Circuits and Systems, 2016, 7, 2593-2602
Published Online July 2016 in SciRes. http://www.scirp.org/journal/cs
http://dx.doi.org/10.4236/cs.2016.79224

Design of an Efficient Binary Vedic Multiplier for High Speed Applications Using Vedic Mathematics with Bit Reduction Technique

S. K. Manikandan[1*], C. Palanisamy[2]

[1]Department of EEE, Velalar College of Engineering and Technology, Erode, India
[2]Department of Information Technology, Bannari Amman Institute of Technology, Sathyamangalam, India
Email: manikandankandasamy@gmail.com

Received 2 April 2016; accepted 20 April 2016; published 27 July 2016

Abstract

Vedic mathematics is the system of mathematics followed in ancient Indian and it is applied in various mathematical branches. The word "Vedic" represents the storehouse of all knowledge. Because using Vedic Mathematics, the arithmetical problems are solved easily. The mathematical algorithms are formed from 16 sutras and 13 up-sutras. But there are some limitations in each sutra. Here, two sutras Nikhilam sutra and Karatsuba algorithm are considered. In this research paper, a novel algorithm for binary multiplication based on Vedic mathematics is designed using bit reduction technique. Though Nikhilam sutra is used for multiplication, it is not used in all applications. Because it is special in multiplication. The remainder is derived from this sutra by reducing the remainder bit size to N-2 bit. Here, the number of bits of the remainder is constantly maintained as N-2 bits. By using Karatsuba algorithm, the overall structure of the multiplier is designed. Unlike the conventional Karatsuba algorithm, the proposed algorithm requires only one multiplier with N-2 bits only. The speed of the proposed algorithm is improved with balancing the area and the power. Even though there is a deviation in lower order bits, this method shows larger difference in higher bit lengths.

Keywords

Karatsuba, Nikhilam Sutra, Bit Reduction, Remainder, Multipliers

How to cite this paper: Manikandan, S.K. and Palanisamy, C. (2016) Design of an Efficient Binary Vedic Multiplier for High Speed Applications Using Vedic Mathematics with Bit Reduction Technique. *Circuits and Systems*, **7**, 2593-2602.
http://dx.doi.org/10.4236/cs.2016.79224

1. Introduction

Vedic Mathematics is the technique used in an Ancient India for solving arithmetical problems mentally and in easier way. It contains 16 formulas and 13 sub-formulas. These sutras are used in solving complex computations, and executing them manually. It is operated on 16 sutras and 13 up-sutras. The algorithms and principles of all sutras were given in [1]. Urdhva Tiryakbhyam Sutra is mainly used for multiplication which means "Vertically and Crosswise". The multiplier based on this sutra is known as Vedic multiplier. It is based on a novel concept of array multiplication. In [2], the design and implementation of Triyakbhyam were done and the speed was compared with Nikhilam sutra, squaring and cubing algorithm.

In [3], the implementation of Arithmetic unit that used the Vedic mathematics algorithm for multiplication was discussed. The arithmetic unit was designed to perform multiplication, addition and subtraction and multiply accumulation operations. The MAC unit uses a fast multiplier built with Vedic Mathematics algorithm. The square, cube algorithms along with Karatsuba algorithm have been discussed in [3], to reduce the multiplications required. In [4], a ROM based multiplier is proposed. One out of two inputs is fixed here. So this method can be used in matrix multiplication and other applications which use constant coefficient multiplication. The faster Vedic multiplier consumes more power than convention ones. In [5], the multiplier based on Nikhilam sutra was presented with modification in 2's complement block and multiplication block. 7-bit encoding technique is employed in the design of multiple radix multipliers. The high speed 32 × 32 bit Vedic multiplier was presented in [6]. The binary implementation using carry save adder was presented in [6] and comparison result was made.

The iterative algorithm for Nikhilam sutra was presented in [7]. It reduces two larger numbers in to two smaller numbers by neglecting zeros from the least significant side and performing bit shifting to complete the multiplication operation. But this method demands for closeness of the multiplicands to nearer to power of 10. This algorithm is same as array multiplier final product can be derived based on array of address. The Urdhava Triyagbhyam Vedic is designed for actual development of multiplication and it is presented in [8]. Here the partial products are generated at the same time. In [8], 128 × 128 bit Vedic multiplier is implemented.

In [9], the Urdhava Vedic multiplier is compared with Booth multiplier to analyse their speed and delay. From the results, it is concluded that Urdhava multiplier is superior in delay and power. The speed is improved to the extent of 32% compared with Booth multiplier. The Urdhava multiplier is designed using standard cell approach. The higher bit multiplier is constructed using lower order bits [10]. In [11], Vedic multiplication is implemented on 8085/8086 microprocessors. In [12], a multiplication algorithm based on Nikhilam Sutra is designed. In [13], high speed and low power Vedic multiplier based on Tridhava approach is designed using BEC (Binary-to Excess-one code converter) based carry select adder. Here, instead of two RCA (Ripple Carry Adder), one BEC and one RCA are used. Based on the carry generation, either RCA or BEC outputs are selected. If Cin = "0", RCA is selected. If Cin = "1", BEC is selected. From this, power optimal high speed multiplier is designed. In [14], the Vedic multiplier is designed using modified Carry Select adder. By this, area is reduced by 40% compared with [13].

