Vedic Mathematics Based 32-Bit Multiplier Design for High Speed Low Power Processors

P. Saha[1], A. Banerjee[2], A. Dandapat[3], P. Bhattacharyya[1*]
[1]Bengal Engineering and Science University, Shibpur, Howrah-711103, India
[2]Dept. of ECE, JIS College of Engineering,Kalyani, Nadia-741235, India
[3]Department of ETCE.Jadavpur University, Kolkata-700032, India
*Corresponding author: Tel.: +913326684561; fax: +913326682916
E-mail: pb_etc_besu@yahoo.com

Submitted: May 9, 2011 ***Accepted: May 25, 2011*** ***Published: June 1, 2011***

Abstract- Vedic Mathematics is the ancient methodology of Indian mathematics which has a unique technique for arithmetic computations based on 16 Sutras (Formulae). Transistor level implementation (ASIC) of Vedic Mathematics based 32-bit multiplier for high speed low power processor is reported in this paper. Simple Boolean logic is combined with 'Vedic' formulas, which reduces the partial products and sums generated in one step, reduces the carry propagation from LSB to MSB. The implementation methodology ensure substantial reduction of propagation delay in comparison with Wallace Tree (WTM), modified Booth Algorithm (MBA), Baugh Wooley (BWM) and Row Bypassing and Parallel Architecture (RBPA) based implementation which are most commonly used architectures. The functionality of these circuits was checked and performance parameters like propagation delay and dynamic power consumption were calculated by spice spectre using standard 90nm CMOS technology. The propagation delay of the resulting 32×32 multiplier was only ~1.06 us and consumes ~132 uW power. The implementation offered significant improvement in terms of delay and power from earlier reported ones.

Index terms: Vedic Formulae, Multiplication, High Speed, Low Power, Latency.

1. INTRODUCTION

A multiplier is one of the key hardware blocks in most digital signal processing systems, like frequency domain filtering (FIR and IIR), frequency-time transformations (FFT), Correlation, Digital Image processing etc [1,4]. With advances in technology, many researchers have tried to design multipliers which offer either of the following- high speed, low power consumption, regularity of layout and hence less area or even combination of them in multiplier.

Many research efforts have been devoted to implement high speed and reducing the power dissipation of multipliers like Wallace Tree Multiplier (WTM) [5], Modified Booth Array (MBA) [6], Baugh Wooley Multiplier (BWM) [7] and Row Bypassing and Parallel Architecture (RBPA) [8] based multiplier etc. The basic idea behind all these attempts was the fast implementation of the addition of the partial products. For this purpose, the carry-save addition (CSA) technique has been extensively used. In this technique, intermediate results are always in a redundant form of two numbers [9]. The carry-select-adder (CSA)-based radix multipliers, which have lower area overhead, employ a greater number of active transistors for the multiplication operation and hence consume more power [10]. Among other multipliers, shift-and-add multipliers have been used in many other applications for their simplicity and relatively small area requirement [11]. Higher-radix multipliers are faster but consume more power since they employ wider registers, and require more silicon area due to their more complex logic [12].

In algorithmic and structural levels, a lot of multiplication techniques had been developed to enhance the efficiency of the multiplier; which encounters the reduction of the partial products and/or the methods for their partial products addition, but the principle behind multiplication was same in all cases. The Vedic mathematics approach is totally different and considered very close to the way a human mind works. Vedic Mathematics is the ancient system of Indian mathematics which has a unique technique of calculations based on 16 Sutras (Formulae). “Urdhva-tiryakbyham” is a Sanskrit word means vertically and crosswise formula is used for smaller number multiplication. “Nikhilam Navatascaramam Dasatah” also a Sanskrit term indicating “all from 9 and last from 10”, formula is used for large number multiplication. All these formulas are adopted from ancient Indian Vedic Mathematics. Mehta et al. [13] have been proposed a multiplier design using “Urdhva-tiryakbyham” sutras, which was adopted from the Vedas. The

formulation using this sutra is similar to the modern array multiplication, which also indicating the carry propagation issues. A multiplier design using "Nikhilam Navatascaramam Dasatah" sutras has been reported by Tiwari et. al [14] in 2009, but he has not implemented the hardware module for multiplication. Recently Saha et. al [15] has been reported a multiplier based on the same principle of "Nikhilam Navatascaramam Dasatah" sutra for special types multiplier design of same bases, but he has not extended his work for general purpose multiplier design.

