Asian Journal of Information Technology 15 (18): 3598-3604, 2016
ISSN: 1682-3915

Implementation of Hybrid Vedic Multiplier Nikhilam Sutra and Karatsuba Algorithm for N-bit Multiplier Using Successive Approximation of N-1 Bit Multiplier

[1]M. Nisha Angeline and [2]S. Valarmathy
[1]Department of ECE, Velalar College of Engineering and Technology, Erode, Tamil Nadu, India
[2] Department of ECE, Bannari Amman Institute of Technology,
Sathyamangalam, Tamil Nadu, India

Abstract: Vedic mathematics is the technique to solve complex arithmetic computations. Using this technique, complex problems can be solved easily. Normally, Urdhva Tiryakbhyam Sutra is generally known as Vedic Multiplier. Nikhilam Sutra is a special case in Vedic Mathematics. But there is no proper implementation hardware for Nikhilam Sutra for binary multiplication. The aim of this study is to design hardware for Nikhilam Sutra using Karatsuba algorithm using successive approximation of N-1 bit multiplier. Multipliers are the basic components used in many digital systems, digital signal processing operations and multimedia applications. Digital multipliers are the major source of power dissipation. Multiplications are often implemented with shift-and-add operations. In this study, we propose a method that combines the principles of Nikhilam sutra and Karatsuba sutra for the multiplication of binary numbers. The calculation of remainder is based on Nikhilam sutra using complement method and the weight reduction is carried out in the remainder by removing the MSB. The numerical transformation of the numbers is done by Karatsuba algorithm. For the remainder multiplication, only N-1 bit multiplier is required. Therefore, the algorithm requires only (N-1) x (N-1) bit multiplier for the calculation remainder. By combining both algorithms, the number of multiplier is reduced and also the number of bit for multiplier is also reduced. By applying this modification in the algorithm, strength of the multiplier is reduced. The research is implemented in Xilinx vertex device. The power, area and delay are measured using Cadence tool with 180 and 90nm technology. From the results', the product of delay and area is reduced.

Key words: Nikhilam sutra, multiplication, numerical strength reduction, karatsuba algorithm, FPGA

INTRODUCTION

For any portable and super-fast electronic gadgets, the optimization of power and speed are the important concern. In the modern Digital world, the high speed processors require fastest device for the computation. Multiplication find many applications like digital signal processing, Image processing or arithmetic units in microprocessors. The performance of the system depends on the performance of the multiplier because the multiplier consumes more area and more power. It slows down the system performance. By reducing the delay in the multiplication, the speed of the system will be improved. Researchers focuss the design of the multiplier towards area, speed and power.

The multiplier algorithm researches on the principle of shift and add method. The computation is done either by serially or by parallelly. The parallel computation does not require feedback connections because it uses only combinational circuits for its computation. Depending upon the connectivity in parallel structures, there are two types namely, array multiplication and tree multiplication. The basic array multiplier is designed based on the add and shift algorithm. The main advantage of the array multipier is that it has regular linear structure. Therefore, this multiplier can be implemented using VLSI technology. But the problem arises when the number of bits increase. The carry propogation in ceah stage is considerable here.. Braun multiplier is another example for array multiplication. The modification is done in the portion of carry propagation. Here, the carry is passed to next stage by using carry save adder instead of passing the carry in the same stage using Ripple Carry adder. There is no change in the number of components used for their computarion. Both are linear structures only. They differ only by the way of propagating the delay to the next

Corresponding Author: M. Nisha Angeline, Department of ECE, Velalar College of Engineering and Technology, Erode, Tamil Nadu, India

stage and patial product generation. Further the delar can be reduced by incorporating bypassing technique in row or in column or both in row and column. This technique is discussed in study written by (Anitha *et al*., 2012) to increase the speed.

