International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:7, 2008

A Reduced-Bit Multiplication Algorithm for Digital Arithmetic

Harpreet Singh Dhillon and Abhijit Mitra

***Abstract*—**A reduced-bit multiplication algorithm based on the ancient Vedic multiplication formulae is proposed in this paper. Both the Vedic multiplication formulae, Urdhva tiryakbhyam and Nikhilam, are first discussed in detail. Urdhva tiryakbhyam, being a general multiplication formula, is equally applicable to all cases of multiplication. It is applied to the digital arithmetic and is shown to yield a multiplier architecture which is very similar to the popular array multiplier. Due to its structure, it leads to a high carry propagation delay in case of multiplication of large numbers. Nikhilam Sutra, on the other hand, is more efficient in the multiplication of large numbers as it reduces the multiplication of two large numbers to that of two smaller numbers. The framework of the proposed algorithm is taken from this Sutra and is further optimized by use of some general arithmetic operations such as expansion and bit-shifting to take advantage of bit-reduction in multiplication. We illustrate the proposed algorithm by reducing a general 4×4-bit multiplication to a single 2×2-bit multiplication operation.

***Keywords*—**Multiplication, algorithm, Vedic mathematics, digital arithmetic, reduced-bit.

I. Introduction

DIGITAL multipliers [1], [2] are the core components of all the digital signal processors (DSPs) and the speed of the DSP is largely determined by the speed of its multipliers [3]. They are indispensable in the implementation of computation systems realizing many important functions such as fast Fourier transforms (FFTs) and multiply accumulate (MAC). Two most common multiplication algorithms followed in the digital hardware are array multiplication algorithm and Booth multiplication algorithm [4]. The computation time taken by the array multiplier is comparatively less because the partial products are calculated independently in parallel. The delay associated with the array multiplier is the time taken by the signals to propagate through the gates that form the multiplication array. Booth multiplication is another important multiplication algorithm [5]. Large booth arrays are required for high speed multiplication and exponential operations which in turn require large partial sum and partial carry registers. Multiplication of two n-bit operands using a radix-4 booth recording multiplier requires approximately $n/(2m)$ clock cycles to generate the least significant half of the final product, where m is the number of Booth recoder adder stages. Thus, a large propagation delay is associated with this case. Due to the importance of digital multipliers in DSP, it has always been an active area of research and a number of interesting multiplication algorithms have been reported in the literature [6]–[9].

This paper presents one such new multiplication algorithm which circumvents the need of large multipliers by reducing the multiplication of large numbers to that of smaller numbers. This reduces the propagation delay associated with the conventional large multipliers considerably. The framework of the proposed algorithm is primarily based on the Nikhilam Sutra (formula) of Vedic mathematics [10] and is further optimized to take full advantage of reduction in the number of bits in multiplication.

Although Nikhilam Sutra is applicable to all cases of multiplication, it is more efficient when the numbers involved are large. In addition to this Sutra, Vedic mathematics deals with another multiplication formula, Urdhva tiryakbhyam, which is equally applicable to all cases of multiplication. Attempts have been made in the literature to apply this general multiplication formula to binary arithmetic. In [11], this Sutra is shown to be more efficient multiplication algorithm as compared to the conventional counterparts. Another paper [12] has also shown the effectiveness of this Sutra to reduce the $N \times N$ multiplier structure into an efficient 4×4 multiplier structures.

In this paper, Urdhva tiryakbhyam Sutra is first applied to the binary number system and is used to develop a digital multiplier architecture. This is shown to be very similar to the popular array multiplier architecture. Nikhilam Sutra is then discussed and is shown to be much more efficient in the multiplication of large numbers as it reduces the multiplication of two large numbers to that of two smaller ones. The proposed multiplication algorithm is then illustrated to show its computational efficiency by taking an example of reducing a 4×4-bit multiplication to a single 2×2-bit multiplication operation. The basic framework of the proposed algorithm is taken from the Nikhilam Sutra of Vedic mathematics and is further optimized to take full advantage of the bit-reduction in multiplication.

