The Plum-Blossom Product Method of Large Digit Multiplication and Its Application to Computer Science

Yongwen Zhu
School of Mathematics and Information Science,
Yantai University
Yantai City, 264005, CN

ABSTRACT
In this paper, we introduce a novel method for multiplication of large integers called plum-blossom product method. This is not only an effective mental method of multiplication, but also can be used to computer science. Compared with the rapid multiplication method of Shi Fengshou and the Indian Vedic algorithm, the plum-blossom product method is more systematic, suitable for computing the multiplication of any multi-digit numbers in mind, and it has less formulae so that it is very simple and easy to learn. For the aspect of application to compute science, the corresponding multiplication algorithm with plum-blossom products is given for integers in the binary number system. Furthermore, an effective multiplier with the plum-blossom product method is designed, which can directly calculate the product of two 27-bit binary numbers. When associated with other methods such as Karatsuba algorithm, it may be able to compute product of any two large integers.

General Terms
Algorithm, Multiplier, Arithmetic Logic Unit

Keywords
Multiplier; multiplication; large integer; carry; plum-blossom product.

1. INTRODUCTION
Studying the fast multiplication method of multi-digit numbers is not only of pedagogical and psychological significance [1][2], but also of great scientific significance [3]. In the face of large number multiplication, traditional multipliers are unable to adapt to the rapid increase of the number of digits needed. Larger bit length multipliers have an important implications on the performance of computer applications in many aspects such as cryptography, digital signal processing (DSP) and image processing, and so on [3]. For large integer multiplication, novel optimization design is required, because previous methods, such as textbook multiplication, are not suitable for large numbers. In cryptography, for example in lattice-based and completely homomorphic encryption (FHE) schemes, it usually needs parameter bit lengths of up to millions of bits. In [3], hardware complexity analysis showed that NTT-Karatsuba-Schoolbook combination is the most appropriate for large multipliers larger than 16,384 bits, where the so-called NTT means the fast number theory transform, and the Karatsuba method is a multiplication method with segmented numbers. An other method to multiply is the modular multiplication, see [5][6]. In recent years, there has been a lot of research on applying Indian Vedic mental arithmetic to design more efficient multipliers [7-17]. Vedic arithmetic works well for some special types of numbers, but it is difficult to multiply arbitrary numbers [7]. Comparatively speaking, the fast multiplication method of Shi Fengshou is a relatively systematic mental arithmetic theory suitable for any two integers [18], but it requires to remember too many carry formulae, and it is also difficult to really realize on computer. In [19], the author introduced an other systematic mental multiplication method called scissor product method, which is superior to Shi's method as well as the Vedic algorithm. As an improvement of scissor product method, the concept of the plum-blossom product will be introduced in the present paper. This note can be used to design a more efficient multiplier, which incorporating into other methods such as FTT and/or Karatsuba algorithm makes it possible to compute the product of any two large integers.

2. THE PLUM-BLOSSOM PRODUCT METHOD

The basic formula of fast multiplication
First we introduce a notation. Let

$$\left\langle \begin{matrix} a_1 & a_2 & \cdots & a_k \\ b_1 & b_2 & \cdots & b_k \end{matrix} \right\rangle = a_1 \times b_k + a_2 \times b_{k-1} + \cdots + a_k \times b_1$$

which we called the cross product with length k. Multi-digit multiplication requires the following general formula [7][8][18][19], which is called the basic formula of fast multiplication. It boils the product of two n-digits or n-node segmented numbers into a series of cross products:

$$\left\langle \begin{matrix} a_1 a_2 \cdots a_n \\ b_1 b_2 \cdots b_n \end{matrix} \right\rangle = \left(\left\langle \begin{matrix} a_1 \\ b_1 \end{matrix} \right\rangle, \left\langle \begin{matrix} a_1 & a_2 \\ b_1 & b_2 \end{matrix} \right\rangle, \cdots, \left\langle \begin{matrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{matrix} \right\rangle, \left\langle \begin{matrix} a_2 & \cdots & a_n \\ b_2 & \cdots & b_n \end{matrix} \right\rangle, \cdots, \left\langle \begin{matrix} a_n \\ b_n \end{matrix} \right\rangle \right)$$