In this work we formulate this mathematics for designing the 32×32 bit multiplier architecture in transistor level with two clear goals in mind such as: i) Simplicity and modularity multiplications for VLSI implementations and ii) The elimination of carry propagation for rapid additions and subtractions. By employing the Vedic mathematics, an (N×N) bit multiplier implementation was transformed into one small number multiplication, one addition/subtraction and shifting operations. "Urdhva-tiryakbyham" method is used for the implementation of small number multiplication, "Nikhilam Navatascaramam Dasatah" and "Urdhva-tiryakbyham" methodology is used for generating the whole (N×N) bit multiplier. The multiplier is fully parameterized, so any configuration of input and output word-lengths could be elaborated. Transistor level implementation for performance parameters such as propagation delay, dynamic leakage power and dynamic switching power consumption calculation of the proposed method was calculated by spice spectre using 90 nm standard CMOS technology and compared with the other design like Wallace Tree Multiplier (WTM) [5], Modified Booth Array (MBA) [6], Baugh Wooley Multiplier (BWM) [7] and Row Bypassing and Parallel Architecture (RBPA) [8]. The calculated results revealed (32×32) bit multiplier have propagation delay only ~1.06 us and consumes ~132 uW dynamic switching power.

2. MATHEMATICAL FORMULATION OF VEDIC SUTRAS

The gifts of the ancient Indian mathematics in the world history of mathematical science are not well recognized. The contributions of saint and mathematician in the field of number theory, 'Sri Bharati Krsna Thirthaji Maharaja', in the form of Vedic Sutras (formulas) [11] are significant for calculations. He had explored the mathematical potentials from Vedic primers and showed that the mathematical operations can be carried out mentally to produce fast answers using the Sutras.

In this paper we are concentrating on “Urdhva-tiryakbyham”, and “Nikhilam Navatascaramam Dasatah” formulas and other formulas are beyond the scope of this paper.

2.1 “Nikhilam Navatascaramam Dasatah” Sutra

Nikhilam sutra means “all from 9 and last from 10”. Mathematical description of this sutra can be formulated as:

Assuming A and B are two n-bit numbers to be multiplied and their product is equals to Z.

$A = \sum_{i=0}^{n-1} A_i\, 10^i$ where $a_i \in \{0,1,\dots\dots\dots 9\}$ (1)

$B = \sum_{i=0}^{n-1} B_i\, 10^i$ where $b_i \in \{0,1,\dots\dots\dots 9\}$ (2)

Multiplication Rule:

Z= AB (3)

Equation (3) can be reformulated as by adding and subtracting the term $10^{2n}+10^n(A+B)$ in the right hand side

$$P = AB + 10^{2n} + 10^n(A+B) - 10^{2n} - 10^n(A+B) \quad (4)$$

$$= \{10^n(A+B) - 10^{2n}\} + 10^{2n} - 10^n(A+B) + AB \quad (5)$$

Equation no (5) can be derived for both the numbers if the number is greater than the base or less than the base.

If the number is greater than the base:

$$= 10^n\{(A+B) - 10^n\} + \{(10^n - A)(10^n - B)\} \quad (6)$$

If the number is less than the base:

$$= 10^n\{(A+B) - 10^n\} + \{(10^n - A)(10^n - B)\} \quad (7)$$

If the number is less than the base:

$$= 10^n\{(A+B) - 10^n\} + \{(A - 10^n)(B - 10^n)\} \quad (8)$$

$$= 10^n\{A - (10^n - B)\} + \{(10^n - A)(10^n - B)\} \quad (9)$$

$$= 10^n(A - \bar{B}) + (\bar{A}\,\bar{B})\} \quad (10)$$

Where $\bar{A}$ and $\bar{B}$ are the 10^n’s complement of A and B.

