Journal of Theoretical and Applied Information Technology
15th March 2018. Vol.96. No 5
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MIRRORED VEDIC VERTICALLY AND CROSSWISE MULTIPLICATION TECHNIQUE (MVVCMT): LONG INTEGER MULTIPLICATION ALGORITHM

EVON M. O. ABU-TAIEH, PHD

Faculty of Information System, Jordan University-Aqaba-Jordan
E-mail: abutaieh@gmail.com

ABSTRACT

The paper presents Mirrored Vedic Vertically and Crosswise Multiplication Technique (MVVCMT) which is an algorithm based on Vedic Vertically and Crosswise Multiplication Technique. Vedic Vertically and Crosswise Multiplication Technique is an ancient Indian technique used to shorten the process of mental multiplication especially for big numbers. In India, the multiplication technique is still taught to kids to enhance their skills in mental multiplication. The proposed algorithm in this research was inspired by this ancient yet practical, easy to understand and apply multiplication technique. Vedic Vertically and Crosswise Multiplication Technique was rediscovered in 1965 by Swami Bharati Krishna Tirthaji in his book *Vedic Mathematics.* The proposed algorithm runs with only 100 multiplications rather than n^2 based number of multiplications. In this paper, the first section explains the Vedic Vertically and Crosswise Multiplication Technique with and example and algorithm. Then the paper presents a hands-on example to show the simplicity of the original technique. Next, the paper presents the proposed algorithm name "Mirrored Vedic Vertically and Crosswise Multiplication Technique" which is dubbed (MVVCMT).

Keywords: *Vedic mathematics; Multiplication Technique; complex numbers; Complex multiplication.*

1 INTRODUCTION

Multiplying two n-digits numbers is an open question in computer science. Long integer multiplication is an essential ingredient in public key cryptography methods RSA and Diffie & Helman, and ElGamel algorithms [1], [2], [3], and [4] respectively. Public key cryptosystems allows message encryption and appending unforgeable digital signature. "The RSA public-key cryptosystem is based on the dramatic difference between the ease of finding large prime numbers and the difficulty of factoring the product of two large prime numbers"[5]. Furthermore, "In order to achieve security with the RSA cryptosystem, however, it is advisable to work with integers that are several hundred bits long, to resist possible advances in the art of factoring" as stated by the same source. Long Numbers that have more than 15 digits are hard to multiply in fact computers round the numbers to the closest digit. For example, 999,999,999,999,999,999 is rounded to 999,999,999,999,999,000 hence, when multiplied the product of multiplication is not accurate. Accuracy is needed especially when dealing with numbers that pertains to ciphering applications. Another problem with multiplying long numbers is time; such multiplication takes a long time. A third problem is storage space, just imaging carrying out such a task manually. Furthermore, many used different algorithms to overcome these problems: Schoolbook long multiplication with complexity $O(n^2)$ [6], Karatsuba algorithm with complexity $O(n^{1.585})$ according to [7], 3-way Toom–Cook multiplication with complexity $O(n^{1.465})$, k-way Toom–Cook multiplication with complexity $O(n^{\log (2k-1)/\log k})$, Mixed-level Toom–Cook with complexity $O(n\, 2^{\sqrt{2\log n}} \log n)$ according to [8], Schönhage–Strassen algorithm with complexity O(n*log(n)*log(log(n))) according to [9] and [10], Fürer's algorithm with complexity $O(n\log n 2^{O(\log^* n)})$ according to [11]. In summary accuracy, speed and storage space are the three problems that present themselves when dealing with multiplying long numbers. Hence, reducing storage space, and increasing accuracy and speed when multiplying two large numbers is an open question in the algorithms arena.

In this paper section 2 will discussed the complexities of long integer multiplication algorithms: Karatsuba, 3-way Toom–Cook, k-way Toom–Cook, Mixed-level Toom–Cook, Schönhage–Strassen algorithm, and Fürer's algorithm. The paper then shows the Vedic Vertically and Crosswise Multiplication Technique, and will explain the technique using a hands-on example and suggested an algorithm, in section 3 and section 4 respectively. Then the paper will present the suggested and inspired algorithm named MVVCMT, with an algorithm and example in section 5 and section 6. The results of the research are discussed in section 7.

2 LONG INTEGERE MULTIPLICATION ALGORITHMS

Multiplication algorithms tried to solve the multiplication operation efficiency: Karatsuba, 3-way Toom–Cook, k-way Toom–Cook, Mixed-level Toom–Cook, Schönhage–Strassen algorithm, and Fürer's algorithm. In the next paragraphs these algorithms will be discussed.

The traditional schoolboy multiplication algorithm (classical multiplication algorithm), shown in figure 1, has the running time of O (n^2) and needs a storage (memory) of at least nXn matrix where n is the number of digits in the two integers. The algorithm is carried out by multiplying each digit of the multiplicand by each digit of the multiplier and then adding up all the properly shifted results.