The concept of plum-blossom products
As an improvement of fast multiplication method, the scissor product is introduced in [19], which was defined as $a \wedge b = a \times b - (\min(a,b)-1) \times 10$ for two digits a and b. However, sometimes the absolute value of the the scissor product is too large such as $5 \wedge 5 = 25 - 40 = -15$, which seriously affects the speed of mental arithmetic. The note of plum-blossom products is a further optimization and improvement of scissor products. For any two decimal units, their plum-blossom product is defined as an integer between -6 and 3, being equal to the ones of their ordinary product, or equal to this ones minus 10. The plum-blossom product of two digits a and b is denoted as $a \otimes b$. For example, since $3 \times 7 = 21$, and the ones is 1 so that $3 \otimes 7 = 1$; similarly, from $4 \times 7 = 28$ it follows that $4 \otimes 7 = 8 - 10 = -2$. It is easy to rewrite the ordinary 9×9 multiplication table so as to obtain the plum-blossom product table, Table 1.

Table 1. The plum-blossom product table

$\otimes$	1	2	3	4	5	6	7	8	9
1	1	2	3	-6	-5	-4	-3	-2	-1
2		-6	-4	-2	0	2	-6	-4	-2
3			-1	2	-5	-2	1	-6	-3
4				-4	0	-6	-2	2	-4
5					-5	0	-5	0	-5
6						-4	2	-2	-6
7							-1	-4	3
8								-6	2
9									1

The Plum-Blossom Product Method

The product of any two numbers minus their plum-blossom product must be an integer multiple of 10. We call this multiple the carry corresponding to the plum-blossom product. If the carry corresponding to $a \otimes b$ is denoted as $J(a \otimes b)$, then one has the following formula:

$$a \times b = (J(a \otimes b),\ a \otimes b)$$

For example, since $3 \times 7 - 3 \otimes 7 = 20$, the carry corresponding to $3 \otimes 7$ is 2, that is, $J(3 \otimes 7) = 2$. Similarly, from $4 \times 7 - 4 \otimes 7 = 28 - (-2) = 30$, it follows that $J(4 \otimes 7) = 3$. Suppose $a \le b$. Then by Table 1, one has the following carry formula:

$$J(a \otimes b) = \begin{cases} a, & if\ (a = 1 \vee b = 9) \wedge b - a \ge 3; \\ a, & if\ b - a \ge 5; \\ a - 2, & if\ (3 \le a \le b \le 7) \wedge b - a \le 1; \\ a - 1, & otherwise. \end{cases}$$

Associated with the basic formula of fast multiplication, this carry formula provides a novel method of multiplication, called the plum-blossom product method of multiplication. The following is a practical examples of using this method to mental multiplication.

An Example of Mental Multiplication

To compute the product 456×789 in mind, one can first use the basic formula of fast multiplication and then by the above carrying method transforms all involved cross products into cross plum-blossom products. Note that $J(6 \otimes 7) = 6 - 2$, $J(4 \otimes 9) = 4$, $J(5 \otimes 9) = 5$ and etc. The whole procedure of mental calculation is as follows:

$$\left\langle \begin{matrix} 456 \\ 789 \end{matrix} \right\rangle \to \left(\left\langle \begin{matrix} 4 \\ 7 \end{matrix} \right\rangle, \left\langle \begin{matrix} 4 & 5 \\ 7 & 8 \end{matrix} \right\rangle, \left\langle \begin{matrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \right\rangle, \left\langle \begin{matrix} 5 & 6 \\ 8 & 9 \end{matrix} \right\rangle, \left\langle \begin{matrix} 6 \\ 9 \end{matrix} \right\rangle \right) \to$$

$$\begin{pmatrix} 4 \times 7 + (4 + 5 - 1 - 1) \\ 4 \otimes 8 + 5 \otimes 7 + (4 + 5 + 6 - 1 - 1 - 1 + 1 - 1) \\ 4 \otimes 9 + 5 \otimes 8 + 6 \otimes 7 + (5 + 6 - 1 - 1 + 1) \\ 5 \otimes 9 + 6 \otimes 8 \\ 6 \times 9 \end{pmatrix} \to \begin{pmatrix} 35 \\ 9 \\ 8 \\ -7 \\ 54 \end{pmatrix} \to \begin{pmatrix} 35 \\ 9 \\ 8 \\ -2 \\ 4 \end{pmatrix}$$

$$\to 359784.$$

Therefore, the answer is that $456 \times 789 = 359784$. Note that, The above calculation process can be done in the mind thoroughly. A lot of calculation practice shows that when using the plum-blossom product method to calculate multi-digit multiplication, the intermediate process is basically one digit addition and subtraction operation so that the mental calculation is very easy to implement.