Wallace (1964) proposed another classification of multiplier structure called "tree based structure" to reduce the delay. The stage delay is reduced in the order of O(log n). it consumes less power when compared with basic array multiplier. A tree of carry save adders are used to reduce the carry propagation delay. Log-depth network is used here for tree network reduction. It is the fastest multier among all but it has irregular structure. And it requires more number of logic gates. In the last stage, adder is used to add the carry saved during partial product generation. Normally, Ripple carry adder is used. Analysis is done using various types of adders in the final satge. If Carry Look-Ahead adder is used, the delay will be used further. The modification can also be done during carry save addition. Different compressors like 4:2, 5:2 and 7:2 compressors for the partial product tree reduction (Abhilash *et al*., 2015). Dada multiplier is another tree based multiplier structure. Here also there are three stages similar to Wallace multiplier. But the partial products are not reduced in every stage.

Vedic multipliers are also developed along with the conventional multipliers. Vedic multipliers are designed based the sutras (formulae) and sub-sutras (sub formulae) (Goyal and Shamim, 2015). From these sutras three sutras are used for multiplication. They are Urdhava Tiryabhyam (vertically and cross wise), Nikhilam sutra and Anupreyana (Gupta *et al*., 2012). These sutras are widely used for decimal multiplication with shorter time. The algorithm for binary multiplication is developed using the same logic. The implementation using FPGA was done in. Goyal and Shamim (2015) designed the Vedic multiplier using carry select adder and square root carry select adder. Normally the strength of the multiplication is reduced by Vedic Multiplier beacause it replaces the multiplications by additions. Therefore while comparing with conventional multipliers, Vedic multipliers need less area and it produces the oputput faster (Somani *et al*., 2012).

In this study, the multiplication is done through the calculation of remainder. The remainder is computed based on Nikhilam Sutra. In this work, the sutra is slightly modified to achieve high speed. The computation is different from the existing method. The remainder is determined by finding 2's complement for the number. There is a restriction in the existing method for the usage of input range. But here there is no limit for input. Here, the length of remainder is reduced by 1 bit. The multiplier is required to multiply the remainders derived from the input numbers. The reduction in the component reduces the delay and hence speed is optimized.

MATERIALS AND METHODS

Review of vedic multiplier: Nikhilam sutra is a sutra suitable for multiplication. The sutra is explained for the decimal number. This sutra is special case in multiplication. The sutra is efficient when the number is nearer to 9 or 10. The remainder is calculated from the nearest base number. The positive remainder is derived when the number exceeds the nearest base value. When the number is less than the base value, the remainder will be negative (Singh and Sasamal, 2015) presented a comparative analysis study of binary vedic multiplier using recovery logic by the usuage of CMOS, PFAL and ECRL. PFAL technique consumes less power as compared to ECRL and CMOS designs. PFAL based Vedic multiplier consumes less power even at higher frequencies. But ECRL based Vedic multiplier is good at low frequency as frequency increases its power saving efficiency decreases (Goswami and Pandey, 2014) designed energy efficient Vedic multiplier on FPGA using thermal aware design. They implemented Vedic multiplier on 90nm FPGA, 65nm FPGA and 40nm FPGA respectively by reducing the temperature from 40-20°C (Xiaoping *et al*., 2014) proposed a multiplier using Redundant number system. Arish and Sharma (2015) proposed a multiplier by including Urdhva-Tiryagbhyam algorithm (Vedic mathematics) at Least Significant Bit side and Karatsuba algorithm at Most Significant side.

Priciples of Nikhilam Sutra: The sutra explains the multiplication principles through decimal numbers. Thapliyal and Srinivas (2004) explains the algorithm through different examples. The remainders are initially calculated from the given numbers. From the type of remainders derived from the previous step, the Most Significant portion is computed. Jin Hyukkim and Thapliyal and Srinivas (2004) have shown all possible examples in their work. The steps to calculate the Most Significant portion is given below:

- The two multiplicands (M and N) are added and the base value is subtracted from the above sum L = M+N-B

- The two remainders m and n are added together with the base value L = m+n+B
- The remainders are crossly added with the multiplicands, i.e., L = M+n or L = N+m

The problem arises in the sutra when the input is not taken from the specified range. The correction logic is required when the remainder are with different type. The main advantage is that it reduces the strength of the multiplication.