Karatsuba algorithm with complexity $O(n^{1.585})$ was discovered in 1962 by Anatolii Alexeevitch Karatsuba [12] and [13] the idea is to reduce multiplication operation in a 2-digits numbers base-m from 4 operations to 3 operations, is basically the following:
Suppose the P & Q as follows: P_1*M+P_2 and Q_1*M+Q_2 to calculate their multiplication:

1. Compute $Result_1 = P_1*Q_1$
2. Compute $Results_2= P_1*Q_2$
3. Compute $Result_3 =(P_1+P_2) *(Q_1+Q_2)$
4. Compute $Result_4=Result_3-Result_1-Result_2$
5. Computer $Result_1 * m^2+Result_4 * m + Result_2$

The algorithm used 1 multiplication in the first step, and another multiplication in the second step. In the third step the algorithm consumed one multiplication and 2 additions operations. In the fourth step the algorithm consumed two subtraction operations. And in the final step two multiplications operations and two addition operations but they are just to place in the right index. Hence, the reduction of three multiplications rather than four multiplications, the improvement factor is 4:3. *Karatsuba Multiplication Algorithm* later named divide and conquer algorithm, shown in Figure (2). Another improvement is Toom-Cook algorithm which based on the work of Karatsuba algorithm claimed to a generalization of Karatsuba algorithm. Toom-Cook was developed by Andrei Toom and Stephen Cook. The improvement factor stated is 9:5. Toom-Cook is composed of five steps: Splitting, Evaluation, Pointwise multiplication, Interpolation and Recompositing as described by [14]. Bothe Karatsuba and Toom-Cook use Divide and conquer technique. Hence, preprocessing overheads, as explained in the following quote "Algorithms such as FFT and Toom-Cook have lower algorithm complexity. However, because of the preprocessing overheads such as the divide and conquer, evaluation, and interpolation, the operating cost of these algorithms is actually much higher, making them useful only when the integers are extremely large. Consequently, only classical and Karatsuba multiplication algorithms and their combination are being used in current cryptosystem. This is especially true after considering circumstances such as memory constraints and the practical finite field size. "[15]

```
multiply(a[1..p], b[1..q], base)                        // Operands containing rightmost digits at index 1
 product = [1..p+q]                                     //Allocate space for result
 for b_i = 1 to q                                      // for all digits in b
  carry = 0
  for a_i = 1 to p                      //for all digits in a
   product[a_i + b_i - 1] += carry + a[a_i] * b[b_i]
   carry = product[a_i + b_i - 1] / base
   product[a_i + b_i - 1] = product[a_i + b_i - 1] mod base
  product[b_i + p] += carry                            // last digit comes from final carry
 return product
```

Figure 1: Schoolboy multiplication [5].

```
procedure karatsuba(num1, num2)
  if (num1 < 10) or (num2 < 10)
    return num1*num2
  /* calculates the size of the numbers */
  m = max(size_base10(num1), size_base10(num2))
  m2 = m/2
  /* split the digit sequences about the middle */
  high1, low1 = split_at(num1, m2)
  high2, low2 = split_at(num2, m2)
  /* 3 calls made to numbers approximately half the size */
  z0 = karatsuba(low1,low2)
  z1 = karatsuba((low1+high1),(low2+high2))
  z2 = karatsuba(high1,high2)
  return (z2*10^(2*m2))+((z1-z2-z0)*10^(m2))+(z0)
```

Figure 2: Karatsuba Multiplication Algorithm [5].

"Cook showed how the actions of this machine could be simulated by an extremely complicated and long, but still polynomial, Boolean formula. This Boolean formula would be true if and only if the program which was being run by the Turing machine produced a "yes" answer for its input" [16] and [17].

As for the Schönhage–Strassen algorithm with complexity O(n*log(n)*log(log(n))) according to [9]. The two numbers multiplied are treated as two separate matrices of one column and 2*n rows. The two matrices are multiplied with FFT matrix of base 2*n using w_8 and the modulo integer I_{nteger}, the result is two matrices that are multiplied by each other: element by element. The result is one matrix that must go through Invers FFT. Again, the result is recombined using the carry operation.

Fürer's algorithm with complexity $O(n \log n 2^{O(\log^* n)})$ according to [11] is faster the Schönhage–Strassen algorithm. Furer (2007) states that his algorithm run very much like Schönhage–Strassen algorithm with two exceptions: the ring of integers modulo used, and FFT is divided "more evenly". Another fact stated by Furer [11] is "All known methods for integer multiplication (except the trivial school method) are based on some version of the Chinese Remainder Theorem". The Chinese Remainder Theorem states "There are certain things whose number is unknown. If we count them by threes, we have two left over; by fives, we have three left over; and by sevens, two are left over. How many things are there?" as [18] by a chines mathematician (Sun Zi) in the third century. The Chinese remainder theorem was finally stated and proved in its full generality by L. Euler in 1734[5]. The Chinese Remainder Theorem was stated formally as follows: let ***p***, ***q*** be co-prime then the system of equations has a unique solution for ***x*** modulo ***pq***. The same thing goes if we have more than two equations.

$$\boldsymbol{x} \equiv \boldsymbol{a} \pmod{\boldsymbol{p}}$$
$$\boldsymbol{x} \equiv \boldsymbol{b} \pmod{\boldsymbol{q}}$$

Looking back at Multiplication algorithms who tried to solve the long integer multiplication operation efficiency: Karatsuba algorithm, 3-way Toom–Cook, k-way Toom–Cook, Mixed-level Toom–Cook, Schönhage–Strassen algorithm, and Fürer's algorithm; and comparing their complexity with the traditional schoolboy, figure (1) multiplication algorithm $O(n^2)$ one can conclude the following: Fürer's algorithm is the least complex, followed by Schönhage–Strassen algorithm, then Toom-Cook versions, then Karatsuba algorithm. The constant quests in all the algorithms are: accuracy, speed and storage. All quested elements are reflected in the word algorithms complexity.