This paper is organized as follows. In Section 2, a brief overview of Vedic mathematics is provided. Section 3 deals with the Vedic multiplication Sutras followed by the proposed algorithm. Concluding remarks are presented in Section 4.

II. Vedic Mathematics

Vedic mathematics is the name given to the ancient Indian system of mathematics that was rediscovered in the early twentieth century from ancient Indian sculptures (Vedas) by Sri B. K. Tirtha (1884-1960) [10]. It mainly deals with Vedic mathematical formulae and their application to various

Manuscript received Mar 01, 2008.

The authors are with the Department of Electronics and Communication Engineering, Indian Institute of Technology (IIT) Guwahati, India. E-Mail: {harpreet; a.mitra}@iitg.ernet.in.

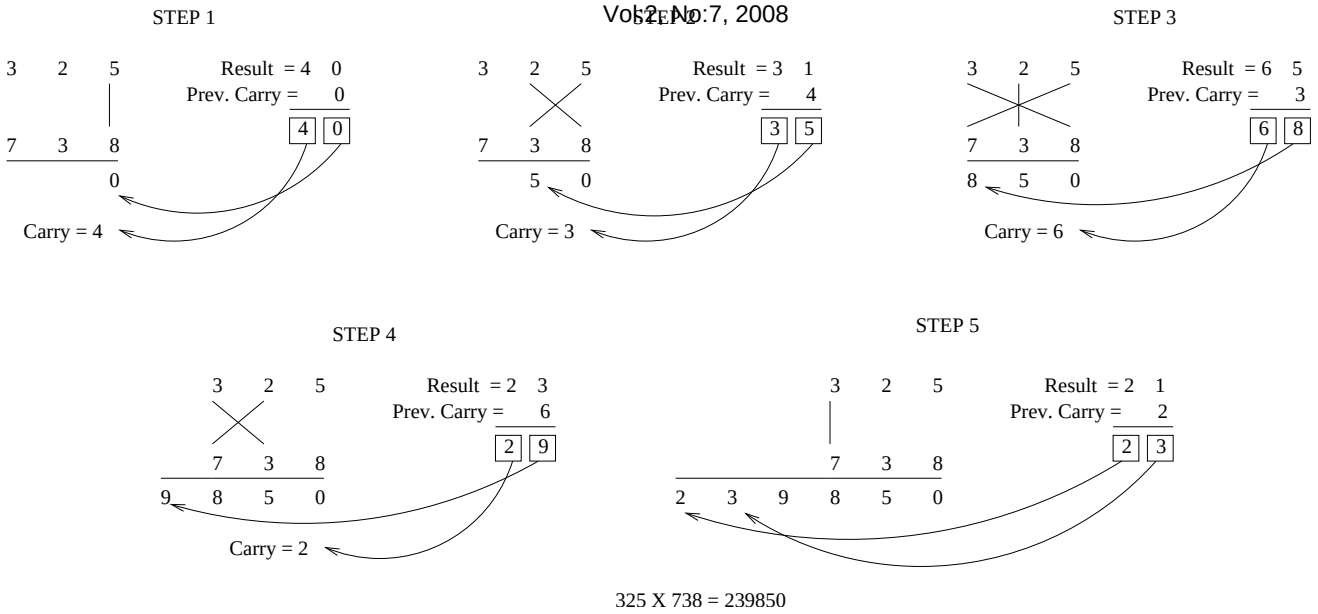

Fig. 1. Multiplication of two decimal numbers by Urdhva tiryakbhyam Sutra.