In [15], Vedic multiplier is designed using modified square-root carry select adder (MSQRT-CSA). They compared the performance with various adders like CSA, Carry look-ahead adder (CLA), Square root CSA (SQRT-CSA). Compared with other methods, multiplier with MSQRT-CSA is faster. In [16], the comparison between traditional binary multipliers is made.

From the survey, Tridhava multiplier can be used for any range of inputs. Different papers were proposed based on the modification in the adders. Though Nikhilam sutra covers all range of inputs, it is efficient when the multiplicands are closer to the multiple of 10. When it is implemented for binary numbers, normally 2's complement will be taken. The changes are made in the portion of adders. Similarly, Karatsuba algorithm is good for higher order bits. In Karatsuba algorithm, three multipliers are required along with shift operations. But in the proposed method, Karatsuba and Nikhilam sutras are combined. The multiplier required is reduced to one and the number of bit to the multiplier is reduced by two.

2. Principle of Nikhilam Sutra and Karatsuba Algorithm

2.1. Nikhilam Sutra

First, Nikhilam Navatascharam Dashtah sutra means all from 9 and last from 10. The Sutra is well explained for multiplication of decimal numbers. The steps for multiplying decimal numbers using this sutra are given below.

1. The nearest base value of multiplicands is considered. Let us assume the multiplicands are A and B.
2. The remainders of A and B are calculated by subtracting the chosen base value from step 1.
3. The product of remainders of A and B are computed and considered as Right part of final product.
4. The Left part can be computed in three ways. The methods are listed below.

a) The two multiplicands A and B are added and the nearest base value is subtracted from the sum A and B. LP = A + B-10.

b) The Left part is calculated by adding the remainders of A and B with Base value.

c) The remainders of A and B are crossly added with the multiplicands A and B.

5. The Final result is attained by removing the demarcation between LP and RP and concatenating the same.

The remainder is derived using this sutra. Based on the type of the remainder, the algorithm is developed. The two types are:

a) Positive remainder.

b) Negative remainder.

The mathematical expression explaining this sutra is given in Equation 1. Let *A* and *B* are the two input numbers. The product *P* is derived as follows:

$$\begin{aligned} P = A \cdot B &= (x-a)(x-b) \\ &= x(x-a) - b(x-a) \\ &= x(x-a-b) + ab \end{aligned} \tag{1}$$

Here *x* acts as base *B* and it is considered as multiple of 10 depending upon the closeness of multiplicands. The remainders of *A* and *B* are known as "*a*" and "*b*" respectively. The product "*ab*" denotes the Right Part (RP). The Left Part (LP) is represented by the first term in (1).

2.2. Karatsuba Algorithm

The Karatsuba algorithm is suited well for multiplying very large numbers. It is a divide and conquer method, in which the number is divided in to Most Significant half and Least Significant half. The multiplication operations are replaced by addition operations and hence the delay of the algorithm is reduced. This algorithm is more efficient when the number of inputs increase. The algorithm is optimal if width of inputs is more than 16 bits.

The numbers are divided as follows

$$\begin{aligned} A &= 2^{\frac{n}{2}} A_l + A_r \\ B &= 2^{\frac{n}{2}} B_l + B_r \end{aligned} \tag{2}$$

where A_l, B_l and A_r, B_r in Equation 2 are the most significant half and least significant of the numbers *A* and *B* respectively and "*n*" represents the number of bits. Then, the product can be written as,

$$\begin{aligned} P &= \left(2^{\frac{n}{2}} A_l + A_r\right)\left(2^{\frac{n}{2}} B_l + B_r\right) \\ &= 2^n A_l B_l + 2^{\frac{n}{2}}\left(A_l B_r + B_l A_r\right) + A_r B_r \end{aligned} \tag{3}$$

From Equation (3), four multiplications and two shift operations are needed. The number is divided equally into two parts. This method is efficient for higher bit length.

3. Proposed Method

In the proposed method, both Nikhilam sutra and Karatsuba algorithm are combined. Using Nikhilam sutra, the remainder is calculated from the nearest value of base 2 by taking 2's complement. Afterwards, the multiplication is done using Karatsuba algorithm. By doing this, the multiplication required is less.