3 VERTICALLY AND CROSSWISE MULTIPLICATION TECHNIQUE (VCMT)

Vertically and Crosswise Multiplication Technique is an ancient Indian technique popularized by Swami Bharati Krishna Tirthaji's *Vedic Mathematics*, published posthumously in 1965. The book can be found on the website [19]. Multiplying two n-digit integer numbers using VCMT entails the work shown in (1)

$$A_{i+j} = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i y_{j+c_{i+j+1}} \ldots\ldots\ldots(1)$$

When multiplying two 4-digit numbers the multiplier named P and made up of digits $P_1P_2P_3P_4$ and the multiplicand Q is made up of four digits $Q_1Q_2Q_3Q_4$, where the first digit in each P and Q are the highest order digit.

Table (1): A trace table of the operations and expected results of 4-digits P and Q.

i+j	**A_{i+j}**	**Multiplication operations**	**Addition operations**
1	C_2		0
2	$P_1 Q_1+C_3$	1	1
3	$P_1Q_2+P_2Q_1+C_4$	2	2
4	$P_1Q_3+Q_1P_3+P_2Q_2+C_5$	3	3
5	$P_1Q_4+Q_1P_4+P_2Q_3+Q_2P_3+C_6$	4	4
6	$P_2Q_4+Q_2P_4+P_3Q_3+C_7$	3	3
7	$P_3Q_4+Q_3P_4+C_8$	2	2
8	P_4Q_4	1	0
Total Number of Operations		**16**	**15**

P_i is the ith digit in the multiplier, i=1...n. and Q_j is the jth digit in the multiplicand, j=1..n. The letter C is the carry from the carry operation. The 8th digit of the result is product of multiplying P_4Q_4 which produces 8th digit and the carry number C_8. The 7th digit is the result of $P_3Q_4+Q_3P_4+C_8$and produces the 7th carry. The 6th digit is the result of $P_2Q_4+Q_2P_4+P_3Q_3+C_7$ and produces the 6th carry. The 5th digit is the result of $P_1Q_4+Q_1P_4+P_2Q_3+Q_2P_3+C_6$ and produces the 5th carry. The 4th digit is the result of $P_1Q_3+Q_1P_3+P_2Q_2+C_5$ and produces the 4th carry. The 3ed digit is the result of $P_1Q_2+P_2Q_1+C_4$and produces the 3ed carry. The 2ed digit is the result of P_1 Q_1+C_3 and produces the 2ed carry. The first digit is the 2ed carry. The whole operation which is reflected in table 1 including 16 multiplication operations and 15 addition operations. The addition operations are: 10 addition operations and 5 addition operations for the carry. Therefore, when multiplying the two 4-digit numbers the following results table 1 is expected:

Same thing happens when multiply two 5-digits integer numbers P and Q as shown in digits' form P is $P_1P_2P_3P_4P_5$ and Q is $Q_1Q_2Q_3Q_4Q_5$. , where the first digit in each P and Q are the highest order digit. P_i is the ith digit in the multiplier, i=1...n. And Q_j is the jth digit in the multiplicand, j=1..n. The letter C is the carry from the carry operation. The 10th digit of the result is product of multiplying P_5Q_5 which produces 10th digit and the carry number C_{10}. The 9th digit of the result is product of multiplying $P_4Q_5+Q_4P_5+C_{10}$ which produces 9th digit and the carry number C_9. The 8th digit of the result is product of multiplying $P_3Q_5+Q_3P_{5+}P_4Q_4+C_9$ which produces 8th digit and the carry number C_8. The 7th digit of the result is product of multiplying $P_2Q_5+Q_2P_5+P_3Q_4+P_4Q_3+C_8$ which produces 7th digit and the carry number C_7. The 6th digit of the result is product of multiplying $P_1Q_5+Q_1P_5+P_2Q_4+Q_2P_4+P_3Q_3+C_7$which produces 6th digit and the carry number C_6. The 5th digit of the result is product of multiplying $P_1Q_4+Q_1P_4+P_2Q_3+Q_2P_3+C_6$ which produces 5th digit and the carry number C_5. The 4th digit of the result is product of multiplying $P_1Q_3+Q_1P_3+P_2Q_2+C_5$ which produces 4th digit and the carry number C_4. The 3th digit of the result is product of multiplying $P_1Q_2+P_2Q_1+C_4$ which produces 3ed digit and the carry number C_3. The 2ed digit of the result is product of multiplying P_1 Q_1+C_3which produces 2ed digit and the carry number C_2. The first digit is the 2ed carry. The results are summarized in the following tracing table 2:

Table (2): A trace table of the operations and expected results of 5-digits P and Q.

I+J	**A_{i+j}**	**Multiplication operations**	**Addition operations**
1	C_2		
2	$P_1 Q_1+C_3$	1	1
3	$P_1Q_2+P_2Q_1+C_4$	2	2
4	$P_1Q_3+Q_1P_3+P_2Q_2+C_5$	3	3
5	$P_1Q_4+Q_1P_4+P_2Q_3+Q_2P_3+C_6$	4	4
6	$P_1Q_5+Q_1P_5+P_2Q_4+Q_2P_4+P_3Q_3+C_7$	5	5
7	$P_2Q_5+Q_2P_5+P_3Q_4+P_4Q_3+C_8$	4	4
8	$P_3Q_5+Q_3P_5+P_4Q_4+C_9$	3	3
9	$P_4Q_5+Q_4P_5+C_{10}$	2	2
10	P_5Q_5	1	0
Total Number of operations		**25**	**24**

In the following algorithm, Figure 3, a proposed algorithm for the VCMT. Each step, in the algorithm, is counted to calculate the complexity of the algorithm which is based on the VCMT. The running time of the algorithm is *n*n+2n* hence the running time is almost n^2.