3. THE APPLICATION OF THE PLUM-BLOSSOM PRODUCT MEHTOD TO COMPUTER SCIENCE

The Plum-Blossom Product Method in Binary Number System

It is easy to define naturally plum-blossom products for 3-bit binary numbers so that the following corresponding plum-blossom product table, Table 2 is obtained.

Table 2. The plum-blossom product table of 3-bit binary numbers

$\otimes$	001	010	011	100	101	110	111
001	001	010	011	-100	-011	-010	-001
010		-100	-010	000	010	-100	-010
011			001	-100	-001	010	-011
100				000	-100	000	-100
101					001	-010	011
110						-100	010
111							001

For two 3-bit binary numbers a and b, let $a \times b = (J(a \otimes b), a \otimes b)$. Then it follows that

$$J(a \otimes b) = \begin{cases} a, & if\ (a = 001 \vee b = 111) \wedge b - a \ge 011; \\ a, & if\ b - a \ge 100; \\ a - 2, & if\ a = b = 011 \vee 100 \vee 101; \\ a - 1, & otherwise. \end{cases}$$

The plum-blossom product method may be used for multiplication of segmented binary numbers with each segmented block length 3. For convenience, we use octal number system for examples so that each one digit represents 3 bits of the binary numbers system. For example, in the octal number system, $01234567 \times 7 = 11111101$, the mental procedure is as follows: $(1,1,1,1,0+1,0+1,0,1) \to 11111101$. Similarly, one has $45 \times 67 = 3763$, the mental procedure is as follows: $(3, 4+4, -4-2+4, 3) \to 3763$. Similarly, one may get $345 \times 543 = 236617$, the mental procedure is as follows: $(2,4-1,6,7-1,0+1,7) \to 236617$. Similarly, one may obtain that $376 \times 456 = 225644$, the mental procedure is as follows: $(2,2,4+1,6,4,4) \to 225644$. As an other example, one may get $1234 \times 4567 = 6131204$ in the octal number system, which may be obtained directly in mind with the plum-blossom product method. At last, one may obtain that $4671 \times 7534 = 45252574$, the mental procedure is as follows: $(4,4+1,2,4+1,3-1,6-1,7,4) \to 24252574$.

The Plum-Blossom Product Algorithm for Multiplication

Algorithm 1 is Comba multiplication algorithm [3],which in effect corresponds to the basic formula of fast multiplication.

Algorithm 1: Comba Multiplication

Input: *n*-bit integers *a* and *b*

Output: $z = a \times b$

1: **for** i in 0 to $(2n-2)$ **do**
2: **if** $i < n$ **then**
3: $pp_i = \sum_{k=0}^{i} (a_k \times b_{i-k})$
4: **else**
5: $pp_i = \sum_{k=i-n+1}^{n-1} (a_k \times b_{i-k})$
6: **end if**
7: **end for**
8: $z = \sum_{i=0}^{2n-2} (pp_i << 2^i)$
return Z

The numbers are divided into segmented numbers with each block length 3 so that the above algorithm may be rewritten as Algorithm 2.

Algorithm 2: Comba Multiplication with 3n-bits

Input: 3*n*-bit integers *a* and *b*

Output: $z = a \times b$

1: **for** i in 0 to $(2n-2)$ **do**
2: **if** $i < n$ **then**
3:
4: **else**
5:
6: **end if**
7: **end for**
8: $z = \sum_{i=0}^{2n-2} (pp_i << 2^{3i})$
return Z

Algorithm 3 is used to calculate the plum-blossom product of two 3-bit binary numbers by means of Algorithm 1 so that it saves half the computation.