Let *A* and *B* are the numbers and they can be written using Karatsuba algorithm as,

$$A = 2^{n-1} \pm A_r$$
$$B = 2^{n-1} \pm B_r \tag{4}$$

The remainders are derived using Nikhilam Sutra in such a manner that the number of bits for remainder is always N-2. Four modules are generated for different combination of removed MSB values. The weight of remainder is reduced. Then the product is derived as follows,

$$\begin{aligned} P = AB &= \left(2^{n-1} \pm A_r\right)\left(2^{n-1} \pm B_r\right) \\ &= 2^{2n-2} \pm 2^{n-1} A_r \pm 2^{n-1} B_r \pm A_r B_r \\ &= 2^{n-1} A \pm 2^{n-1} B_r \pm A_r B_r \end{aligned} \tag{5}$$

while comparing (3) and (5), the proposed algorithm requires only one multiplier. And there is no change the shift operation. The multiplier used in the proposed algorithm requires only N-2 bits. Therefore, proposed work reduces the number of multipliers as well as the number of bits of multiplier. The sign will be selected based on the remainder type of *A* and *B*. The example for proposed algorithm is shown below:

$$A = 11_{10} = 1011_2 = 8 + 3$$

$$B = 3_{10?} = 0011 \ \ = 4 - 1$$

$$\begin{aligned} AB &= (4 \times 11) - (3 \times 1) - (8 \times 1) \\ &= 44 - 3 - 8 = 33_{10} \end{aligned}$$

3.1. Algorithm I

Input: A, B

Output: P

Step 1: *A* and *B* are the multiplicands with *N* bits. Their remainders are calculated by removing first two bits and taking 2's complement for the remaining N-2 bits. (N-1, N-2 = 11)

$A = 2^N - r_1$ and $B = 2^N - r_2$

r1 = complement (*A*)

r2 = complement (*B*)

Step 2: Multiplying both remainders using N-2 × N-2 bit multiplier $P_1 = r_1 \times r_2$.

Step 3: Shifting A by N times (P_2)

Step 4: Shifting the remainder r_2 by N times (P_3)

Step 5: The product can be calculated as,

$$\begin{aligned} P = AB &= P_2 + P_1 - P_3 \\ &= \left(2^N \times A\right) + \left(r_1 \times r_2\right) - \left(2^N \times r_2\right) \\ &= A << N + \left(r_1 \times r_2\right) - \left(r_2 << N\right) \end{aligned} \tag{6}$$

The proposed architecture for the multiplier for the MSB values 11 is given in **Figure 1**.

3.2. Algorithm II

Input: A, B

Output: P

Step 1: Calculating remainders for both multiplicands by removing first two bits. If the removed bits are 10, the remainders are determined by taking the numbers without the MSB values. (*i.e.* N-1, N-2 = 10)

$A = 2^{N-1} + r_1$ $r_1 = A$ (without first two bits)

$B = 2^{N-1} + r_2$ $r_2 = B$ (without first two bits)

Step 2: Multiplying both remainders using N-2 × N-2 bit multiplier (P_1).

Step 3: Shifting A by N-1 times (P_2)

Step 4: Shifting the remainder by N-1 times (P_3)

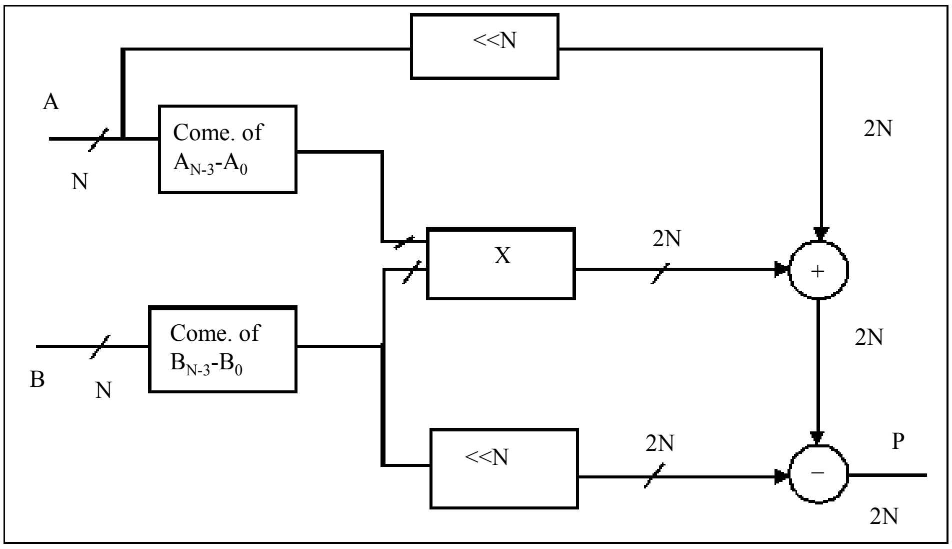


Figure 1. Architecture for Algorithm I.

Step 5: The product can be calculated as,

$$\begin{aligned} P = AB &= P_2 + P_1 + P_3 \\ &= 2^{N-1} \times A + (r_1 \times r_2) + (2^{N-1} \times r_2) \\ &= A << N-1 + (r_1 \times r_2) + (r_2 << N-1) \end{aligned} \tag{7}$$

The architecture for the proposed algorithm is shown in **Figure 2.**

3.3. Algorithm III

Input: A, B N bits

Output: P 2N bits

Step 1: Calculating remainders for both multiplicands by removing first two bits. If the removed bits are 01, the remainders are determined by taking the numbers without the MSB values. (*i.e.* N-1, N-2 = 10)

$A = 2^{N-1} - r_1$ r_1 = complement (A)

$B = 2^{N-1} - r_2$ r_2 = complement (B)

Step 2: Multiplying both remainders using N-2 × N-2 bit multiplier (P_1).