Algorithm	**Running time**
For i=n down to 1	n
For j=n down to 1	n^2
Product [j + i]=(P_i*Q_j)+ Product [j + i]	
Next j	
Next i	
For i=2n down to 1	2n
Result[i]=rightmost digit(product[i])	
Carry[i]=left (digits of product[i])	
Product[i-1] = product[i-1] + carry[i]	
Next i	

Figure 3: A suggested algorithm for VCMT with number of operations.

4 VEDIC MULTIPLICATION EXAMPLE

To visually enhance the understanding of Vertically and Crosswise Multiplication Technique (VCMT) the following example is given shown in the figures below. Multiplying a 4-digit number (5432) by 4-digit multiplicand (3124). The 4-digit multiplier named P is composed of 4 digits $P_1P_2P_3P_4$, the multiplicand Q is respectively made of 4-digits $Q_1Q_2Q_3Q_4$. P_1 is the highest digit in P

The first step is set the window to 1 which entails taking the highest first digit from multiplier and multiplicand in this case (5, 3) and multiply them to produce 15. The 15 is stored in the answer line as shown in the Figure 4 below.

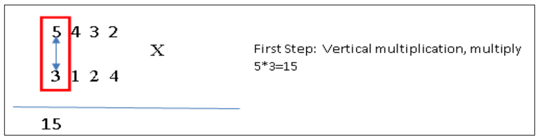

Figure 4: Step #1 of VCMT vertical multiplication.

Next, set the window to 2, choose the highest two digits and crisscross multiplication, see Figure 5. Multiply 4 by 3 and 5 by 1 and add the result: 5*1+4*3=17. The result is divided to two parts: a lowest digit is put in the answer line while the upper digit(s) are added to the answer line. In this example 7 is put on the answer line and the 1 is added to 15 to become 16.

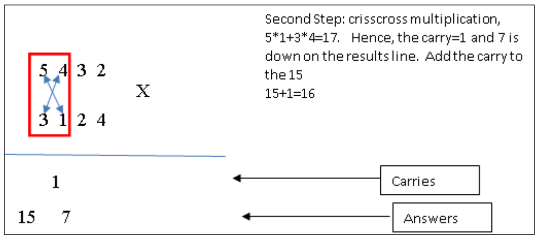
Second Step: crisscross multiplication, 5*1+3*4=17. Hence, the carry=1 and 7 is down on the results line. Add the carry to the 15
15+1=16
5 4 3 2
X
3 1 2 4
1
15 7
Carries
Answers

Figure 5: Step #2 of VCMT crisscross operation.

The third step: the window is increased by 1 to become 3. A crisscross operation is carried out by multiplying P_1*Q_3 (5*2) and multiplying P_3*Q_1 (3*3) and a vertical operation is carried out by multiplying P_2*Q_2 (4*1). And add all the results (10+9+4) which results 23. The 23 is again split where the lowest digit is put in the answer line and the carry=2 is added to previously mentioned (7) in the answer line to become 9, see Figure 6.

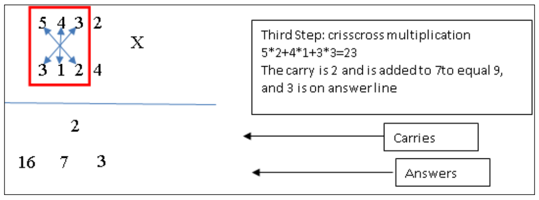
5 4 3 2
X
3 1 2 4
Third Step: crisscross multiplication
5*2+4*1+3*3=23
The carry is 2 and is added to 7to equal 9, and 3 is on answer line
2
16 7 3
Carries
Answers

Figure 6: Step #3 of VCMT crisscross operation.

The fourth step, the window is increased by 1 to become 4. A crisscross operation is carried out by: multiply P_1*Q_4 (5*4), P_2*Q_3 (4*4), P_3*Q_2(3*1), and P_4*Q_1 (2*3) which results to (20+8+3+6) which equals 37. The 37 is again split to answer and carry. The lowest digit (7) is put on the answer line and the 3 is added to previously produced 3 to become 6, Figure 7.

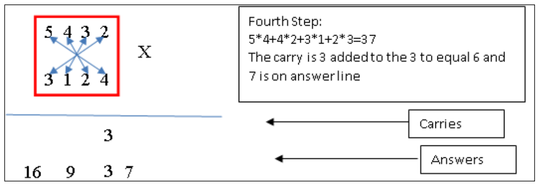

Figure 7: Step #4 of VCMT crisscross operation with window size 4.

In the fifth step and since the window reached the limit 4, the window is decreased by 1 to 3. A crisscross operation is carried out between P_2*Q_4 (4*4) and P_4*Q_2 (2*1). A vertical operation is carried out between P_3*Q_3 (3*2). The result (16+6+2) is 24. The lowest digit 4 is put on the answer line and the highest digit 2 is added to the previously produced 7 to become 9, Figure 8.