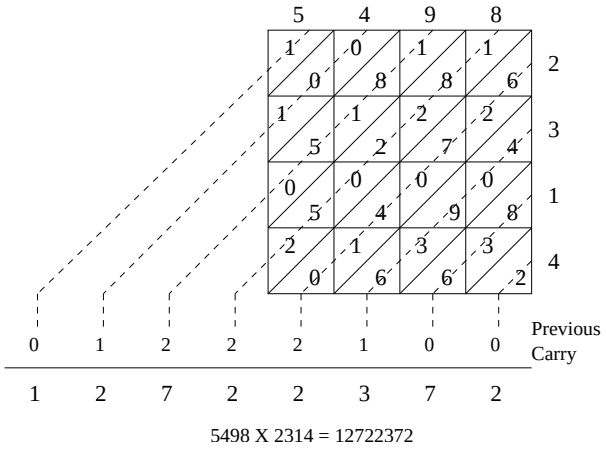

Fig. 2. Alternative way of multiplication by Urdhva tiryakbhyam Sutra.

branches of mathematics. The algorithms based on conventional mathematics can be simplified and even optimized by the use of Vedic Sutras. The word 'Vedic' is derived from the word 'veda' which means the store-house of all knowledge. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. [10]. These Sutras along with their brief meanings are enlisted below alphabetically.

1) *(Anurupye) Shunyamanyat* – If one is in ratio, the other is zero.
2) *Chalana-Kalanabyham* – Differences and Similarities.
3) *Ekadhikina Purvena* – By one more than the previous one.
4) *Ekanyunena Purvena* – By one less than the previous one.
5) *Gunakasamuchyah* – The factors of the sum is equal to the sum of the factors.
6) *Gunitasamuchyah* – The product of the sum is equal to the sum of the product.
7) *Nikhilam Navatashcaramam Dashatah* – All from 9 and the last from 10.
8) *Paraavartya Yojayet* – Transpose and adjust.
9) *Puranapuranabyham* – By the completion or non-completion.
10) *Sankalana-vyavakalanabhyam* – By addition and by subtraction.
11) *Shesanyankena Charamena* – The remainders by the last digit.
12) *Shunyam Saamyasamuccaye* – When the sum is the same that sum is zero.
13) *Sopaantyadvayamantyam* – The ultimate and twice the penultimate.
14) *Urdhva-tiryakbyham* – Vertically and crosswise.
15) *Vyashtisamanstih* – Part and Whole.
16) *Yaavadunam* – Whatever the extent of its deficiency.

These methods and ideas can be directly applied to trigonometry, plain and spherical geometry, conics, calculus (both differential and integral), and applied mathematics of various kinds. As mentioned earlier, all these Sutras were reconstructed from ancient Vedic texts early in the last century [10]. Many Sub-sutras were also discovered at the same time which are not discussed here.

The beauty of Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a very simple ones. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing [13]–[15].

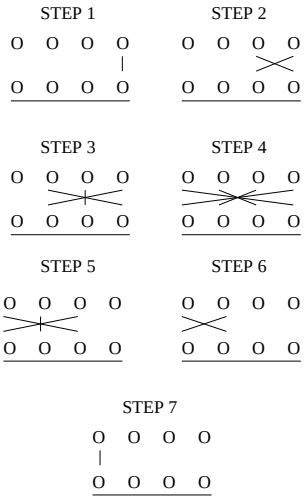

Fig. 3. Line diagram for multiplication of two 4-bit numbers.

III. The Proposed Vedic Multiplier

The proposed Vedic multiplier is based on the Vedic multiplication formulae (Sutra). These Sutra have been traditionally used for the multiplication of two numbers in the decimal number system. In this paper, we apply the same ideas to the binary number system to make the proposed algorithm compatible with the digital hardware. Let us first discuss both these Sutras in detail.

A. Urdhva Tiryakbhyam Sutra

Urdhva tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and Crosswise". To illustrate this multiplication scheme, let us consider the multiplication of two decimal numbers (325×728). Line diagram for the multiplication is shown in Fig. 1. The digits on the two ends of the line are multiplied and the result is added with the previous carry. When there are more lines in one step, all the results are added to the previous carry. The least significant digit of the number thus obtained acts as one of the result digits and the rest act as the carry for the next step. Initially the carry is taken to be zero.