Example of Multiplication using “Nikhilam Navatascaramam Dasatah” Sutra:

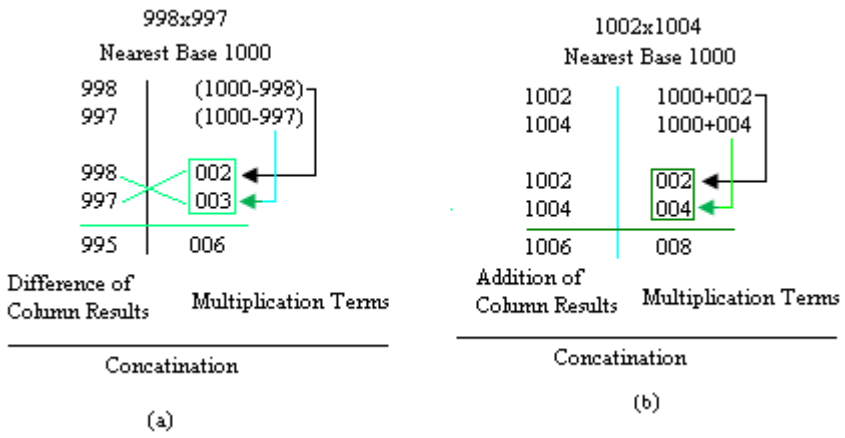


Figure 1. Procedure of multiplication using "Nikhilam Navatascaramam Dasatah" Sutra, (a) Numbers are taken below base, (b) Numbers are taken above base.

As shown in Figure 1, we write the multiplier and the multiplicand in two rows followed by the differences/addition of each of them from the chosen base. 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/ incremented 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 (2×3 = 6 or 2×4=8). The left hand side (LHS) of the product can be found by cross subtraction or addition the second number of Column 2 from the first number of Column 1 or vice versa, i.e., 998-003 = 995 or 997-002 = 995 and 1002+004=1006 or 1004+002=1006. The final result is obtained by concatenating RHS and LHS (Answer = 995006 or 1006008).

The serious drawback of Nikhilam sutra can be summarized as:

(i) Both the multiplier and multiplicand are less or greater than the base.

(ii) Multiplier and multiplicand are nearer to the base.

2.2 "Urdhva-tiryakbyham" Sutra

The meaning of this sutra is "Vertically and crosswise" and it is applicable to all the multiplication operations. Figure 2 represents the general multiplication procedure of the 4×4 decimal digit multiplication. To illustrate this multiplication scheme, let us consider the multiplication of two decimal numbers (1234 × 8765). Multiplication using 'Urdhva

tiryakbhyam' Sutra is shown in Figure 2. The numbers to be multiplied are written on two consecutive sides of the square as shown in the figure. Each of the small squares is partitioned into two equal 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 yellow boxes are added and producing sum and carry digits. Finally results are obtained by the addition of sum digits and the previous carry digits. Carry for the first step is taken to be zero.

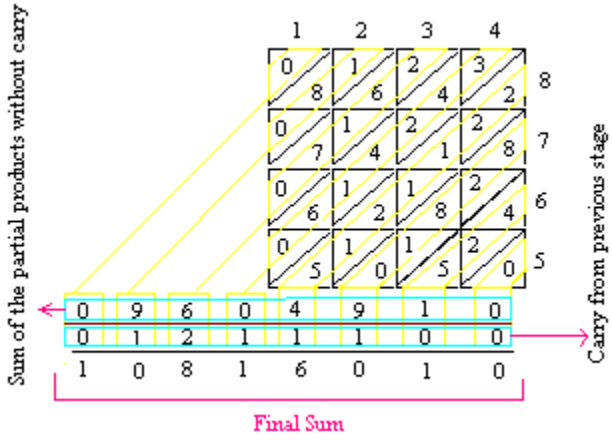


Figure 2. Multiplication implementation using "Urdhva-tiryakbyham" Sutra

2.2.1 Mathematical Background of "Urdhva-tiryakbyham" Sutra

Assume that X and Y are two numbers, to be multiplied. Mathematically X and Y can be represented as:

$$A = \sum_{i=0}^{N-1} A_i \, 10^i \quad (11)$$

$$B = \sum_{j=0}^{N-1} B_j \, 10^j \quad (12)$$

Assume that, their product is equal to Z. Then Z can be represented as:

$$Z = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} A_i \, B_j \, 10^{i+j} \quad (13)$$

Where (A_i , B_j € (0,1,2,.....,9) and 'N' may ne any number.

From the above expression, equation no (13), it can observed that each digit is multiplied consecutively and shifted towards the proper positions for partial product generation. Finally the partial products are added with the previous carry to produce the final results.

3. HARDWARE IMPLEMENTATION OF VEDIC MULTIPLIER

3.1 Extended "Nikhilam Navatascaramam Dasatah" Sutra for Binary Number System

Consider two n bit numbers X and Y. k is the exponents, z_1 and z_2 of both X and Y respectively. X and Y can be represented as:

$$X = 2^k \pm z_1 \quad (14)$$

$$Y = 2^k \pm z_2 \quad (15)$$

Assuming product of the number is equals to P.

$$P = X \times Y = (2^k \pm z_1)(2^k \pm z_2) \quad (16)$$

For the fast multiplication using extended rule of the sutra the bases of the multiplicand and the multiplier assuming same, thus the equation no 16 can be rewritten as

$$P = XY = 2^k(X \pm z_2) \pm z_1 z_2 \quad (17)$$

From equation no 17 it is observed that a large number multiplication can easily decomposed into a small number multiplication, addition/subtraction and shifting, leading towards the reduction of hardware cost, propagation delay and power consumption. Small number of the multiplication can easily implemented using *"Urdhva-tiryakbyham"* sutra (formula).

3.1.1 Hardware Implementation of "Nikhilam Navatascaramam Dasatah" Sutra

The mathematical expression for the proposed algorithm is shown in equation no 17. Hardware implementation of this mathematics is shown in figure 3. It consists of five major segments such as:- (i) RSU, (ii) Sub-tractor, (iii) Adder/Sub-tractor, (iv) Multiplier and (v) Shifter. Since the radix is same for both the inputs so the radix generated from RSU corresponding to input X is fed to the sub-tractor. The other input to the sub-tractor is Y. The output generated is the residue corresponding to Y. Both the residues are multiplied through the multiplier. The residue corresponding to Y is added with or subtracted from X by the first adder/sub-tractor block. The result is shifted left based on the exponent generated from RSU. The multiplier result is added with or subtracted from the shifted result by the second adder/sub-tractor.