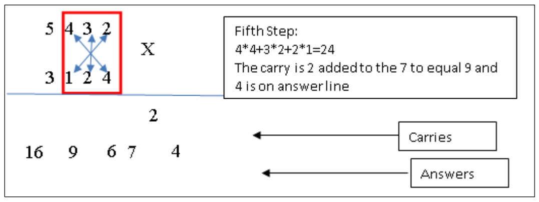

Figure 8: Step #5 of VCMT crisscross operation with window size 3.

The sixth step, the window is reduced by 1 to become two and a crisscross operation is carried out: P_3*Q_4 (3*4) and P_4*Q_3 (2*2). The rest is (12+4) which is 16, again the result is spilt to answer and carry. The lowest digit 6 is put on the answer line and the carry (1) is added to the previously produced 4 to become 5, see Figure 9.

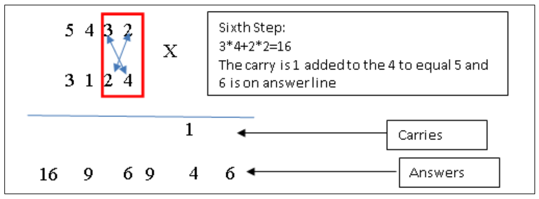
5 4 3 2
X
3 1 2 4
Sixth Step:
3*4+2*2=16
The carry is 1 added to the 4 to equal 5 and 6 is on answer line
1
Carries
16 9 6 9 4 6
Answers

Figure 9: Step #6 of VCMT crisscross operation with window size 2.

The seventh step, the window is reduced by 1 to become 1. And vertical operation is carried out P_4*Q_4 which is (2*4) the result is 8 and is put on the answer line, figure 10. By reading what is on the answer line 16969568 is the answer of the multiplication operation.

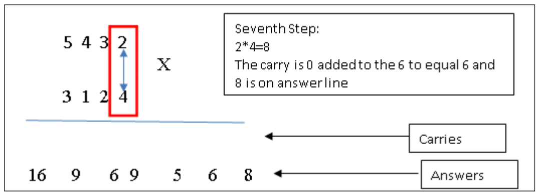
5 4 3 2
X
3 1 2 4
Seventh Step:
2*4=8
The carry is 0 added to the 6 to equal 6 and 8 is on answer line
Carries
16 9 6 9 5 6 8
Answers

Figure 10: Step #7 of VCMT crisscross operation with window size 1.

The example above intended to visually enhance the understanding of Vertically and Crosswise Multiplication Technique (VCMT). The example explained each step and the variation of the window size and the uses of the window. Figures 3 to figure 10 reflected visually the steps to explain the process of the Vertically and Crosswise Multiplication Technique (VCMT).

Looking back at the example above and counting the number of multiplications and additions we fine the following: In the first step there was one multiplication operation. In the second step there were two multiplication operations and one addition operation. In the third step there were three multiplication operations and two addition operations. In the fourth step, there were four multiplication operations and three addition operations. In the fifth step there were three multiplication operations and two addition operations. In the sixth step there were two multiplication operations and one addition operation. In the seventh step there was only one multiplication operation. One can conclude that the number of multiplication operations is 16 and the addition operations is 9 while there is an addition operation for the carry which is 7 addition operations which makes the total of the addition operation also 16. The summery of all operations in a 4-digit number multiplied by 4-digit number is in table1.

By using the lookup table, suggested for the algorithm MVVCMT explained in section 6 of this research paper, the number of multiplication operations will be reduced to 0. Since all multiplication operations are of simple multiplication operation. Hence, looking up each result from the multiplication table provided in figure 14 section 6.1 of this paper.

Journal of Theoretical and Applied Information Technology
15th March 2018. Vol.96. No 5
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5 SUGGESTED ALGORITHM: MIRRORED VEDIC VERTICALLY AND CROSSWISE MULTIPLICATION TECHNIQUE (MVVCMT)

Mirrored Vedic Vertically and Crosswise Multiplication Technique (MVVCMT) is an algorithm that reduces the number of multiplications operation in long integer multiplication to 100 simple multiplication operations. The algorithm works as follows:

- Scan both P & Q (multiplier and multiplicand) and produce the frequency of each digit and the index of each digit. The result is shown like in figure 12 and figure 13.
- Produce the result, carries and answer.

The following section will explain the algorithm with the two major steps in sections 6.1 and section 6.2.

6 MVVCMT

MVVCMT assumes that both multiplier and multiplicand are equal in number of digits and treats both multiplier and multiplicand as an array of digits rather than decimal number. The most significant digit is stored in an array with index 1 and the least significant digit is stored in the array with index *n*. Hence, the multiplier is a one-dimensional array of digits with size *n*. The multiplicand is one-dimensional array of digits with size *n*. The product of both multiplier and multiplicand are stored as individual digits in one-dimensional array with the size of *2n*.

MVVCMT has two parts: the first part named *Scan part*, the second part is the *multiplication lookup part*. The *Scan part* has an input of both multiplier (P) and multiplicand (Q). The *Scan part* will produce two matrices: a matrix for P and a Matrix for Q. The matrices include each digit in the P or Q with the frequency of each digit and indices of each digit. The second part of MVVCMT is the multiplication lookup part which will carry out the multiplication process. Both parts are explained in the next two sections.

```
Frequencey_index(P,Q)                                          Time
    Q_Frequencey=0                                              1
    P_Frequencey=0                                              1
    For i ← 1 to length[P]   ' (or Q)                           n
          P_Frequencey (P[i])= P_Frequencey (P[i])+1            n-1
          Q_Frequencey (P[i])= Q_Frequencey (Q[i])+1            n-1
          P_Index(P_Frequencey (P[i]))=i                        n-1
          Q_Index(Q_Frequencey (Q[i]))=i                        n-1
    Next i
```

Figure 11: The first part of MVVCMT is called Frequencey_index(P,Q).