An alternative method of multiplication using Urdhva tiryakbhyam Sutra is shown in Fig. 2. The numbers to be multiplied are written on two consecutive sides of the square as shown in the figure. The square is divided into rows and columns where each row/column corresponds to one of the digit of either a multiplier or a multiplicand. Thus, each digit of the multiplier has a small box common to a digit of the multiplicand. These small boxes are partitioned into two halves by the crosswise lines. Each digit of the multiplier is then independently multiplied with every digit of the multiplicand and the two-digit product is written in the common box. All the digits lying on a crosswise dotted line are added to the previous carry. The least significant digit of the obtained number acts as the result digit and the rest as the carry for the next step. Carry for the first step (i.e., the dotted line on the extreme right side) is taken to be zero.

Now we extend this Sutra to binary number system. To illustrate the multiplication algorithm, let us consider the multiplication of two binary numbers $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$. As the result of this multiplication would be more than 4 bits, we express it as $...r_3r_2r_1r_0$. Line diagram for multiplication of two 4-bit numbers is shown in Fig. 3 which is nothing but the mapping of the Fig. 1 in binary system. For the sake of simplicity, each bit is represented by a circle. Least significant bit r_0 is obtained by multiplying the least significant bits of the multiplicand and the multiplier. The process is followed according to the steps shown in Fig. 3. As in the last case, the digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result (r_n) and a carry (say c_n). This carry is added in the next step and hence the process goes on. If more than one lines are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and all the other bits act as carry. For example, if in some intermediate step, we get 110, then 0 will act as result bit and 11 as the carry (referred to as c_n in this text). It should be clearly noted that c_n may be a multi-bit number. Thus we get the following expressions:

$$
\begin{aligned}
r_0 &= a_0b_0, && (1)\\
c_1r_1 &= a_1b_0 + a_0b_1, && (2)\\
c_2r_2 &= c_1 + a_2b_0 + a_1b_1 + a_0b_2, && (3)\\
c_3r_3 &= c_2 + a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3, && (4)\\
c_4r_4 &= c_3 + a_3b_1 + a_2b_2 + a_1b_3, && (5)\\
c_5r_5 &= c_4 + a_3b_2 + a_2b_3, && (6)\\
c_6r_6 &= c_5 + a_3b_3 && (7)
\end{aligned}
$$

with $c_6r_6r_5r_4r_3r_2r_1r_0$ being the final product. Hence this is the general mathematical formula applicable to all cases of multiplication. The hardware realization of a 4-bit multiplier using this Sutra is shown in Fig. 4. This hardware design is very similar to that of the famous array multiplier where an array of adders is requited to arrive at the final product. All the partial products are calculated in parallel and the delay associated is mainly the time taken by the carry to propagate through the adders which form the multiplication array. Clearly, this is not an efficient algorithm for the multiplication of large numbers as a lot of propagation delay is involved in such cases. To deal with this problem, we now discuss Nikhilam Sutra which presents an efficient method of multiplying two large numbers.

B. Nikhilam Sutra

Nikhilam Sutra literally means "all from 9 and last from 10". Although it is applicable to all cases of multiplication, it is more efficient when the numbers involved are large. Since it finds out the compliment of the large number from its nearest base to perform the multiplication operation on it, larger the original number, lesser the complexity of the multiplication. We first illustrate this Sutra by considering the multiplication of two decimal numbers (96×93) where the chosen base