Karatsuba algorithm: It is useful to multipliy the numbers with higher bit length. It works on the principle of divide-and-conquer method in which the numbers is divided into two parts. The two parts may be equal or not. For binary numbers generally, it is split into Most significant half and Least Significant half. The multiplication is performed after the number splitting. Karatsuba algorithm replaces the multiplication by addition operations. Thus it reduces the number of multiplication required. Addition operation requires less area while compared with multiplication and hence the speed will be improved:

$$X = 2^{\frac{n}{2}}X_h + X_r$$

$$Y = 2^{\frac{n}{2}}Y_h + Y_r \quad (1)$$

where, X_h, Y_h and X_r, Y_r are Most significant half and Least Significant half of X and Y and n is the number of bits (Arish and Sharma, 2015). The product is calculated as:

$$P = XY = (2^{\frac{n}{2}}X_h + X_r)(2^{\frac{n}{2}}Y_h + Y_r)$$

$$= 2^n X_h Y_h + 2^{\frac{n}{2}}(X_h Y_r + X_r Y_h) + X_r Y_r \quad (2)$$

From Eq. 2 , four multiplications and 2 shift operations are needed. For N bit multiplication, the algorithm requires four number of N/2 multipliers.

Relating Nikhilam Sutra with Karatsuba Algorithm: The problem arises in the exiting method when the given numbers are with different base value. In the proposed paper, the remainder is calculated from the nearest base value 2 (i.e., 4,8,16,….) for binary multiplication based on Nikhilam sutra. The multiplication of two remainders is calculated from N-1 bit multiplier. Here the strength of the multiplication is reduced by reducing the maximum weight (2^{N-1}) of the number. So that the Karatsuba algorithm is modified such that the remainder has N-1 bit. The concept is explained through an example. Hence the computational complexity of the multiplier can be reduced. Let X and Y are the numbers. They are divided as per Karatsuba algorithm and they are given in Eq. 3:

$$X = 2^{N-1} \pm X_r$$

$$Y = 2^{N-1} \pm Y_r \quad (3)$$

The weight of the MSB (X_{N-1} and Y_{N-1}) has to be reduced to 2^{N-2}. The weight reduction is achieved by deriving remainder using Nikhilam Sutra. Unlike Karatsuba algorithm, the number is divided into two parts, namely the maximum weight of the number and the remainder (r). If the number is $>2^{N-1}$, the remainder will be positive. If the number is $<2^{N-1}$, the remainder will be negative. Three algorithms are developed based on the remainder type. They are, Mode I- Both are positive remainders, Mode II-Both are negative remainders and Mode III-Remainders with different type. The number is split based on 2^{N-1}. The negative remainders are derived by taking 2's complement. For example, considering the numbers:

$$X = 2^{N-1} - \left(2^{N-1} - X\right) \quad (4)$$

The second term in Eq. 4 is known as negative remainder. The negative remainder is derived by complementing the number if the number is $<2^{N-1}$. If the number is $>2^{N-1}$, the remainder will be positive and it will be derived by taking the number form X_{N-1}-X_0. The product can be derived by

$$P = XY = (2^{N-1} \pm X_r)(2^{N-1} \pm Y_r)$$

$$= 2^{N-1}2^{N-1} \pm 2^{N-1}X_r \pm 2^{N-1}Y_r \pm X_r Y_r$$

$$= 2^{N-1}(2^{N-1} \pm X_r) \pm 2^{N-1}Y_r \pm X_r Y_r$$

$$= 2^{N-1}X \pm 2^{N-1}Y_r \pm X_r Y_r \quad (5)$$

From Eq. 5 multiplier is required to derive the product XrYr with N-1 bits. The two terms are based on shift operation by N-1 bit. The proposed architecture and its methodology are explained in the next section.