Step 3: Shifting A by N-1 times left side (P_2)

Step 4: Shifting r2 by N-1 times left side (P_3)

Step 5: The product can be calculated as,

$$\begin{aligned} P = AB &= P_2 + P_1 - P_3 \\ &= 2^{N-1} \times A + (r_1 \times r_2) - (2^{N-1} \times r_2) \\ &= A << N-1 + (r_1 \times r_2) - (r_2 << N-1) \end{aligned} \tag{8}$$

The architecture for the proposed algorithm is shown in **Figure 3**.

3.4. Algorithm IV

Input: A, B

Output: P

Step 1: Calculating remainders for both multiplicands by removing first two bits. If the removed bits are 00, the remainders are determined by taking the numbers without the MSB values. (*i.e.* N-1, N-2 = 10)

$A = 2^{N-2} - r_1$ r_1 = complement (A)

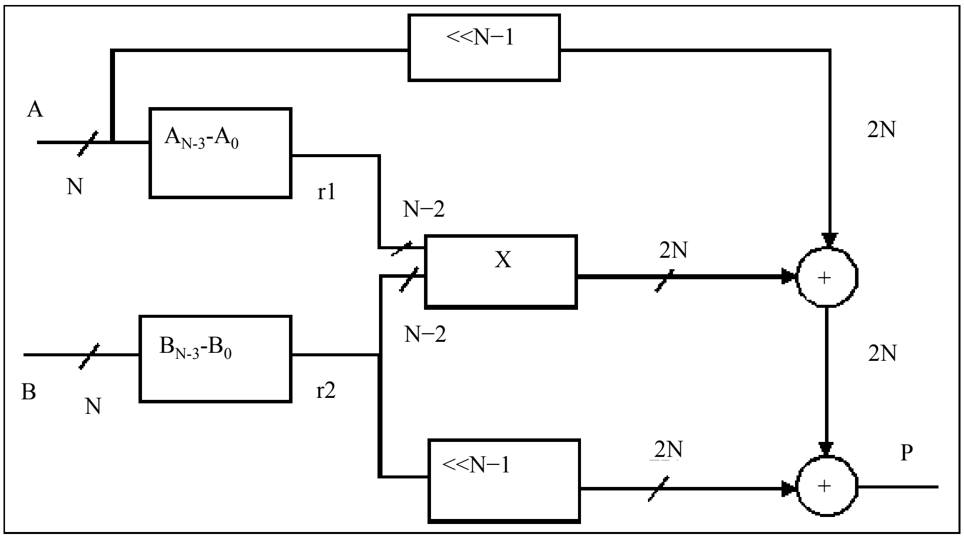


Figure 2. Architecture for Algorithm II.

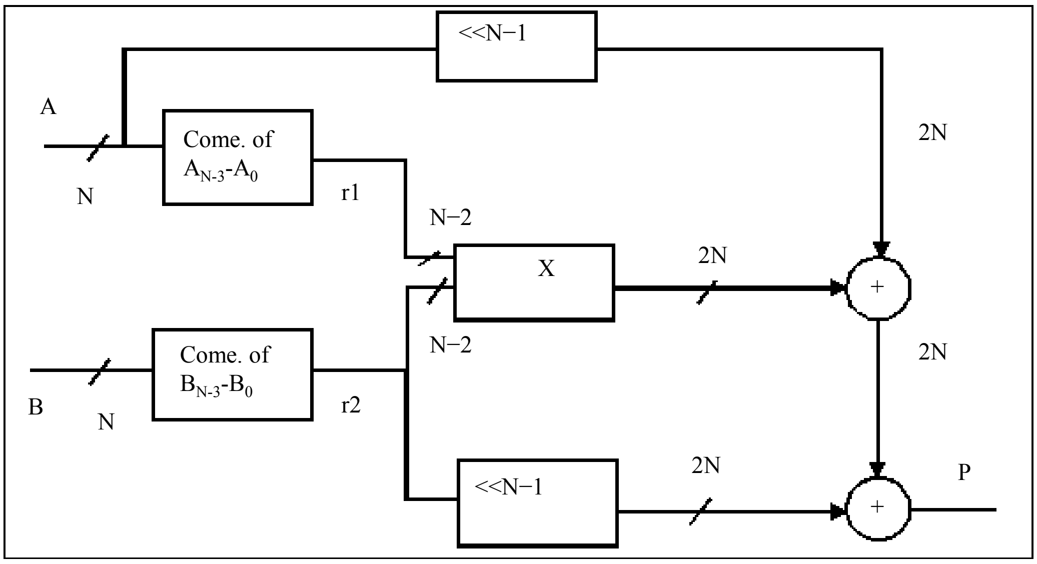


Figure 3. Architecture for Algorithm III.

$B = 2^{N-2} - r_2$ r_2 = complement (B)

Step 2: Multiplying both remainders using N-2×N-2 bit multiplier (P_1).