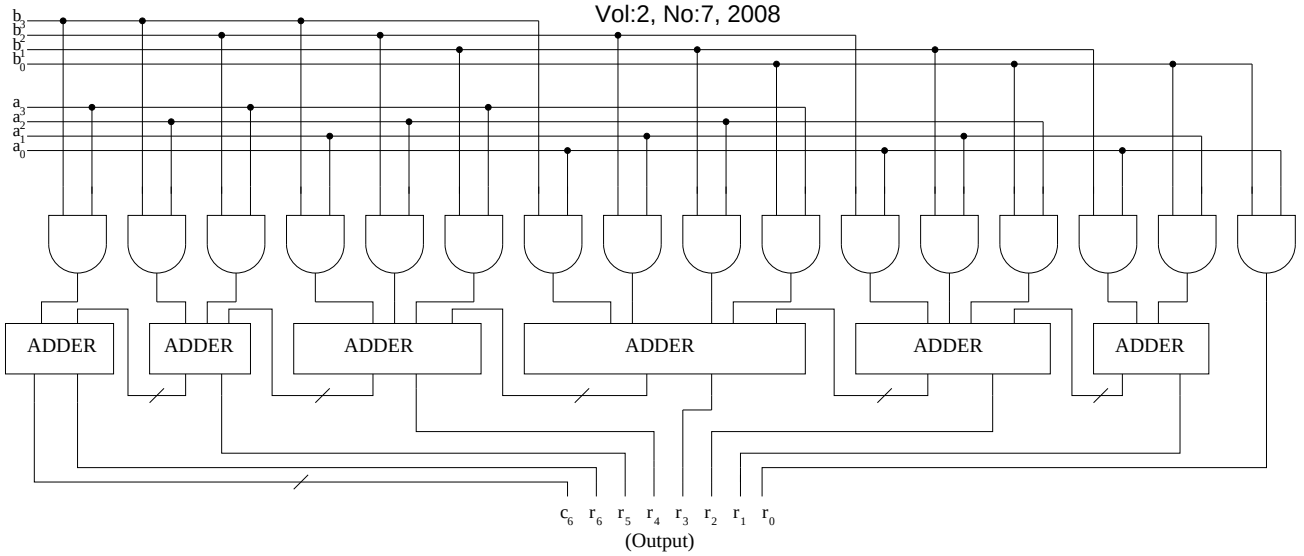

Fig. 4. Hardware architecture of the Urdhva tiryakbhyam multiplier.

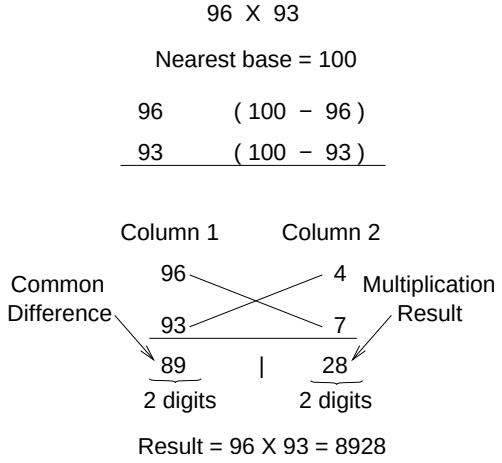

Fig. 5. Multiplication using Nikhilam Sutra.

is 100 which is nearest to and greater than both these two numbers.

As shown in Fig. 5, we write the multiplier and the multiplicand in two rows followed by the differences of each of them from the chosen base, i.e., their compliments. We can now write two columns of numbers, one consisting of the numbers to be multiplied (Column 1) and the other consisting of their compliments (Column 2). The product also consists of two parts which are demarcated by a vertical line for the purpose of illustration. The right hand side (RHS) of the product can be obtained by simply multiplying the numbers of the Column 2 ($7 \times 4 = 28$). The left hand side (LHS) of the product can be found by cross subtracting the second number of Column 2 from the first number of Column 1 or vice versa, i.e., $96 - 7 = 89$ or $93 - 4 = 89$. The final result is obtained by concatenating RHS and LHS (Answer = 8928).

After this illustration, we now discuss the operational principle of Nikhilam Sutra by taking the case of multiplication of two n–bit numbers x and y having compliments $\bar{x} = 10^n - x$ and $\bar{y} = 10^n - y$ respectively. The required product 'p' is defined as:

$$p = xy, \tag{8}$$

which can be reframed by adding and subtracting $10^{2n} + 10^n(x + y)$ to the right hand side as:

$$p = xy + 10^{2n} - 10^{2n} + 10^n(x + y) - 10^n(x + y). \tag{9}$$

The above terms can be clubbed as follows:

$$\begin{aligned} p &= \{10^n(x + y) - 10^{2n}\} + \{10^{2n} - 10^n(x + y) + xy\} \\ &= 10^n\{(x + y) - 10^n\} + \{(10^n - x)(10^n - y)\} \\ &= 10^n\{x - \bar{y}\} + \{\bar{x}\bar{y}\} = 10^n\{y - \bar{x}\} + \{\bar{x}\bar{y}\}. \end{aligned} \tag{10}$$