Proposed hybrid Vedic multiplier: In this study, the proposed multiplier algorithm is derived based on the type of the remainders and their architectures are given. The results are proven theoretically in this study. Three modes are discussed in detail below:

- Mode I- Both numbers are greater than 2^{N-1} (Positive Remainders)
- Mode II- Both numbers are less than 2^{N-1} (Negative Remainders)
- Mode III- Only one number is greater than 2^{N-1} (Remainders with different type)

Algorithm for mode 1:
Input : A, B (N bits)
Output : P (2N bits)

Step 1: Given A and B are greater than 2^{N-1}. The positive remainders are derived by taking the number from $A_{N-2},A_{N-3},\ldots\ldots\ldots A_1,A_0$ and $B_{N-2},B_{N-3},\ldots\ldots\ldots B_1,B_0$ (considering c1and c2)

Step2: Multiply the remainders c1 and c2 i.e. m1=c1*c2 using N-1 bit multiplier.

Step3: Shift the input A left side by N-1 times. (m2=A<<N-1).

Step4: Shift the remainder of B, c2 by N-1 times (m3=c2<<N-1)

Step5: Add all the components to derive the product P = m1+m2+m3

The architecture for Mode-I is shown in Fig.1. Here both remainders are positive. The product of the numbers is calculated as follows

$$P=AB= 2^{N-1}*A+c_1*c_2+2^{N-1}*c_2$$

$$= (A<<N-1)+(c1*c2)+(c2<<N-1) \quad (6)$$

Algorithm for mode 2:
Input : A, B (N bits)
Output : P (2N bits)

Step 1: Given A and B are less than 2^{N-1}. Complement A and B to derive remainders. (Consider c1 and c2)

Step2: Multiply the remainders c1 and c2 m1 = c1*c2.

Step3: Shift the input A left side by N-1 times. (m2=A<<N-1).

Step4: Shift the remainder of B, r2 by N-1 times (m3 = c2<<N-1)

Step5: Add all the components to derive the product P=m1+m2-m3

The architecture for Mode-II architecture for negative remainders is shown in Fig. 2. Here, both remainders are negative. The product of the numbers is calculated as follows:

$$P = AB = 2^{N-1}*A+c_1*c_2-2^{N-1}*c_2$$

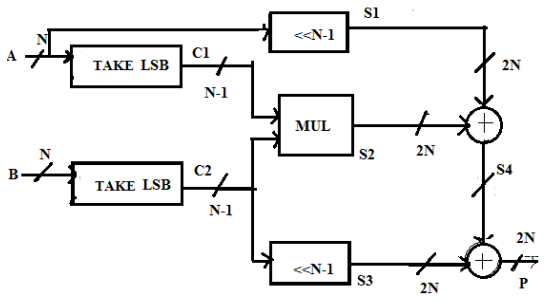

Fig. 1: Proposed multiplier for Mode I

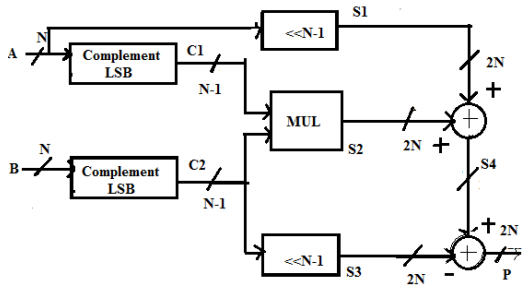

Fig. 2: Proposed multiplier for Mode II

$$= (A<<N-1)+(c1*c2)-(c2<<N-1) \quad (7)$$

Algorithm for mode 3:
Input : A, B (N bits)
Output : P (2N bits)

Step 1: Given A or B may be less than 2^{N-1}. A is considered to be less than B. If B>A, interchange A and B.