6.1. Scan Part: Frequencey_index(*P,Q*)

The scan part of the algorithm, named **Frequencey_index(*P,Q*)**, produces two matrices: the first matrix is a one dimensional array that holds the frequency of each digit in P and Q. The suggested scan algorithm *Frequencey_index(P,Q)* is shown in Figure 11.

The *Frequencey_index(P,Q)* produces the matrix shown in Figure 12. The first part of the matrix reflects the frequency of each digit in P. The second part shows the index /location/order of the digit in P. To further explain assume that P is the following 70-digit number: 62642001874012850961516549482644422193020 37178623509019111660653946049. As can be seen in the matrix (Figure 12) there are 10 zeros and 10 ones and 8 twos in P. The second matrix is the one that keeps the location/position/index of the digit. The positions 38,42,49,64 in P are held by digit value "Three" as can be Figure 12. Hence, producing the Figure 10 and Figure 11:

Journal of Theoretical and Applied Information Technology
15th March 2018. Vol.96. No 5
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

P digits	P Frequency
0	10
1	10
2	8
3	4
4	9
5	5
6	10
7	3
8	4
9	7

P digits	Indices of each digit in P									
0	6	7	12	17	39	41	51	53	61	68
1	8	13	20	22	36	44	54	56	57	58
2	2	5	14	29	34	35	40	48		
3	38	42	49	64						
4	4	11	25	27	31	32	33	66	69	
5	16	21	24	50	63					
6	1	3	19	23	30	47	59	60	62	67
7	10	43	45							
8	9	15	28	46						
9	18	26	37	52	55	65	70			

Figure 12: A matrix that shows each digit in P with the frequency of each digit and the indices of digit.

Assume Q is the following 70 digits' number:3398717423028438554530123627613875835633986495969597423490929302771479. The second matrix is for Q, the digit, 0, has frequency of 4 and the digit 0 is located in index 11,22,58,63.

Again the digit 1 has frequency of 4 and is located in 6, 23, 30 and 67. The scan part of the suggested algorithm has running time of *n*, where n is the length of P or Q.

Q digits	Q Frequency
0	4
1	4
2	7
3	12
4	7
5	7
6	5
7	8
8	6
9	10

Q digits	Indices of each digit in Q											
0	11	22	58	63								
1	6	23	30	67								
2	9	12	24	27	54	60	64					
3	1	2	10	15	21	25	31	36	39	40	55	62
4	8	14	19	44	53	56	68					
5	17	18	20	34	37	46	50					
6	26	29	38	43	48							
7	5	7	28	33	52	65	66	69				
8	4	13	16	32	35	42						
9	3	41	45	47	49	51	57	59	61	70		

Figure 13: A matrix that shows each digit in P with the frequency of each digit and the indices of digit.

Create multiplication table like (figure 14), the purpose of the multiplication table is to be used as a lookup table. The use of the lookup table will reduce the multiplication operations from n^2 in schoolboy multiplication to a constant number 100 multiplication in the worst-case scenario. The use of such table is shown in the figure 15 of the suggested algorithm.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

Figure 14: A standard multiplication table.

6.2. The Multiplication Process Part

The second step is to go through the multiplication process. The algorithm uses the previously shown matrices (frequency and indices matrices). The algorithm has 4 nested loops, see Figure 15. The first loop goes through digits of Q (0-9). The second loop goes through the frequency of each digit. The third loop is like the first loop, but works on the digits of P. The fourth loop is like the second loop works on the frequency of each digit in P. Line 4 of the algorithm looks up the product of two digits in P and Q, thus saving the redundancy of multiplication. In the same token, reducing the number of multiplication to only **100** multiplications. Hence, reducing the look to *n* rather than n^2. The second part of the algorithm merely present the answer by calculating the carry and do the carry addition.

```
Algorithm steps                                                    Time
For Q_digits=0 to 9                                                10
    For JQ=1 to Q_frequencey(Q_digits)                             Digit Frequency
          For P_digits=0 to 9                                      10
                temp=multiplication_table(Q_digits,P_Digits)
                (hence reducing the number of multiplications to 100)
                  For JP=1 to P_frequencey(P_digits)               Digit Frequency
                            Result(JQ+JP+1) = temp+ Result(JQ+JP+1)
                    Next JP
          Next P_digits
    Next JQ
Next Q_digits
                                                                   2n+1
For w=2*length[P]+1 step -1 ' from lowest order digit              2n
          Answer(w)=RightMostDigit(Results(w))                     2n
          Carry=LeftDigits(Results(w))                             2n
          Results(w-1)=Results(w-1)+Carry
Next W
```

Figure 15: The second part of MVVCMT.