Step 3: Shifting A by N-2 times (P_2)

Step 4: Shifting r_2 by N-2 times (P_3)

Step 5: The product can be calculated as,

$$\begin{aligned} P = AB &= P_2 + P_1 - P_3 \\ &= 2^{N-2} \times A + (r_1 \times r_2) - (2^{N-2} \times r_2) \\ &= A << N-2 + (r_1 \times r_2) - (r_2 << N-2) \end{aligned} \quad (10)$$

The architecture for this algorithm is shown in **Figure 4**. From the above said four algorithms, the combined architecture is designed to perform multiplication for binary numbers starting with any value. The combined architecture is shown in **Figure 5**. In [12], the detailed explanation is given. For "10" combination, the remainder

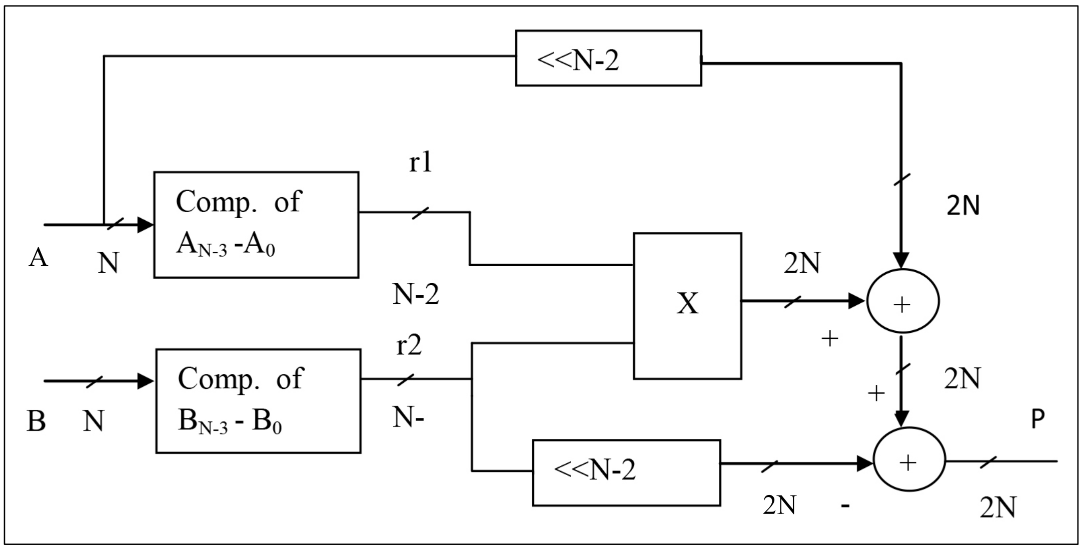


Figure 4. Architecture for Algorithm IV.

derived is positive and for the other combinations the remainders are negative. For negative remainders, the complement should be taken. The input A is shifted N-2, N-1 or N times according to the MSB bits of B. For remainder multiplication, multiplier with N-2 bit is used. The remainder of B is shifted N-1, N-1 or N times. Finally, all the terms added or subtracted according to the algorithm. A simple control circuit is used to select the operation in adder/subtractor module. This architecture is suitable for any input range of inputs. The multiplication is required only when both inputs are nonzero values. The multiplexer is used to reduce the power dissipation when the remainder is zero. In [12], for the multiplication of remainders, various conventional multipliers are used. By comparing other methods, the proposed method gives high speed. Therefore, in this work, proposed algorithm is applied continuously in the algorithm to increase the speed further.

4. Results

AVedic Mathematics is a technique to solve the arithmetic operations easily and mentally. Traditionally, Urdhva Tiryakbhyam Sutra is known as Vedic Multiplier because it covers all range of inputs. It means "vertically and crosswise". In most of the research papers [2]-[9] listed in references concentrated on this type of multiplier. The hardware implementation for binary numbers is also done. This involves RCA and lower order Vedic multipliers. The parallel implementation has also been done [9] [10]. The Nikhilam Sutra is designed for special case. It refers "all from 9". The method is efficient when the multiplicands are closer to the multiple of 10. There is no efficient hardware implementation for Nikhilam Sutra for binary numbers. The algorithm is well defined when both numbers are of with positive or negative remainder. When one number is with positive remainder and the other is with negative remainder, the calculation is different.

Depending upon the value on Right part, correction is made on left part. In the proposed method, all range of inputs can be given. Karatsuba algorithm is efficient for higher order multipliers. Karartsuba multiplier uses three multipliers based on Equation. (3). But in the proposed method, the numbers of multiplier is reduced to one and number of bits of the multiplier is reduced to N-2. Here, Nikhilam sutra and Karatsuba algorithm are combined to get the high speed. The implementation for various Vedic multipliers is done using Xilinx Spartan 3e kit. In the main module, the proposed multiplier is itself used. The comparison with Vedic multiplier for delay is listed in **Table 1**. The comparison with conventional methods is also given in **Table 2**.

The proposed algorithms are written in VHDL and simulated using ModelSim . The simulation result for 32 bit size is shown in **Figure 6**. We implemented the algorithms of existing and proposed work in Xilinx SPARTAN 3E FPGA. Additionally, implementation result in Xilinx SPARTAN 6 for the proposed method is shown in **Table 1**. While comparing with SPARTAN 3e, SPARTAN6 delivers high speed but consumes more power. While comparing delay with other methods, the combination of Nikhilam sutra and Karatsuba algorithm gives minimum delay. From **Table 1**, the delay difference between other methods for higher bits is high. Therefore, this algorithm can be used for high speed applications with wider bit length.

