & a_5a_4 & a_5\overline{a_4} & a_4a_3 & a_4\overline{a_3} & a_3a_2 & a_3\overline{a_2} & a_2a_1 & a_3a_0 & a_2a_0 & a_1\overline{a_0} & 0 & a_0 \\
 & a_7a_5 & a_7a_4 & a_7a_3 & a_7a_2 & a_7a_1 & a_6a_1 & a_6a_0 & a_5a_0 & a_3a_1 & a_2\overline{a_1} & a_1a_0 & & & & \\
 & & a_6a_4 & a_6a_3 & a_5a_3 & a_7a_0 & a_5a_1 & a_4a_1 & a_4a_0 & & & & & & & \\
 & & & a_6a_2 & a_5a_2 & a_4a_2 & & & & & & & & & &
\end{array}$$

Fig. 2. The results after applying steps 1 and 2.

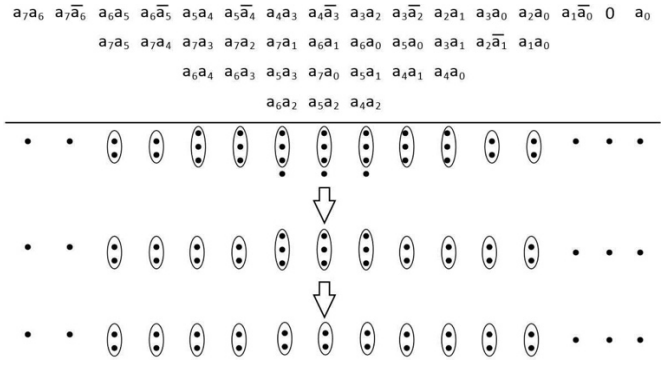

Fig. 3. Modified Wallace Tree Adder for proposed 8-bit squaring.

$$\begin{array}{cc}
\begin{array}{r} a_na_m \\ + \; a_na_m \\ \hline a_na_m \;\; 0 \end{array} & \begin{array}{r} a_na_n \\ + \; a_na_m \\ \hline a_na_m \;\; \overline{a_n}a_m \end{array} \\
\text{(a)} & \text{(b)}
\end{array} \qquad (1)$$

European Journal of Information Technologies and Computer Science
www.ej-compute.org

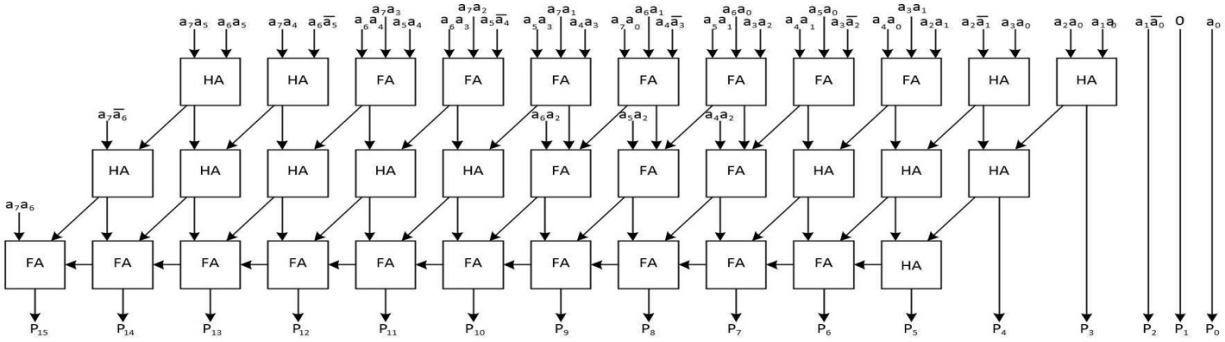

Fig. 4. The proposed 8-bit squaring structure.