As can be seen in the previous the number of multiplication operations is reduced to 100 multiplications. The lookup operations of the multiplication operations are 100*P_ Digit Frequency* Q_Digit Frequency. The frequency of any particular digit is worth discussing here: P and Q are made of digits from 0 to 9. The frequency of any digit in P or Q must always be equal or less than *n*. If the frequency of certain digit is *n* this implies that the other 9 digits have frequency equal to 0. Hence, using the frequency & the index of each digit to look the multiplication result reduces the number of multiplication operation to only 100. The number of additions is reduced to *2n* where n is length of P or Q. The carry operation is also *2n* where n is number of digits in P or Q. Furthermore, one can state the following regarding the three basic operations:

Number of multiplication operation =100
Number of carry operations =2*n+1
Number of additions =2*n +10*digit frequency multiplier*10*digit frequency multiplicand.

Again, digit frequency is *n* in worst case which means the other 9 digits will have 0 frequencies. Regarding the storage problem (memory) the suggested algorithm needs for storage is matrix 10X10 to hold the multiplication table. Frequency matrix with 10 rows and *n* columns (worst case). Two arrays with each one row and length *2n*+1 to store the answer and carry results. Hence, the storage needs are minimal, and any basic computer handles such requirements.

Regarding accuracy, the algorithm never round results as was shown previously in the introduction section. Where computers round numbers more than 15 digits long, this algorithm can handle numbers up to 200 digits using Microsoft Excel. Furthermore, numbers being added or multiplied are kept to basic digits 0-9 and additions with carry are used on small numbers.

7 RESULTS AND DISCUSSION

The suggested algorithm *MVVCMT* is made of two parts. The two parts explained in section 6.1 and section 6.2 respectively. The algorithm succeeded in reducing the number of the simple multiplication operations to a constant which is 100 simple multiplication operations in the multiplication table. The algorithm also succeeded in reducing the need of storage to 2n+1 for the answer storage and to 2n+1 for the carry operations. Although the time complexity of the algorithm is n^2 still, the operation entailed are all simple lookup operations from the multiplication table and addition operations for the carry operations. The suggested algorithm MVVCMT, unlike *Karatsuba*'s algorithm, Schönhage–Strassen algorithm, Toom-Cook, and schoolboy algorithm.

The MVVCMT algorithm needs NOT to preprocess the numbers like *Karatsuba*'s algorithm there are no: Splitting, Evaluation, Pointwise multiplication, Interpolation and Recompositing as in Toom-Cook. Further, MVVCMT needs no two separate matrices like in Schönhage–Strassen algorithm. Nor does MVCCMT needs a nXn Matrix like the traditional schoolboy multiplication algorithm.

8 CONCLUSION

Long integer multiplication has many applications and uses. Long integer multiplication is an essential ingredient in public key cryptosystems. Many public key crypto systems require such long integer multiplication algorithms that are fast, precise and require less memory. "Multiplying polynomials with real or complex coefficients is a major area where long integer multiplication is very useful. Long integer multiplication is used extensively for finding large prime numbers. Another application is the computation of billions of digits of π to study patterns. A very practical application is the testing computational hardware" stated by[11].

This paper suggested an algorithm Mirrored Vedic Vertically and Crosswise Multiplication Technique (MVVCMT) which inspired by Vedic Vertically and Crosswise Multiplication Technique. The algorithm is to deal with multiplying long integers with digits count 2 to 200 digits. The 200 digit integer multiplication is recommended by [1] as they stated in their paper " We recommend that n be about 200 digits long. Longer or shorter lengths can be used depending on the relative importance of encryption speed and security in the application at hand. An 80-digit n provides moderate security against an attack using current technology; using 200 digits provides a margin of safety against future developments" which was reflected in in the work [20]. The basic requirements for such algorithm are speed, accuracy and storage space (memory). The paper discussed the complexities of Karatsuba, 3-way Toom–Cook, k-way Toom–Cook, Mixed-level Toom–Cook, Schönhage–Strassen algorithm, and Fürer's algorithm. The paper showed the Vedic Vertically and Crosswise Multiplication Technique, and explained the technique using a hands-on example and suggested an algorithm. Then the paper presented (MVVCMT), with an algorithm and example. MVVCMT has advantage over other algorithms since it reduces the number of digit multiplication to only 100 operations regardless of the number of the digits of P & Q. MVVCMT also has the running time of n^2 yet the number of multiplications and additions are reduced: the number of carry operations is reduced to 2n and the number of additions is 2*n +10*digit frequency multiplier*10*digit frequency multiplicand. Furthermore, the storage needed (memory) is 10X10 matrix, 2*n to store the multiplier and multiplicand and a 2*n array to hold the result. Each element in the previously mentioned matrices is of size one digit (0-9) hence storage space is minimal. MVVCMT is inspired by a Technique taught to children to improve their mental multiplication, where children can enjoy multiplying long integer. Hence, the algorithm is simple, easy to use, needs minimum storage and utilizes only 100 multiplication operations. MVVCMT algorithm is unlike: *Karatsuba*'s algorithm, Schönhage–Strassen algorithm, Toom-Cook algorithm and schoolboy algorithm. While such algorithms needed: preprocessing, Splitting, evaluation, pointwise multiplication, Interpolation, recompositing, nXn matrices. The MVVCMT algorithm reduced the number of multiplication operations and reduced the need for memory.