From (10), the expressions of LHS and RHS can be deduced, which come out to be:

$$LHS = \{x - \bar{y}\} = \{y - \bar{x}\}, \tag{11}$$

$$RHS = \{\bar{x}\bar{y}\}. \tag{12}$$

Hence the multiplication of two n- bit numbers is reduced to the multiplication of their compliments. To take full advantage of this reduction, it should be ensured that the numbers obtained after taking the compliments are lesser than the original numbers. This condition is satisfied if both the original numbers are greater than $10^n/2$, i.e., $x > 10^n/2$ and $y > 10^n/2$. This is the reason why it is said that the Nikhilam Sutra is more efficient in the multiplication of large numbers than the smaller ones.

An important point to note here is the number of digits required in the RHS of the product. From (10), it is clear that RHS should have n digits irrespective of number of digits in the product $\bar{x}\bar{y}$. We illustrate this point by considering a special case of the multiplication of two 2- digit numbers in which RHS comes out to be a single digit (99×97). As shown in Fig. 6, the LHS of the product comes out to be $99 - 3 =$

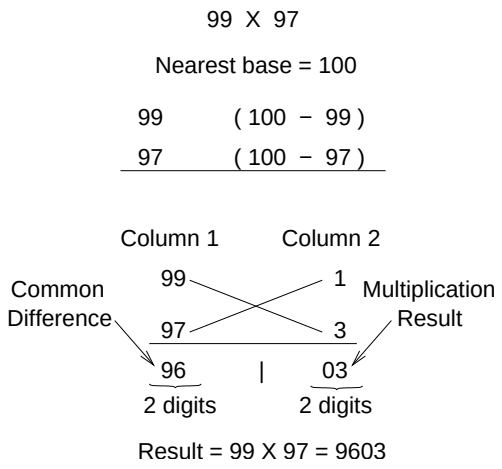

Fig. 6. Multiplication using Nikhilam Sutra.

$97 - 1 = 96$ and the RHS comes out to be $3 \times 1 = 3$. As $n = 2$ in this case, we need to append a leading zero to the RHS making it to be 03. The final result thus comes out to be 9603. On the other hand, if the number of digits in RHS would have been three, then the most significant digit would be the carry digit to LHS. With this knowledge of the Vedic formulae, we now describe the developed multiplication algorithm.

C. The Proposed Algorithm

We recast the Nikhilam Sutra in binary arithmetic and further exploit certain basic properties of multiplication like shifting to propose a new reduced-bit multiplication algorithm. The proposed algorithm is summarized in Table 1 for the case of multiplying two 4-bit numbers. It is shown that the proposed algorithm reduces 4×4-bit multiplication to a single 2×2-bit multiplication operation. As a result, it reduces the delay for carry propagation than any standard 4×4-bit multiplier.