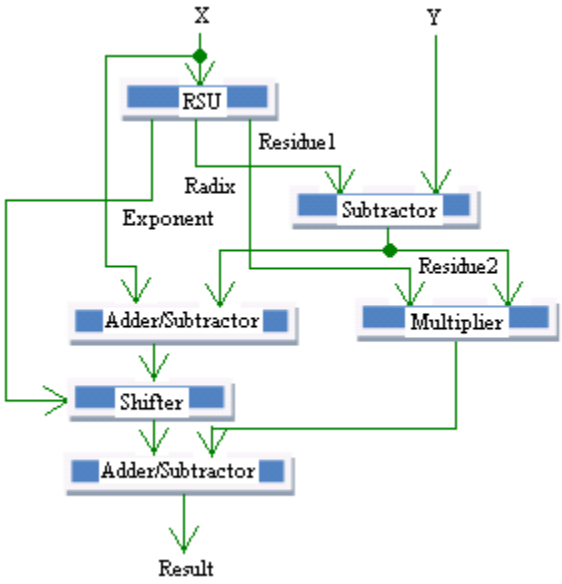


Figure 3. Hardware Implementation of "Nikhilam Navatascaramam Dasatah" Sutra

3.2. Mathematical Background of *"Urdhva-tiryakbyham" sutra for binary number system:*

Assume the product of two N-bit words are described as

$$X = \sum_{i=0}^{N-1} x_i 2^i \quad (18)$$

$$Y = \sum_{j=0}^{N-1} y_j 2^j \quad (19)$$

where $x_i , y_j \varepsilon \{ 0, 1\}$

Multiplication can be described as

$$Z = XY = \sum_{i=0}^{N-1} x_i 2^i \sum_{j=0}^{N-1} y_j 2^j = \sum_i \sum_j x_i y_j 2^{i+j} \quad (20)$$

Let k= i+j

$$Z= \sum_{k=0}^{2N-1} \sum_{i=0}^{N-1} x_i y_{k-i} 2^k \quad (21)$$

$$Z= \sum_{k=0}^{2N-1} z_k 2^k \quad (22)$$

Where $Z_k = \sum_{i=0}^{N-1} x_i y_{k-i}$ (23)

The equation (21) shows that the co-efficient of multiplication that can be achieved by the convolution sum of the two finite number sequences.

3.3. Circuit Modules

3.3.1 Radix Selection Unit (RSU):

3.3.1.1 Mathematical Representation of RSU:

Consider an 'n' bit binary number X, and it can be represented as

$X = \sum_{i=0}^{n-1} x_i 2^i$ Where $x_i \epsilon \{0,1\}$. Then the values of X must lie in the range $2^{n-1} \leq X < 2^n$. Consider the mean of the range is equals to A.

$$A = \frac{2^{n-1}+2^n}{2} \quad (24)$$

$$= 2^{n-2} \times 3$$

If X >A Then radix = 2^n

If $X \leq A$ Then radix = 2^{n-1}

3.3.1.2 Hardware Implementation of RSU:

The architecture of RSU is shown in Figure 4. RSU consists of seven major segments such as:- (i) Exponent Determinant (ED), (ii) Adder, (iii) Mean Determinant (MD), (iv) Multiplexer (MUX), (v) Comparator, (vi) Shifter and (vii) Sub-tractor. E.D. determines the maximum power of the base (here 2) corresponding to the input X. The maximum power of '2' is called the exponent. The exponent is incremented by '1' through the adder block. The exponent is fed to the mean determinant block which computes the mean of the two radices between which the input lies. The output of MD and the input are fed to the comparator. The output of the comparator is used as select input to the multiplexer. The incremented exponent and the previous exponent are fed to the multiplexer which selects the proper exponent. The output of the multiplexer is fed to the shifter which is initialized by '1'. The MUX output generates the radix from which the input is subtracted to generate the residue.