Algorithm 3: Plum-blossom Product in 3-bits

Input: 3-bit integers *a* and *b*

Output: $z = a \otimes b$

1: **for** i in 0 to 2 **do**
2: $pp_i = \sum_{k=0}^{i} (a_k \times b_{i-k})$
7: **end for**
8: $z = \sum_{i=0}^{2} (pp_i << 2^i)$
9: **if** $z_2 = 1$ **then**
10: $z \leftarrow z - 1{,}000$
11: **end if**
return Z

According to the carry formula, one may obtain Algorithm 4, which is used to calculate the carry corresponding to the plum-blossom product of two 3-bit binary numbers.

Furthermore, the algorithm 5 is obtained, which is used to compute the plum-blossom multiplication.

Algorithm 4: Carry Corresponding to the Plum-blossom Product in 3-bits

Input: 3-bit integers *a* and *b*

Output: 3-bit integer $c = J(a,b)$

1: $m \leftarrow \min(a,b)$
2: $M \leftarrow \max(a,b)$
3: $c \leftarrow m-1$
4: **if** $m = M = (011 \vee 100 \vee 101)$ **then**
5: $c \leftarrow c-1$
6: **else**
$(m = 010 \wedge M = 110)$
8: $c \leftarrow c+1$
9: **end if**
10: **end if**
return C

The Design of the Multiplier by Means of Plum-Blossom Products

According to Algorithm 3, one may design the calculation unit with two 3-bit binary numbers a and b as input and a 3-bit binary number $a \otimes b$ as the output, whose logic circuit diagram is as in Fig. 1. This arithmetic logic unit may be

simplified to Fig. 2. By Algorithm 4, one may design the calculation unit with two 3-bit binary numbers a and b as input and a 3-bit binary number $J(a,b)$ as its output, which represents the carry corresponding to the plum-blossom product of a and b. This unit may be simplified into that as in Fig. 3. Associated with the arithmetic logic units of computing the plum-blossom product and carry for two 3-bit binary numbers, one may use Algorithm 5 to construct the multiplier with plum-blossom products, which may be used to calculate the product of any two 27-bit binary numbers. This architecture is as in Figs 4 and 5. Fig. 4 is the local diagram of the i'th vertical partial product of the multiplier with $i > n$.

If $i \le n$, the diagram would be slightly different. In Fig. 5, the unit T realizes the function of adding the lowest 3-bit of the PP_{i+1} and the highest 3-bit of PP_i and justifying the lowest 3-bit of the sum into non-negative integers and outputting PP_i as a part of the final product and outputting the highest 3-bit (with a sign) as the carry C_i as well as the input of the higher position. Since each PP_i consists of 3 bits, the final result consists of at most 36 bits.

Algorithm 5: Plum-blossom Multiplication

Input: 3*n*-bit integers *a* and *b*

Output: $z = a \times b$

1: **for** i in 0 to $(2n-2)$ **do**

2: **if** $i < n$ **then**

3: $pp_i = \sum_{k=0}^{i} (a_{3k+2}a_{3k+1}a_{3k} \otimes b_{3(i-k)+2}b_{3(i-k)+1}b_{3(i-k)})$

4: $pp_i = pp_i + \sum_{k=0}^{i-1} J(a_{3k+2}a_{3k+1}a_{3k}, b_{3(i-k-1)+2}b_{3(i-k-1)+1}b_{3(i-k-1)})$

5: **else**

4: $pp_i = \sum_{k=i-n+1}^{n-1} (a_{3k+2}a_{3k+1}a_{3k} \otimes b_{3(i-k)+2}b_{3(i-k)+1}b_{3(i-k)})$

5: $pp_i = pp_i + \sum_{k=i-n}^{n-1} J(a_{3k+2}a_{3k+1}a_{3k}, b_{3(i-k-1)+2}b_{3(i-k-1)+1}b_{3(i-k-1)})$