In the preprocessing stage, the input binary numbers are right shifted to remove the least significant consecutive zero bits. This decreases the computational time by reducing the number of bits in the multiplier and the multiplicand. The effect of the removed zero bits is efficiently incorporated by shifting the final product to the left by equal number of bits. As explained in Table 1, if both the multiplier and the multiplicand obtained after the preprocessing stage are 4-bit numbers, Nikhilam Sutra is directly applied to reduce the numbers to atmost 3-bits. If the numbers thus obtained are exactly 3-bit numbers, the sutra is again applied to reduce the multiplication to 2×2-bit which can be done with any standard multiplier. In another case, if the numbers obtained after preprocessing stage are 4-bit and 3-bit numbers, the larger number is expanded as a sum of binary 1000 and a 3-bit number. This reduces the entire multiplication to a 3-bit multiplication operation, followed by a shift and an addition operation. A 3-bit multiplication is further reduced to a 2-bit multiplication operation as explained in the previous case. On the other hand, if one of the numbers is 3-bit long and the other one 2-bit, the larger number is expanded as a sum of binary 100 and a 2-bit number. This reduces the entire multiplication to a 2-bit multiplication, shift and an addition operation. If the preprocessing stage outputs 3-bit or 2-bit numbers, they are processed as explained earlier. Finally, another case might arise in our processing stage where either of the numbers is 1. In that case, the output always equals to the other number irrespective of the value of the number obtained after processing. The entire algorithm to be followed in the 4×4 multiplication operation is presented in Table 1. Here, we have exploited the basic operational principle of Nikhilam Sutra in conjunction with certain other basic arithmetic operations like decomposition and bit shifting as explained above.

The proposed multiplier algorithm can further be extended for larger numbers with some modifications in the algorithm condition checking steps accordingly.

TABLE I
THE PROPOSED REDUCED-BIT MULTIPLICATION ALGORITHM

(a) **Initialization**
Predefine: $flag_1 = flag_2 = flag_3 = flag_4 = 0$
(b) **Preprocessing**
Input 4-bit binary numbers a and b
n_1 = Number of least significant consecutive zeros in a
n_2 = Number of least significant consecutive zeros in b
$n = n_1 + n_2$
$\bar{a}$ = Right shift a by n_1
$\bar{b}$ = Right shift b by n_2
(c) **Processing**
1. IF ($\bar{a} > 1000$ & $\bar{b} > 1000$) THEN
$\bar{a} = 10000 - \bar{a}$; $\bar{b} = 10000 - \bar{b}$;
$flag_1 = 1$
2. IF ($\bar{a} > 100$ & $\bar{b} > 1000$) THEN $\bar{b} = \bar{b} - 1000$;
[*Solution* $= \bar{a} \times 1000 + \bar{b} \times \bar{a}$]
$flag_2 = 1$
[*If* $\bar{b} > 100$ & $\bar{a} > 1000$, *THEN* $\bar{a} = \bar{a} - 1000$]
3. IF $\bar{a} > 100$ & $\bar{b} > 100$ THEN
$\bar{a} = 1000 - \bar{a}$; $\bar{b} = 1000 - \bar{b}$;
$flag_3 = 1$
4. IF ($\bar{a} > 10$ & $\bar{b} > 100$) THEN $\bar{b} = \bar{b} - 100$;
[*Solution* $= \bar{a} \times 100 + \bar{b} \times \bar{a}$]
$flag_4 = 1$
[*If* $\bar{b} > 10$ & $\bar{a} > 100$, *THEN* $\bar{a} = \bar{a} - 100$]
5. IF ($\bar{a} = 1$) THEN $\bar{p} = \bar{b}$ | IF ($\bar{b} = 1$) THEN $\bar{p} = \bar{a}$
GOTO Step 7
6. Perform 2-bit multiplication: $\bar{p} = \bar{a} \times \bar{b}$
7. IF ($flag_4 = 1$) THEN
$\bar{p} = \bar{a} \times 100 + \bar{p}$; $\bar{b} = 100 + \bar{b}$
8. IF ($flag_3 = 1$) THEN
$\bar{p} = \{LHS = 1000 - (\bar{a} + \bar{b}) + \text{carry of } RHS\} | \{RHS = \text{(3-bit)}\bar{p}\}$;
$\bar{a} = 1000 + \bar{a}$; $\bar{b} = 1000 + \bar{b}$
9. IF ($flag_2 = 1$) THEN
$\bar{p} = \bar{a} \times 1000 + \bar{p}$; $\bar{b} = 1000 + \bar{b}$
10. IF ($flag_1 = 1$) THEN
$\bar{p} = \{LHS = 10000 - (\bar{a} + \bar{b}) + \text{carry of } RHS\} | \{RHS = \text{(4-bit)}\bar{p}\}$
11. p = Left shift $\bar{p}$ by n bits
12. Return the product p
13. END.