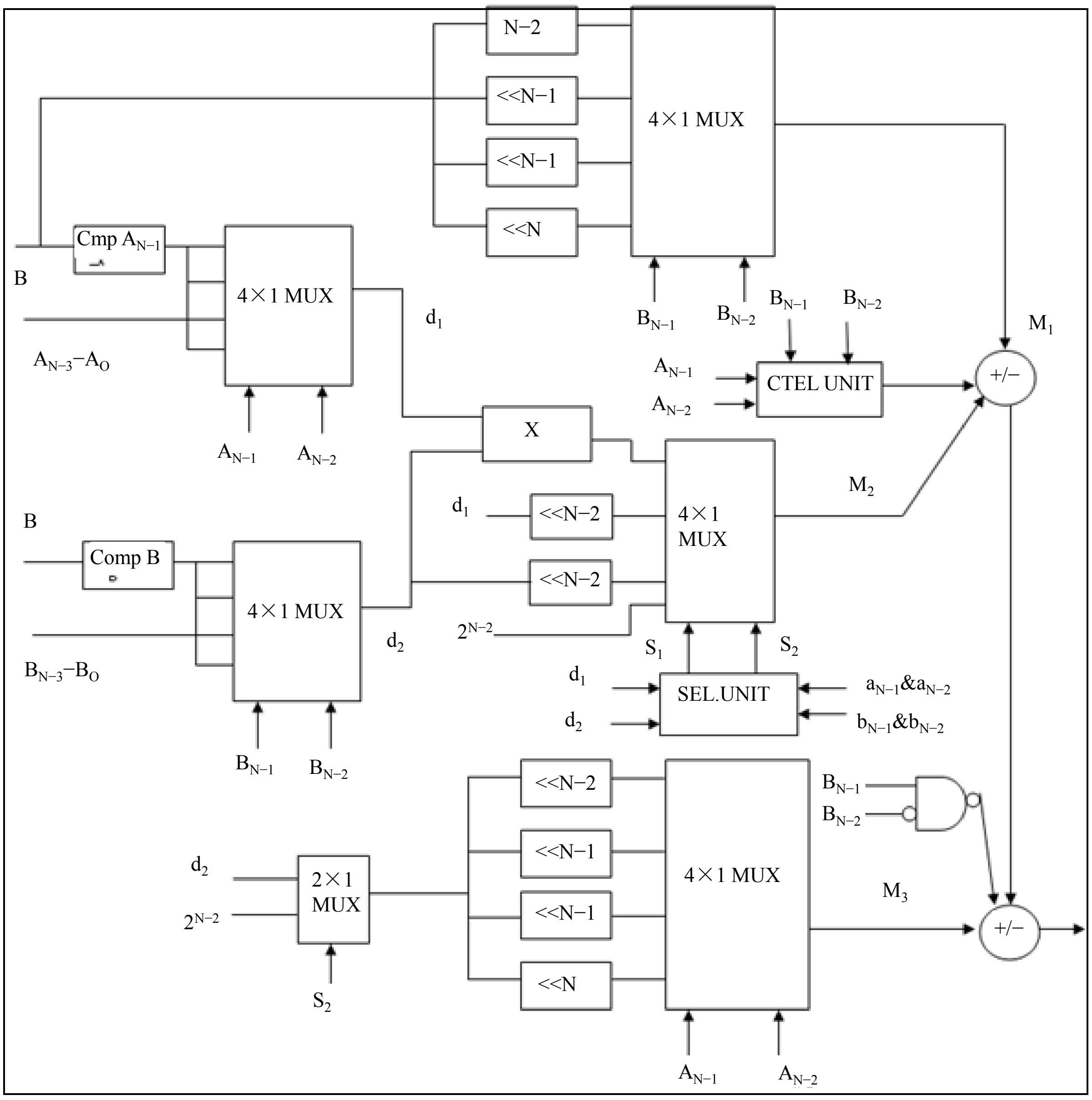


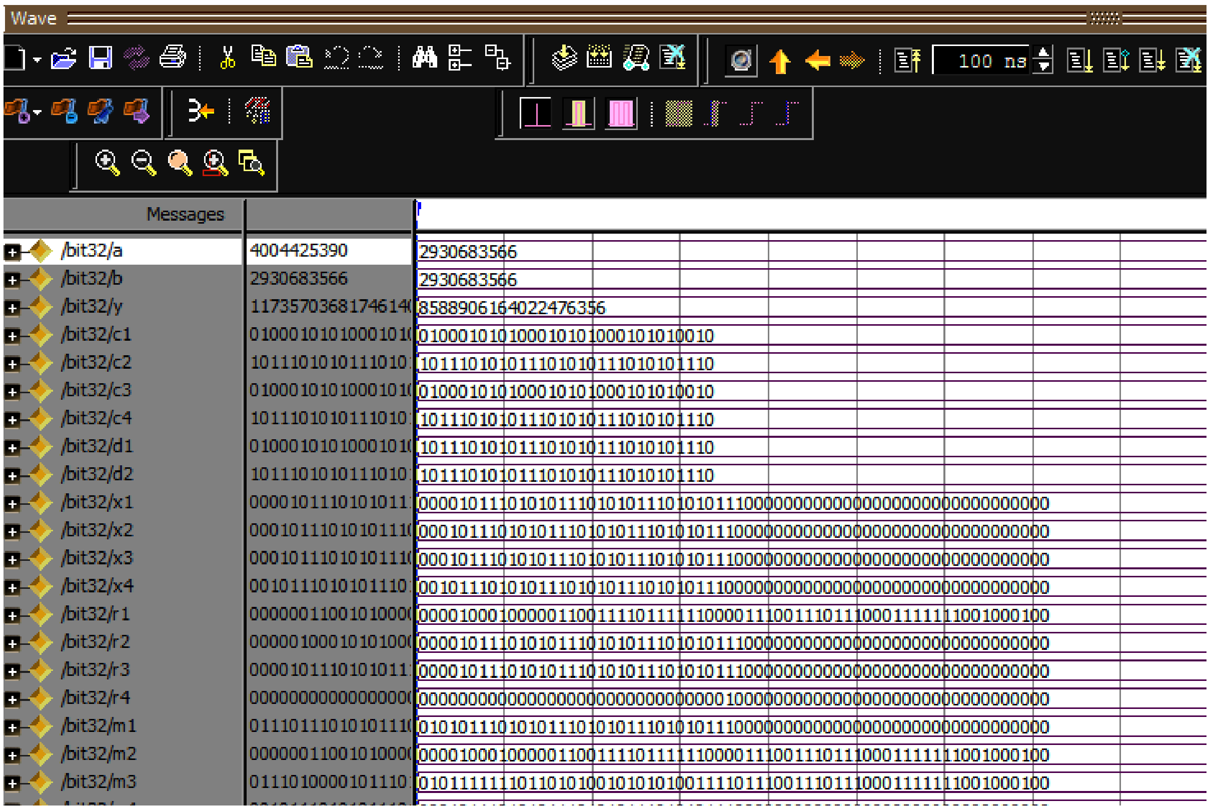
Figure 5. Combined architecture for multiplier.

Table 1. Comparison with other vedic multipliers for delay.

Device	Methods	Delay in nS				
		4 Bit	8 Bit	16 Bit	32 Bit	64 Bit
Xilinx Spartan 3e	Urdhva Tiryakbhyam Sutra [2]	11.412	30.628	49.413	85.359	155.364
	Karatsuba Algorithm [3]	15.231	31.029	46.811	82.834	150.922
	Vedic-Karatsuba Algorithm [3]	10.197	18.005	27.81	49.864	92.448
	Karatsuba-Urdhva Multiplier	15.072	18.492	26.398	42.174	87.235
	Proposed Karatsuba-Nikhilam Multiplier	14.561	16.916	22.147	32.156	64.198
Xilinx Spartan 6	Urdhva Tiryakbhyam Sutra [2]	9.247	26.321	49.413	77.924	142.638
	Karatsuba Algorithm [3]	10.124	28.965	46.811	74.529	138.953
	Vedic-Karatsuba Algorithm [3]	8.621	18.005	27.81	41.953	88.627
	Karatsuba-Urdhva Multiplier	11.632	25.362	26.398	37.439	69.837
	Proposed Karatsuba-Nikhilam Multiplier	10.564	23.624	22.147	28.634	49.391

Table 2. Delay comparison with other conventional binary multipliers.

Methods	Delay in nS				
	4 Bit	8 Bit	16 Bit	32 Bit	64 Bit
Array Multiplier	15.269	31.111	62.437	123.387	240.542
Shift and Add Multiplier	15.677	33.840	63.089	124.112	243.619
Braun Multiplier	13.088	23.331	62.437	127.776	242.325
Wallace Tree Multiplier	12.756	22.863	44.258	87.776	152.584
Proposed Karatsuba-Nikhilam Multiplier	14.561	16.916	22.147	32.156	64.198



```
Messages
/bit32/a    4004425390          2930683566
/bit32/b    2930683566          2930683566