C. *Modified Wallace Tree Adder*

To calculate the summation of the extracted terms in step 2 and calculating the final output, modified Wallace Tree Adder is used. These types of adders are based on summing the partial product bits in parallel which apply a tree to carry save adder. The modified Wallace Tree Adder has a higher speed with lower computations compared to the Wallace Tree structure used in [10], [11]. According to this structure, the summation of partial products for every three rows is calculated in each step. In this case, at first, each column of the first three rows is grouped according to the number of partial products. As a result, if the column consists of three partial products, the FA is used, otherwise, the HA is used for calculating the summation of grouped partial products. Afterward, the results of the sum and the carry bits are transferred to the next rows which are considered as one row in the next step. Again, three rows of partial products are considered, and the columns are grouped according to the mentioned way. This process is repeated until only two rows of partial products remain. Finally, the carry ripple adder is used to calculate the summation of these two last rows. This process is shown in Fig. 3. According to this figure, after two-step grouping, only two rows of partial products remain. The proposed squaring structure for an 8-bit number is shown in Fig. 4. As it is shown, by using this structure, only 20 FAs and 13 HAs are required for 8-bit squaring. As a result, the proposed method has a simple structure and high speed with low power consumption.

IV. Simulation Results

The proposed structure for 8-bit is synthesized and implemented on the Xilinx FPGA Spartan-6 family, device XC6SLX4-TQG144 with a speed grade of -3 and Virtex-4 family, device XC4VLX15-SF363 with a speed grade of -12. The comparison results are presented in Table I. A sample simulation waveform of the squaring operation is illustrated in Fig. 5. In this figure, the parameters a and b are input signal and output signal which is the square of parameter a, respectively.

To investigate the performance of the proposed squaring structure and give a comprehensive comparison between this structure and other related works which are implemented on Xilinx FPGA (Virtex-4 family), the propagation delay and hardware utilization of the previous works for 8-bit squaring are provided in Table II. Also, the comparisons of utilization and propagation delay results are shown in Fig. 6 and Fig. 7, respectively. There is a trade-off between area and propagation delay for any digital design. Comparison results show that the proposed structure, besides acceptable hardware utilization, has lower propagation delay compared to other works. Hence, the proposed structure can be used in the systems needing a high-speed operation with low power consumption.

Fig.8 shows the gate-level layout design of the proposed 8-bit squaring structure using Cadence layout tools in 65 nm CMOS technology. The layout design has a total area of 0.095 mm^2 which consists of 20 FAs, 13 HAs, and 34 inverters.

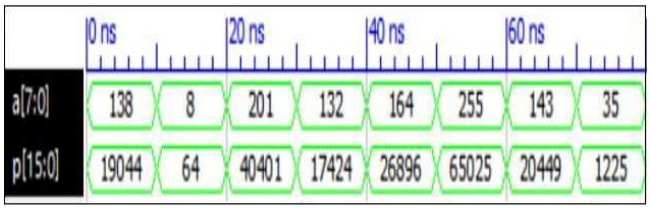

Fig. 5. The sample simulation waveform of 8-bit squaring.