6: **end if**

7: **end for**

8: $z = \sum_{i=0}^{2n-2} (pp_i << 2^{3i})$

return Z

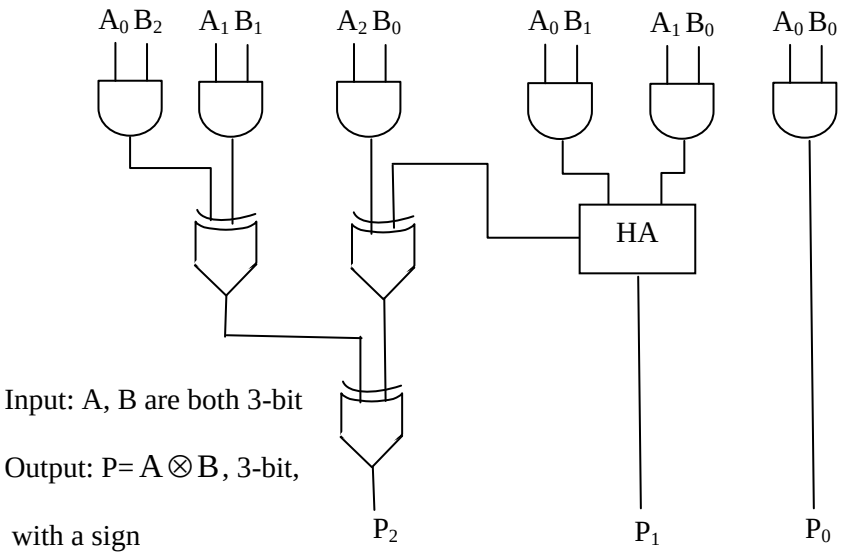

Fig 1: Logical circuit diagram for the plum-blossom product of 3-bit binary numbers

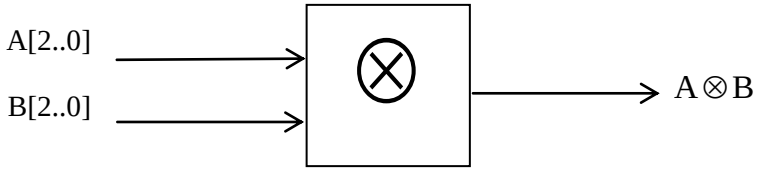

Fig 2. Arithmetic logic unit schematic diagram of the plum-blossom product of 3-bit binary numbers

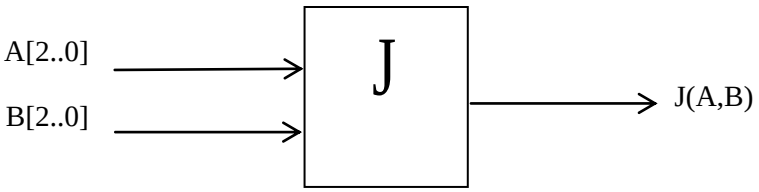

Fig 3. Arithmetic logic unit schematic diagram for calculating the carry corresponding to the plum-blossom product of 3-bit binary numbers

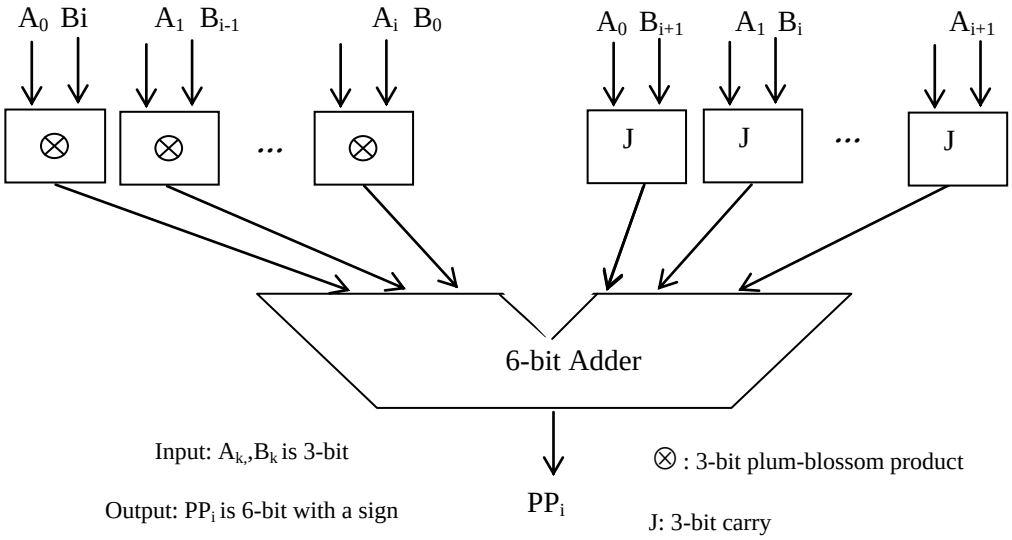

Fig 4. Architecture of the Plum-blossom Product Multiplier (Local)