REFERENCES

[1] Rivest, Ronald L., Shami,r Adi, and Adleman, Leonard M. 1978. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978. See also U.S. Patent 4,405,829. Retrieved 20 Jun 2017 from: https://people.csail.mit.edu/rivest/Rsapaper.pdf

[2] Diffie, W. and Hellman, M. E., 1976. New directions in cryptography, Institute of Electrical and Electronics Engineers, vol. 22, no. 6, pp. 644–654, 1976. Retrieved 20 Jun 2017 From: https://www-ee.stanford.edu/~hellman/publications/24.pdf

[3] ElGamal, T. 1985. A public key cryptosystem and a signature scheme based on discrete logarithm, in Advances in Cryptology, vol. 196 of Lecture Notes in Computer Science, pp. 10–18, 1985. Retrieved 20 Jun 2017 From: http://people.csail.mit.edu/alinush/6.857-spring-2015/papers/elgamal.pdf

[4] Pollard, J.M. Theorems on factorization and primality testing. Proc. Camb. Phil. Soc. 76 (1974), 521-528. Retrieved 20 Jun 2017 from: http://maths-people.anu.edu.au/~brent/pd/rpb120.pdf

[5] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. 2001. Introduction to Algorithms (2nd ed.). MIT

Press and McGraw-Hill. pp. 55–56. ISBN 0-262-03293-7. Retrieved 20 Jun 2017 From: http://is.ptithcm.edu.vn/~tdhuy/Programming/Introduction.to.Algorithms.pdf

[6] Aboud S. J. ,Abu-Taieh E. M, A. 2006. New Deterministic RSA-Factoring Algorithm, Jordan Journal of Applied Science, Volume 8, No. 1, pp. 54-66, Amman-Jordan, 2006.

[7] Karatsuba A. 1995. The Complexity of Computations. Proceedings of the Steklov Institute of Mathematics. 211: 169–183. Translation from Trudy Mat. Inst. Steklova, 211, 186–202. Accessed 21/1/2017 from http://www.ccas.ru/personal/karatsuba/divcen.pdf

[8] Knuth, D. 1997. The Art of Computer Programming, Volume 2. Third Edition, Addison-Wesley. Section 4.3.3.A: Digital methods, pg.294. Retrieved 20 Jun 2017 from: http://broiler.astrometry.net/~kilian/The_Art_of_Computer_Programming%20-%20Vol%201.pdf

[9] Schönhage A., Strassen V., 1971. Schnelle Multiplikation großer Zahlen", Computing 7 (1971), pp. 281–292.

[10] Aho, Alfred V., Hopcroft, John E., and Ullman, Jeffrey D. 1974. The design and analysis of computer algorithms, Addison-Wesley, Reading, Massachusetts, 1974. Retrieved 20 Jun 2017, from: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/The%20Design%20and%20Analysis%20of%20Computer%20Algorithms%20%5BAho%2C%20Hopcroft%20%26%20Ullman%201974-01-11%5D.pdf

[11] Fürer, M. 2007. Faster Integer Multiplication. Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11–13, 2007, pp. 55–67. Retrieved 20 Jun 2017 from: https://web.archive.org/web/20130425232048/http://www.cse.psu.edu/~furer/Papers/mult.pdf

[12] Karatsuba A. and Ofman Y. 1962. Multiplication of Many-Digital Numbers by Automatic Computers. Proceedings of the USSR Academy of Sciences. 145: 293–294. Translation in the academic journal Physics-Doklady, 7 (1963), pp. 595–596

[13] Warren Jr., Henry S. (2013). Hacker's Delight (2 ed.). Addison Wesley - Pearson Education, Inc. ISBN 978-0-321-84268-8. Retrieved 20 Jun 2017 From: https://doc.lagout.org/security/Hackers%20Delight.pdf

[14] Bodrato M. 2007. Towards Optimal Toom–Cook Multiplication for Univariate and Multivariate Polynomials in Characteristic 2 and 0. In WAIFI'07 proceedings, volume 4547 of LNCS, pages 116–133. June 21–22, 2007. Retrieved 20 Jun 2017 From: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.790&rep=rep1&type=pdf

[15] Jahani, S., Samsudin, A., Subramanian,K.G. 2014. Efficient Big Integer Multiplication and Squaring Algorithms for Cryptographic Applications, Journal of Applied Mathematics, vol. 2014, Article ID 107109, 9 pages, 2014. doi:10.1155/2014/107109

[16] Weiss, M. 2014. Data Structures and algorithms analysis in C++, fourth edition, Pearson, USA. Retrieved 20 Jun 2017 from: http://iips.icci.edu.iq/images/exam/DataStructuresAndAlgorithmAnalysisInCpp_2014.pdf

[17] Cook, S. 1971. The Complexity of Theorem Proving Procedures, Proceedings of the Third Annual ACM Symposium on Theory of Computing (1971), 151–158. Retrieved 20 Jun 2017 From: https://www.cs.toronto.edu/~sacook/homepage/1971.pdf

[18] Dence, Joseph B., Dence ,Thomas P. 1999. Elements of the Theory of Numbers . Retrieved 20 Jun 2017 From: https://books.google.jo/books?id=YiYHw7evhjkC&pg=PA156&redir_esc=y#v=onepage&q&f=false

[19] Swami Bharati Krishna Tirthaji. 1965. Vedic Mathematics. Retrieved 20 Jun 2017 from: http://www.indiadivine.org/content/files/file/40-vedic-mathematics-by-shankaracharya-bharati-krishna-tirtha-pdf/

[20] Abu-Taieh, E. M. 1994. A genetic factoring algorithm for RSA's N= P* Q and performance comparison to factorization algorithm of Fermat, Pollard's rho, and Eratosthenes' Sieve. Master thesis, Pacific Lutheran University, Tacoma, WA, USA.