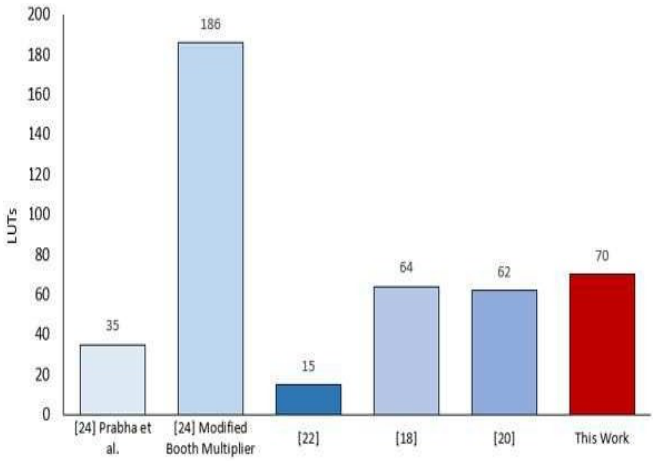

Fig. 6. Comparison of Utilization Device for 8-bit squaring

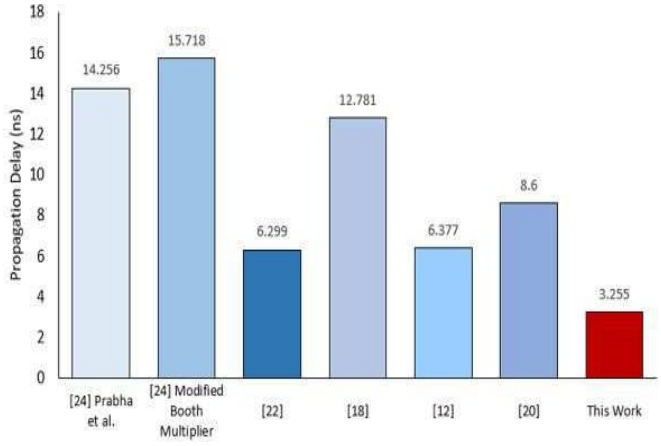

Fig. 7. Comparison of Propagation Delay (ns) for 8-bit squaring.

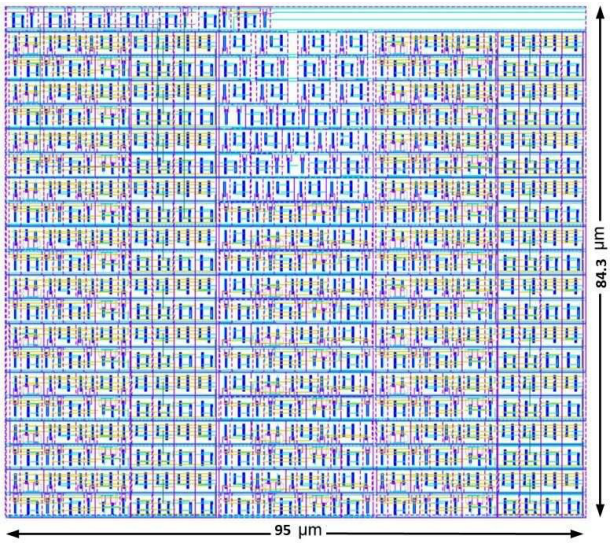

Fig. 8. The gate-level layout design of proposed 8-bit squaring.

TABLE I: DEVICE UTILIZATION SUMMARY AND PROPAGATION DELAY

Device Parameters	Device Utilization		
	4-Input LUTs	Slices	[a] PD (ns)
Virtex-4 XC4VLX15	70	39	3.255
Spartan-6 XC6SLX4	32	0	5.22

[a] Propagation Delay.

TABLE II: COMPARISON OF THE UTILIZATION AND PROPAGATION DELAY

Method	[b]DU and PD	
	LUTs	PD (ns)
Reference [24]	35	14.256
Reference [24]	186	15.718
Reference [22]	15	6.299
Reference [18]	64	12.781
Reference [12]	NA	6.377
Reference [20]	62	8.6
This work	70	3.255

Device Utilization.

V. CONCLUSION

In this research, a high-performance and area-efficient square architecture based on three main steps is presented. The proposed square structure is successfully synthesized and implemented on the Xilinx FPGA Spartan-6 family and Virtex- 4 family. The comparison results indicate that the proposed architecture has the lowest propagation delay compared to previous architecture. Thus, the proposed squaring architecture is suitable for high-performance processors that require intensive computations.

REFERENCES

[1] Deepa A, Marimuthu CN, Murugesan C. An efficient high speed squaring and multiplier architecture using yavadunam sutra and bit reduction technique. *Journal of Physics: Conference Series.* 2020; 1432(1): 012080.

[2] George ML, Tomar GS. Comparative review of floating-point multiplier systems. *International Journal of Hybrid Information Technology*. 2019; 12(2): 21-48.

[3] Ramanammma P, Malashree N. Low power square and cube architectures using Vedic Sutras. *International Journal of Engineering Research and General Science*. 2017; 5(3): 241-8.