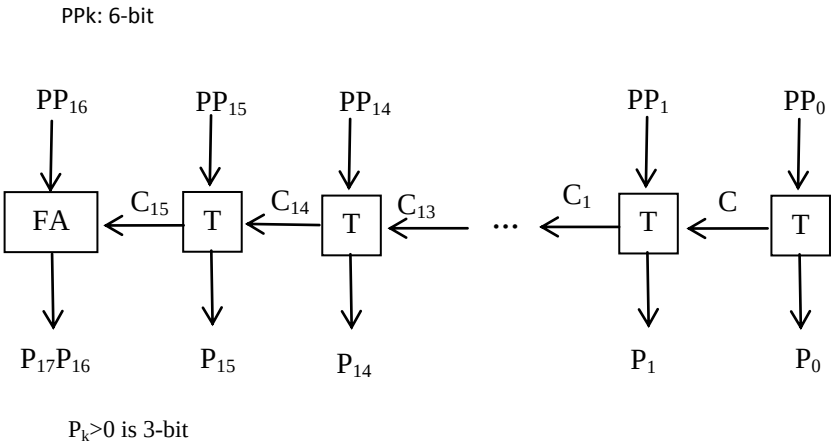

Fig 5. Architecture of the Plum-blossom Product Multiplier (Global)

4. CONCLUSION AND DISCUSSION

From the point of view of the pure mathematics, the plum-blossom product multiplication method under the decimal system is an effective fast mental multiplication method, which does not require the numbers to multiply to have the particularity, and for the product of any two multi-digit numbers within the ten bits can almost directly tell the answer. This method is superior to Vedic mind algorithm from the perspective of systematization and universality, and also superior to the rapid multiplication method proposed by Shi Fengshou since the latter has too more formulae for carry required to be remembered. When faced with multi-digit number multiplication, Shi's method still produces many carries on each vertical partial product, which is why it does not perform well when implemented on computers. However, the plum-blossom product is different, because it may be negative so that the local product sum may cancel out most of time, and that's the secret of the plum-blossom product method.

From the perspective of computer science, the multiplier with plum-blossom products can directly compute the product of two 27-bit binary numbers, which is equivalent to computing the product of over 100 million numbers in decimal since $2^{27} = 8^9 = 134{,}217{,}728$. If incorporating into other methods such as Karatsuba multiplication and/or FFT [3][8], it may calculate the product of any two long integers. The multiplier with plum-blossom products has at least the following advantages. First, The number of digits of the longitudinal partial product is limited to 6 , making the addition of the

partial products relatively easy. Second, the plum-blossom product of two 3-bit numbers only needs to calculate the lower 3 bits of the usual product, which is almost half of the calculation amount of ordinary multiplication method. Third, carry is very simple, just needs to compare the size of the numbers. At last, the longitudinal partial products can be calculated in parallel, thus saving time.

Why is it 27-bit binary numbers? Let's look at it in a nutshell. Suppose that it can calculate $3n$ digits. If each three digits is in a section, both of the multiplicand and multiplier can be divided into n sections. When PP_{n+1} is calculated, there are n carries, each of which has a maximum value of 110, equivalent to 6 in decimal number system, and by the plum-blossom product table there is only one way to get the maximum carry 6, which is $110 \otimes 110 = 001$. Now, in decimal system, $PP_{n+1} = 1 \times (n-1) + 6 \times n = 7n - 1$. To ensure no more than 6 bits in binary system, it must be $7n - 1 \le 63$, that is, $n \le 9$. With the aid of the plum-blossom product table it is verified that $n = 9$ is suitable for other cases. Therefore, $3n = 27$.