Step 2: If $A_{N-1} = 1$, take from $A_{N-2},A_{N-3},\ldots\ldots A_1,A_0$ or complement A to derive remainder. (Consider c1). If $B_{N-1} = 1$, take from $B_{N-2},B_{N-3},\ldots\ldots B_1,B_0$ or complement B to derive remainder.(c2)

Step3: Multiply the remainders c1 and c2 , m1 = c1*c2.

Step4: Shift the input A left side by N-1 times (m2 = A<<N-1).

Step5: Shift the remainder of B, r2 by N-1 times (m3 = c2<<N-1)

Step6: Add all the components to derive the product P = m1-m2+m3

The architecture for Model 3 is shown in Fig. 3. Here one remainder is positive and the other is negative. The product of the numbers is calculated as follows:

$$P = AB = 2^{N-1}*A-c_1*c_2+2^{N-1}*c_2$$

$$=(A<<N-1)-(c1*c2)+(c2<<N-1) \quad (8)$$

The input multiplexer is used to select the complement value if the number is $<2^{N-1}$. If $A_{N-1} = 0$ the

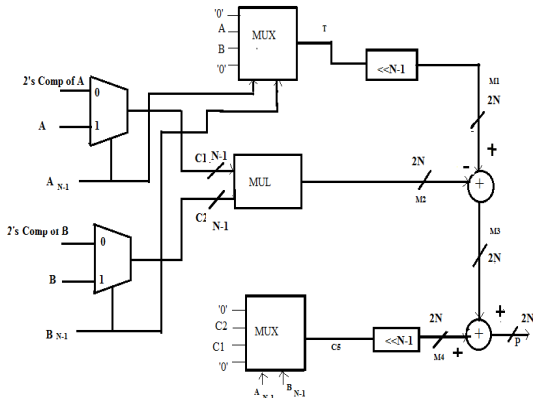

Fig. 3: Proposed multiplier for Mode 3

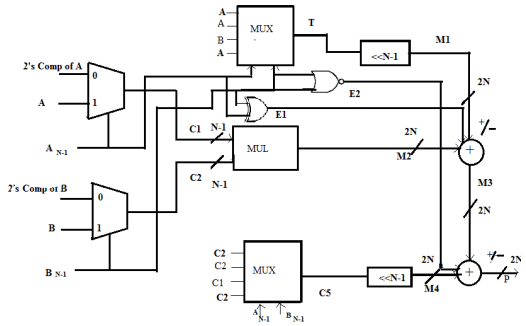

Fig. 4: Proposed combined multiplier

complement value will be selected or the LSB values of A will be selected. The other multiplexers are used to interchange the inputs A and B and accordingly the remainders of A and B i.e., c1 and c2. The multiplier unit is used to multiply the remainders A_rB_r. M1 is derived by shifting the value of A by N-1 bits. (i.e ., $2^{N-1}A_r$). M4 is the term that represents the multiplication of $2^{N-1}B_r$. The adder/subtractor is designed to select the operation based on the type of remainders.

The combined structure is shown in Fig.4. Using the combined structure, the number in any mode can be calculated. This structure is similar to the structure shown in Fig. 3. Here the control signal to select adder/subtractor is generated by simple logic gate. When any one of the remainders is negative, the product c1c2 will be negative. Therefore, m2 will be negative. A simple Ex-or gate is used to produce a control signal for adder/subtractor unit. The problem of existing method is solved by this proposed combined multiplier. While comparing with exiting Karatsuba algorithm, the proposed multiplier requires only one multiplier with N-2 bits instead of four multipliers with N/2 bits. In, subtractor is used to derive the remainders. In this research, the complement is used.

RESULTS AND DISCUSSION

For comparison, both conventional methods and the proposed multiplier are considered for different bit values. And it is functionally verified using MENTOR GRAPHICS tool. The VHDL codes are implemented in Xilinx Spartan 3e and Virtex FPGA. The power analysis is also done using Mentor Graphics tool. The simulation result is shown in Fig. 5. The array multiplier is widely used due to its linear structure. It has identical cells generating partial products simultaneously and accumulating same time. Pipelining can be done easily at each level. Here delay is logarithmically proportional to the bit size of multiplicands. But it requires large amount of gates. For minimum number of bits, this method will be efficient. Here, for comparison four conventional methods considered. While comparing with Karatsuba algorithm, in the proposed method, the multiplier required is minimum. And the number of bits is also reduced. Only N-1 bit multiplier is required. In the proposed algorithm various N-1 bit multipliers were used in the implementation of proposed method.