[4] Liu Z, Seo H, Kim H. A synthesis of multi-precision multiplication and squaring techniques for 8-bit sensor nodes: state-of-the-art research and future challenges. *Journal of Computer Science and Technology*. 2016; 31(2): 284-99.

[5] Chung K, Kim LS. Area-efficient special function unit for mobile vertex processors. *Electronics Letters*. 2009; 45(16): 826-7.

[6] Donofrio DD, Li X, inventors; Intel Corp, assignee. Enhanced floating-point unit for extended functions. United States patent US 7,676,535. 2010 Mar 9.

[7] Pasca B. Correctly rounded floating-point division for DSP-enabled FPGAs. IEEE, International Conference on Field Programmable Logic and Applications (FPL), 2012: 249-254.

[8] De Dinechin F, Pasca B. Designing custom arithmetic data paths with FloPoCo. *IEEE Design & Test of Computers*. 2011; 28(4): 18-27.

[9] Uma R, Dhavachelvan P. Logic optimization using technology independent mux based adders in FPGA. *International Journal of VLSI design & Communication Systems.* 2012; 3(4): 133.

[10] Rashidi B. High performance and low-power finite impulse response filter based on ring topology with modified retiming serial multiplier on FPGA. *IET Signal Processing*. 2013; 7(8): 743-753.

[11] Mohanty PS. Design and implementation of faster and low power multipliers. 2009.

[12] Sharma R, Kaur M, Singh G. Design and FPGA implementation of optimized 32-bit Vedic multiplier and square architectures. *2015 International Conference on Industrial Instrumentation and Control (ICIC).* 2015: 960-964.

[13] Poornima M, Patil SK, Shivukumar SK, Sanjay H. Implementation of multiplier using Vedic algorithm. *International Journal of Innovative Technology and Exploring Engineering (IJITEE).* 2013; 2(6): 219-23.

[14] Vaithiyanathan G, Venkatesan K, Sivaramakrishnan S, Siva S, Jayakumar S. Simulation and implementation of Vedic multiplier using VHDL code. *International Journal of Scientific & Engineering Research.* 2013; 4(1).

[15] Bathija RK, Meena RS, Sarkar S, Sahu R. Low power high speed 16x16 bit multiplier using vedic mathematics. International Journal of Computer Applications. 2012; 59(6).

[16] Ramachandran S, Pande KS. Design implementation and performance analysis of an integrated vedic multiplier architecture. *International Journal of Computational Engineering Research*. 2012; 2(3): 697-703.

[17] Kumar GG, Charishma V. Design of high speed vedic multiplier using vedic mathematics techniques. *International Journal of Scientific and Research Publications.* 2012; 2(3): 1.

[18] Sethi K, Panda R. An improved Squaring circuit for binary numbers. *International Journal of Advance Computer Science and Applications*. 2012; 3(2).

[19] Siddhi AK. Hardware Implementation of 16* 16 bit Multiplier and Square using Vedic Mathematics.

[20] Reddy BN. Design and implementation of high performance and area efficient square architecture using Vedic Mathematics. *Analog Integrated Circuits and Signal Processing*. 2020; 102(3): 501-6.

[21] Wang JS, Kuo CN, Yang TH. Low-power fixed-width array multipliers. *In Proceedings of the 2004 International Symposium on Low Power Electronics and Design.* 2004: 307-312.

[22] Mishra S, Dhakad SK. A High Speed and device efficient FPGA based Squaring circuit. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering.* 2013; 2(11): 5647-54.

[23] Zhang W, Lin D, Zhang H, Zhou X, Gao Y, Chen C. A lightweight multi-precision squaring on embedded processors for ECC. In2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). 2018: 1014-1019.

[24] Kasliwal PS, Patil BP, Gautam DK. Performance evaluation of squaring operation by Vedic mathematics. *IETE journal of Research.* 2011; 57(1): 39-41.