In a word, the plum-blossom product multiplication method is a very effective mental multiplication method for multi-digit numbers, and the above multiplier based on the plum-blossom products is an efficient multiplier which is suitable for large digit multiplication.

5. ACKNOWLEDGMENTS

The author is very grateful to the editors and reviewers for their helpful suggestions.

6. REFERENCES

[1] MA D Z., He S. B., Sun K. H. 2021. A Modified Multivariable Complexity Measure Algorithm and Its Application for Identifying Mental Arithmetic Task. Entropy, 23 (8): 931.

[2] GOWERS W. T. 2007. Mathematics, memory, and mental arithmetic. Mathematical knowledge, 33--58, Oxford Univ. Press, Oxford.

[3] RAFFERTY C., O'Neill M., HANLEY N. 2017. Evaluation of large integer multiplication methods on hardware. IEEE Trans. Comput. 66(8): 1369--1382.

[4] SAN I., AT N. 2012. "On increasing the computational efficiency of long integer multiplication on FPGA". In Proc. 11th IEEE Int. Conf. Trust Secur. Privacy Comput. Commun., pp. 1149-1154.

[5] LIU W., NI J., LIU Z., et al. 2019. Optimized Modular Multiplication for Supersingular Isogeny Diffie-Hellman. IEEE Transactions on Computers, 68(8): 1249-1255. doi: 10.1109/TC.2019.2899847.

[6] GU Z., LI S. 2019. A Generalized RNS Mclaughlin Modular Multiplication with Non-Coprime Moduli Sets. IEEE Transactions on Computers, 68(11):1689-1696. doi: 10.1109/TC.2019.2917433.

[7] MATHUR M., AARNAV. 2017. Demystification of Vedic Multiplication Algorithm. American Journal of Computational Mathematics, 07(1):94-101.

[8] KAVITA U. G. 2013. Performance Analysis of Various Vedic Techniques for Multiplication. International Journal of Engineering Trends & Technology, 4(3): 231-234.

[9] RAMALATHA M., DAYALAN K D., DHARANI P., et al. 2009. "High speed energy efficient ALU design using Vedic multiplication techniques". In International Conference on Advances in Computational Tools for Engineering Applications, IEEE.

[10] PARAMASIVAM, SABEENIAN R. 2010. "An efficient bit reduction binary multiplication algorithm using Vedic methods". In Advance Computing Conference, IEEE.

[11] KAYAL D., MOSTAFA P., DANDAPAT A., et al. 2014. Design of High Performance 8 bit Multiplier using Vedic Multiplication Algorithm with McCMOS Technique. Journal of Signal Processing Systems, 76(1):1-9.

[12] GURUMRUTHY K. S., PRAHALAD M. S. 2010. "Fast and power efficient 16×16 array of array multiplier using Vedic multiplication". In 2010 5th International Microsystems Packaging Assembly and Circuits Technology Conference, IEEE. doi:10.1109/IMPACT.2010.5699463.

[13] PRADHAN M., PANDA R., SAHU S. K. 2011. MAC Implementation using Vedic Multiplication Algorithm. International Journal of Computer Applications, 21(7): 26-28.

[14] BANSAL Y., MADHU C. 2016. A novel high-speed approach for 16×16 Vedic multiplication with compressor adders. Computers & Electrical Engineering, 49:39-49.

[15] SAHU S. R., BHOI B. K., PRADHAN M. 2020. Fast signed multiplier using Vedic Nikhilam algorithm. IET Circuits Devices & Systems,14(8):1160-1166.

[16] RASHNO M., HAGHPARAST M., MOSLEH M. 2020. A new design of a low-power reversible Vedic multiplier. International Journal of Quantum Information, 18(5):2050002.

[17] GARG A., HOSHI G. 2018. Gate Diffusion Input based 4-bit Vedic Multiplier Design. IET Circuits Devices & Systems, 12(6):764-770. doi: 10.1049/iet-cds.2017.0454.

[18] SHI F. S. 1989. The rapid calculation method of Shi Fengshou. Beijing: Science Press (in Chinese)

[19] ZHU Y. W. 2020. The theory of scissor products and applications[EB/OL]. Beijing: Sciencepaper Online [2020-11-25]. http://www.paper.edu.cn/releasepaper/content/202011-59.