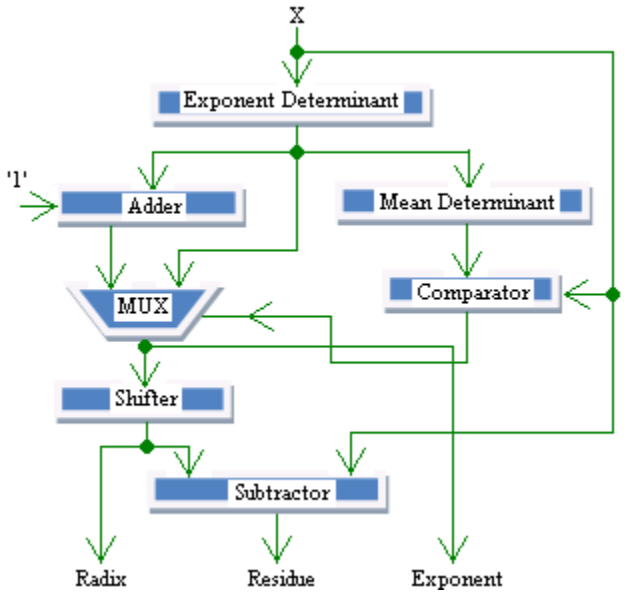


Figure 4. Hardware Implementation of RSU

3.3.2. Exponent Determinant (ED):

The hardware implementation of the exponent determinant is shown in Figure 5.The integer part or exponent of the number from the binary fixed point number can be obtained by the maximum power of the radix. For the non-zero input, shifting operation is executed using parallel in parallel out (PIPO) shift registers. The number of select lines (in Figure 5 it is denoted as S_1, S_0) of the PIPO shifter is chosen as per the binary representation of the number $(N-1)_{10}$. 'Shift' pin is assigned in PIPO shifter to check whether the number is to be shifted or not (to initialize the operation 'Shift' pin is initialized to low). A decrementer has been integrated in this architecture to follow the maximum power of the radix. A sequential searching procedure has been implemented here to search the first '1' starting from the MSB side by using shifting technique. For an N bit number, the value $(N-1)_{10}$ is fed to the input of decrementer. The decrementer is decremented based on a control signal which is generated by the searched result. If the searched bit is '0' then the control signal becomes low then decrementer start decrementing the input value (Here the decrementer is operating in active low logic). The searched bit is used as a controller of the decrementer. When the searched bit is '1' then the control signal becomes high and the decrementer stops further decrementing and shifter also stops shifting operation. The output of the decrementer shows the integer part (exponent) of the number.

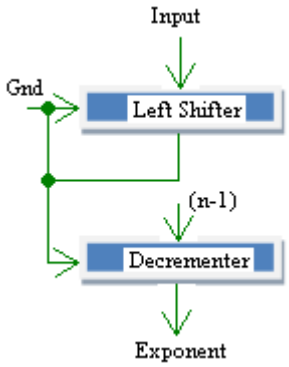


Figure 5. Hardware Implementation of ED

3.3.2.1 Mean Determinent (MD):

The architecture of the Mean Determinant is exhibited in Figure 6. Mean Determinant takes the exponent (power of MSB) as input and is decremented by one. Decrement operation is performed in the sub-tractor. The decremented value is fed to the shifter which acts as select input for the shifter. The Binary value "11" ('3_{10}') is shifted to the left depending upon the select input and the result from the shifter is the desired mean value.

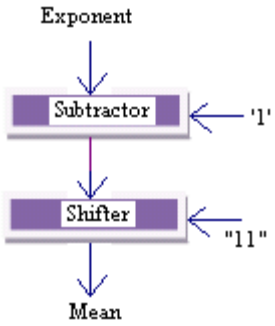


Figure 6. Architecture of Mean Determinent.

3.3.2.2 Comparator:

The architecture of comparator is shown in Figure 7. The comparison task is performed based on subtraction operation. The subtraction result has two parts:- (i) result and (ii) Borrow Out. The result is not the matter of interest for this architecture so it has not been exhibited in the figure. From the "Borrow Out" signal, the comparison decision has been taken.

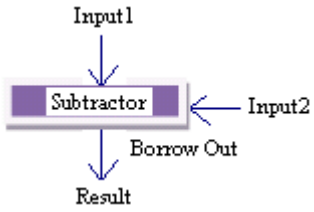


Figure 7. Architecture of Comparator

3.4 Multiplier Implementation using “Urdhva-tiryakbyham” Sutra

Considering the long-hand sequences of equation no 20, 4 bit multiplier algorithm is shown in Figure 8. Higher bit multipliers can be realized in similar manner. Partial products are added in two stages. Adders and 4 to 3 compressors are used to minimize the stage operations. Compressors and adders are used carefully so that a minimum number of outputs would be generated. Thus by using minimum number of adders/compressors partial products are added without compromising the number of bits generation for the next stage operation. The recursive method of multiplication involves building wider vector elements out of several of the narrower vector elements and then multiple results are added together.