/bit32/y    1173570368174614(   8588906164022476356
/bit32/c1   0100010101000101(   01000101010001010100010101010010
/bit32/c2   1011101010111010    10111010101110101011101010101110
/bit32/c3   0100010101000101(   01000101010001010100010101010010
/bit32/c4   1011101010111010    10111010101110101011101010101110
/bit32/d1   0100010101000101(   10111010101110101011101010101110
/bit32/d2   1011101010111010    10111010101110101011101010101110
/bit32/x1   0000101110101011    0000101110101011101010111010101110000000000000000000000000000000
/bit32/x2   000101110101011 1(  0001011101010111010101110101011100000000000000000000000000000000
/bit32/x3   000101110101011 1(  0001011101010111010101110101011100000000000000000000000000000000
/bit32/x4   0010111010101110    0010111010101110101011101010111000000000000000000000000000000000
/bit32/r1   0000001100101000(   0000100010000011001111011111100001110011101110001111111001000100
/bit32/r2   0000010001010100(   0000101110101011101010111010101110000000000000000000000000000000
/bit32/r3   0000101110101011    0000101110101011101010111010101110000000000000000000000000000000
/bit32/r4   0000000000000000(   0000000000000000000000000000000010000000000000000000000000000000
/bit32/m1   0111011101010111(   0101011101010111010101110101011100000000000000000000000000000000
/bit32/m2   0000001100101000(   0000100010000011001111011111100001110011101110001111111001000100
/bit32/m3   0111010000101110    0101111111011010100101010100111101110011101110001111111001000100
```