While comparing delay with other methods, Vedic multiplier has minimum delay. The speed is improved. While comparing the power with other conventional methods, Wallace and Braun multipliers consume less power. Therefore, for low power application, this combination will produce better result. The deviation is high for higher order bits for other methods. The array multiplier and shift-and-add multipliers are widely used in most of the applications. From Eq. 5, the Karatsaba algorithm requires 4 multiplications but the proposed algorithm requires only one multiplication and also it requires only N-1 bit. In used different multipliers in the proposed work. Here, the proposed Vedic multiplier is used in the main Vedic multiplier.

While comparing area among all multipliers, Vedic multiplier requires minimum area. The table shows the comparson result among conventional multipliers. The standard bit sizes are considered. From the results', it can be concluded that the proposed method can be used for high speed applications. The overall performance of the multiplier depends on the N-1 bit multiplier used. Wallace also works in high speed but the power is more compared with Vedic multiplier. The results obtaining from the successive approximation of proposed multiplier is shown in Table 1-4.

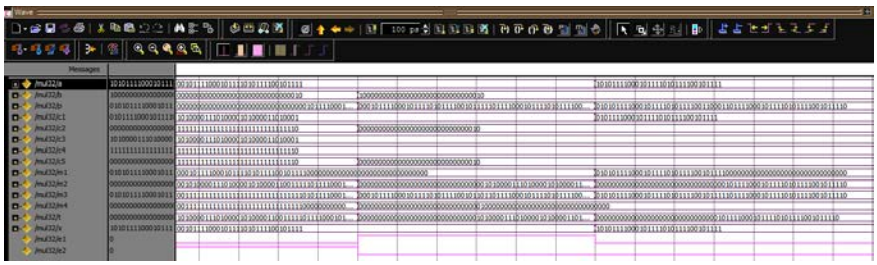

Fig. 5: Simulation result for multiplier with bit size 32

Table 1:Delay comparison with various methods using Xilinx Spartan 3e

	Delay in nS			
Methods	4BIT	8BIT	16BIT	32BIT
Array multiplier	15.269	31.111	62.437	123.387
Shift and add multiplier	15.677	33.840	63.089	124.112
Braun multiplier	13.088	23.331	62.437	127.776
Wallace tree multiplier	12.756	22.863	44.258	87.776
Vedic multiplier (urdhva)	11.752	21.564	41.684	82.289
Proposed multiplier	7.925	15.634	31.216	72.592

Table 2: Power comparison with various methods

	Power in mW			
Methods	4BIT	8BIT	16BIT	32BIT
Array multiplier	298	312	368	440
Shift and add multiplier	310	368	501	636
Braun multiplier	113	149	406	706
Wallace tree multiplier	113	154	375	706
Vedic multiplier	123	142	250	489
Proposed multiplier	120	137	235	469

Table 3: Delay comparison of proposed multiplier using spartan and vertex devices

	Delay in nS			
Methods	4BIT	8BIT	16BIT	32BIT
Xilinx spartan	7.925	15.634	31.216	72.592
Xilinx vertex	6.543	13.578	28.957	68.485

Table 4: Area comparison with various methods

	Area (LUTs)			
Methods	4BIT	8BIT	16BIT	32BIT
Array multiplier	30	125	511	2033
Shift and add multiplier	26	123	508	2044
Braun multiplier	33	77	294	1194
Wallace tree multiplier	27	122	512	2077
Vedic multiplier	20	79	501	1932
Proposed multiplier	11	61	439	1858