IV. CONCLUSION

A new reduced-bit multiplication algorithm based on a formula of ancient Indian Vedic mathematics has been proposed. Both the Vedic multiplication formulae, Urdhva tiryakbhyam and Nikhilam, have been investigated in detail. Urdhva tiryakbhyam, being general mathematical formula, is equally applicable to all cases of multiplication. A multiplier architecture based on this Sutra has been developed and is seen

to be similar to the popular array multiplier where an array of adders is required to arrive at the final product. Due to its structure, it suffers from a high carry propagation delay in case of multiplication of large numbers. This problem has been solved by introducing Nikhilam Sutra which reduces the multiplication of two large numbers to the multiplication of two small numbers. The framework of the proposed algorithm is taken from this Sutra and is further optimized by use of some general arithmetic operations such as expansion and bit-shifting to take full advantage of bit-reduction in multiplication. The computational efficiency of the algorithm has been illustrated by reducing a general 4×4-bit multiplication to a single 2×2-bit multiplication operation.

REFERENCES

[1] K. Hwang, *Computer Arithmetic: Principles, Architecture And Design*. New York: John Wiley & Sons, 1979.
[2] M. M. Mano, *Computer System Architecture*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[3] G.-K. Ma, F. J. Taylor, "Multiplier Policies for Digital Signal Processing", *IEEE ASSP Mag.*, vol. 7, no. 1, pp. 6–20, Jan. 1990.
[4] D. Goldberg, "Computer Arithmetic", in *Computer Architecture: A Quantitative Approach*, J.L. Hennessy and D.A. Patterson ed., pp. A1-A66, San Mateo, CA: Morgan Kaufmann, 1990.
[5] A.D. Booth, "A Signed Binary Multiplication Technique", *Qrt. J. Mech. App. Math.,*, vol. 4, no. 2, pp. 236–240, 1951.
[6] G. Goto. "High Speed Digital Parallel Multiplier." U. S. Patent 5 465 226, Nov. 7, 1995.
[7] L. Ciminiera and A. Valenzano, "Low Cost Serial Multiplier for High Speed Specialised Processors", *IEE Proc.*, vol. 135, no. 5, pp. 259–265, Sept. 1988.
[8] D. Ait-Boudaoud, M. K. Ibrahim and B. R. Hayes-Gill, "Novel Pipelined Serial/Parallel Multiplier", *Electron. Lett.*, vol. 26, no. 9, pp. 582–583, April 1990.
[9] R. Gnanasekran, "A Fast Serial-Parallel Binary Multiplier", *IEEE Trans. Comput.*, vol. 34, no. 8, pp. 741- 744, Aug. 1985.
[10] B. K. Tirtha, *Vedic Mathematics*. Delhi: Motilal Banarsidass Publishers, 1965.
[11] 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, pp. 153–158, 2004.
[12] H. Thapliyal and M. B. Srinivas, "High Speed Efficient $N \times N$ Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics", *Enformatika Trans.*, vol. 2, pp. 225-228, Dec. 2004.
[13] H. Thapliyal, R. V. Kamala and M. B. Srinivas, "RSA Encryption/Decryption in Wireless Networks Using an Efficient High Speed Multiplier", in *Proc. IEEE Int. Conf. Personal Wireless Comm. (ICPWC-2005)*, New Delhi, Jan. 2005, pp. 417–420.
[14] H. Thapliyal and M. B. Srinivas, "An Efficient Method of Elliptic Curve Encryption Using Ancient Indian Vedic Mathematics", in *Proc. IEEE MIDWEST Symp. Circuits. Systems*, Cincinnati, Aug. 2005, pp. 826–829.
[15] H. Thapliyal, M. B. Srinivas and H. R. Arabnia , "Design And Analysis of a VLSI Based High Performance Low Power Parallel Square Architecture", in *Proc. Int. Conf. Algo. Math. Comp. Sc.*, Las Vegas, June 2005, pp. 72–76.