Figure 6. Simulation waveform for the multiplier with 32 bit.

5. Conclusion and Future Work

In this research paper, successive approximation of Vedic multiplier is proposed for high speed applications. The algorithm of Karatsuba is modified to reduce the number of multipliers required in the calculation. Instead of splitting the binary number into half, the number is split based on remainder value. The remainder is calculated using Nikhilam Sutra such that the number of bits is reduced to N-2. By combining Nikhilam Sutra and Karatsuba algorithm, the number of bits to the multiplier is reduced. Four modules were created based on remainders. From the results', it is clear that the proposed method produces output faster than other methods. This hybrid multiplier is best suited for multiplying large numbers in high speed applications.

References

[1] Vasantha Kandasamy, W.B. and Smarandache, F. (2006) Vedic Mathematics—"Vedic" or "Mathematics": A Fuzzy & Eutrosophic Analysis. Automaton, Los Angels.

[2] Kumar, V. (2009) Analysis, Verification and FPGA Implementation of Vedic Multiplier with BIST Capability. Thesis, Thapar University, Patiala.

[3] Singh, A. (2010) Design and Hardware Realization of a 16 Bit Vedic Arithmetic Unit. Thesis, Thapar University, Patiala.

[4] Sriraman, L. and Prabakar, T.N. (2012) Design and Implementation of Two Variable Multiplier Using KCM and Vedic Mathematics. *First International Conference on Recent Advances in Information Technology RAIT*, Kolkata, March, 782-787.

[5] Sree Nivas, A. and Kayalvizhi, N. (2012) Implementation of Power Efficient Vedic Multiplier. *International Journal of Computer Applications*, **43**, 21-24.

[6] Kokila, S., Ramadhurai, A. and Sarah, L. (2012) VHDL Implementation of Fast 32X32 Multiplier Based on Vedic Mathematics. *International Journal of Engineering Technology and Computer Applications*, **2**, 46-50.

[7] Vedavathi, K. and Sitamaha Lakshmi, T. (2012) An Iterative Binary Multiplication Algorithm Using Nikhilam Sutra. *International Journal of Current Research*, **4**, 187-190.

[8] Mishra, N. and Haveliya, A. (2013) An Advancement in the N×N Multiplier Architecture Realization via the Ancient Indian Vedic Mathematics. *International Journal of Electronics Communication and Computer Engineering*, **4**, 544-548.

[9] Anju (2013) Performance Comparison of Vedic Multiplier and Booth Multiplier. *International Journal of Engineering and Advanced Technology* (*IJEAT*), **2**, 336-339.

[10] Badal Sharma, S. (2013) Design and Hardware Implementation of 128-Bit Vedic Multiplier. *International Journal for Advance Research in Engineering and Technology*, **1**, 10-14.

[11] Chidgupkar, P.D. and Karad, M.T. (2004) The Implementation of Vedic Algorithms in Digital Signal Processing. *Global Journal of Engineering Education*, **8**, 153-158.

[12] Manikandan, S.K. and Palanisamy, Dr.C. (2016) Design Of Binary Vedic Multiplier Based on Nikhilam Sutra Using Bit-Reduction Algorithm. *Sylwan*, **160**, 106-117.

[13] Bhavani, P.Y., Chokkakula, G., Reddy, S.P. and Samhitha, N.R. (2014) Design of Low Power and High Speed Modified Carry Select Adder for 16bit Vedic Multiplier. *International Conference on Information Communication and Embedded Systems* (*ICICES*), Madras, 27-28 February 2014, 1-4.

[14] Gokhale, Ms.G.R. and Bahirgonde, P.D. (2015) Design of Vedic-Multiplier Using Area-Efficient Carry Select Adder. *International Conference on Advances in Computing*, *Communications and Informatics* (*ICACCI*), Trivandrum, 10-13 August 2015, 576-581.

[15] Goyal, H. and Akhter, S. (2015) VHDL Implementation of Fast Multiplier Based on Vedic Mathematic Using Modified Square Root Carry Select Adder. *International Journal of Computer Applications*, **127**, 24-27.

[16] Chris, Y.H.L., Lo, H.H., Sean, W.F.L. and Nor, H.H. (2010) A Performance Comparison Study on Multiplier Designs. *International Conference on Intelligent and Advanced Systems* (*ICIAS*), Kuala Lumpur, 15-17 June 2010, 1-6.