CONCLUSION

In this study, the hybrid binary Vedic multiplier based on Nikhilam Sutra and Karatsuba algorithm was presented. The modification of binary multiplier was done in the calculation of remainder. The remainder calculation is done by Nikhilam sutra. By using Karatsuba, the multiplier required is reduced. And the number of bits for the multiplier is reduced to N-1 bit. Therefore, the interconnection delay and computation time are reduced. The speed and the area are optimized using this modified Vedic multiplier. The performance of the modified multiplier depends only on the multiplier and the other operations are shift operations only. The results are implemented in Xilinx Spartan 3e and Virtex 6 kit. sFor high speed applications with wide range of bits, this method is suitable.

ACKNOWLEDGMENTS

The researchers wish to thank the helpful comments and suggestions from my colleagues in Velalar College of Engineering and Technology, Erode and the members in Bannari Amman Institute of Technology, Sathyamangalam for their support to complete this work.

REFERENCES

Abhilash, R., I.K. Raju, G. Chary and S. Dubey, 2015. Area-power efficient vedic multiplier using compressors. Proceedings of the 2015 International Conference on Electrical Electronics Signals Communication and Optimization (EESCO), January 24-25, 2015, IEEE, Visakhapatnam, India, ISBN: 978-1-4799-7676-8, pp: 1-5.

Anitha, R., A. Nelapati, W.L. Jesima and V. Bagyaveereswaran, 2012. Comparative study of high performance Brauns multiplier using FPGA. IOSR. J. Electron. Commun. Eng. (IOSRJECE.), 1: 33-37.

Arish, S. and R.K. Sharma, 2015. An efficient binary multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm. Proceedings of the 2015 Global Conference on Communication Technologies (GCCT), April 23-24, 2015, IEEE, Thuckalay, India, ISBN: 978-1-4799-8552-4, pp: 192-196.

Goswami, K. and B. Pandey, 2014. LVCMOS Based thermal aware energy efficient vedic multiplier design on FPGA. Proceedings of the 2014 International Conference on Computational Intelligence and Communication Networks (CICN), November 14-16, 2014, IEEE, Bhopal, India, ISBN: 978-1-4799-6928-9, pp: 921-924.

Goyal, H. and A. Shamim, 2015. An advancement in the N×N multiplier architecture realization via the Ancient Indian Vedic mathematic. Int. J. Comput. Appl., 127: 24-27.

Gupta, A., M. Utsav and K. Vinod, 2012. A novel approach to design high speed arithmetic logic unit based on ancient vedic multiplication technique. Int. J. Modern Eng. Res. (IJMER.), 2: 2695-2698.Kokila, S., R. Ramadhurai, L. Sarah, 2012. VHDL implementation of fast 32x32 multiplier based on Vedic mathematics. Int. J. Eng. Technol. Comput. Appl., 2: 46-50.

Singh, S. and T.N. Sasamal, 2015. Design of vedic multiplier using adiabatic logic. Proceedings of the 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), February 25-27, 2015, IEEE, Noida, India, ISBN: 978-1-4799-8432-9, pp: 438-441.

Somani, A., D. Jain, J. Sanjay, M.V. Kumku and K. Swati, 2012. Compare vedic multiplier with conventional hierarchical array of array multipliers. Int. J. Comput. Technol. Electron. Eng. (IJCTEE), 2: 52-55.

Thapliyal, H. and M.B. Srinivas, 2004. High speed efficient N x N Bit parallel hierarchical overlay multiplier architecture based on ancient Indian vedic mathematics. Enformatika Trans., 2: 225-228.

Wallace, C.S., 1964. A suggestion for a fast multiplier. IEEE Trans. Electr. Comput., EC-13: 14-17.

Xiaoping, C., H. Wei, C. Xin and W. Shumin, 2014. A new redundant binary partial product generator for fast 2n-Bit multiplier design. Proceedings of the 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), December 19-21, 2014, IEEE, Chengdu, China, ISBN: 978-1-4799-7980-6, pp: 840-844.