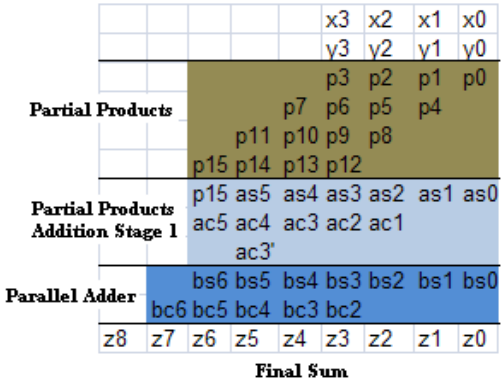


Figure 8. 4×4 Bit Multiplier implementation using “Urdhva-tiryakbyham” Sutra

4. RESULTS AND DISCUSSIONS

The entire algorithm in this paper was simulated and their functionality was examined by Spice Spectre Simulator. Performance parameters such as propagation delay and power consumptions analysis of this paper was calculated using standard 90nm CMOS technology. As shown, the application of the Vedic multiplication methodology cuts the amount of the hardware as a result

the propagation delay, dynamic switching power consumptions, and dynamic leakage power consumptions were reduced.

Critical path delay (latency) of a VM can be defined as the total time taken by the circuit for the computation of the entire multiplication operation, thus latency can be computed in terms of propagation delay of the circuit. The critical path delay of an n bit Vedic multiplier (VM) consists four sections i.e., (a) Radix Selection Unit (RSU), (b) Addition/Subtraction Unit, (c) Multiplier Unit and (d) Shifting module. So the total propagation delay for the complex operation can be expressed as:

$$t_{pd} = t_{RSU} + t_{adsb} + t_m + t_{shft} \quad (25)$$

Where t_{pd} =Total propagation delay,

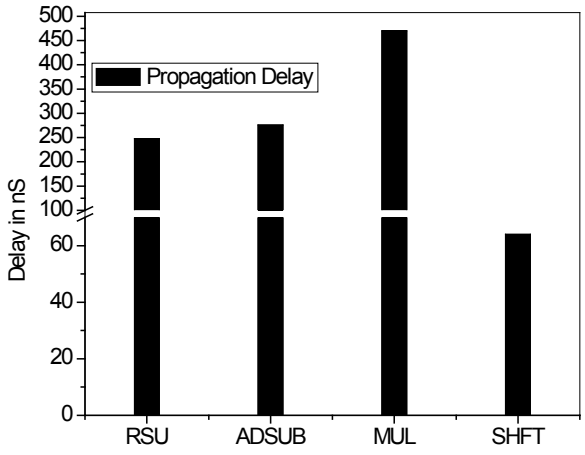
t_{RSU} =Propagation delay for radix selection unit,

t_{adsb} =Propagation delay for addition/subtraction unit,

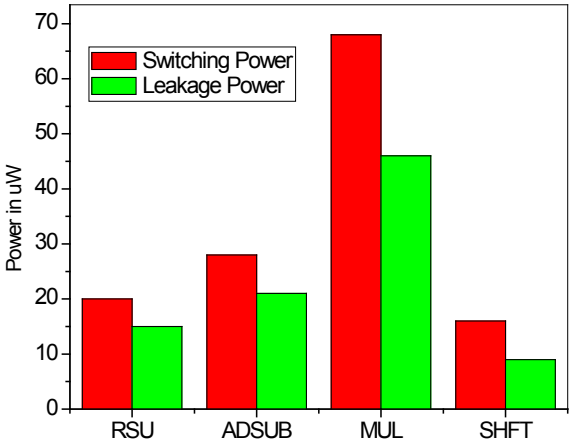
t_m =Propagation delay for multiplication unit,

t_{shft} = Propagation delay for shifting module.

The performance parameters such as propagation delay, dynamic switching power consumption and dynamic leakage power consumption to implement Vedic multiplier is shown in figure 9. Input data is taken as a regular fashion for experimental purpose. We have kept our main concentration for reducing the propagation delay, dynamic switching power and dynamic leakage power consumption and energy delay product.



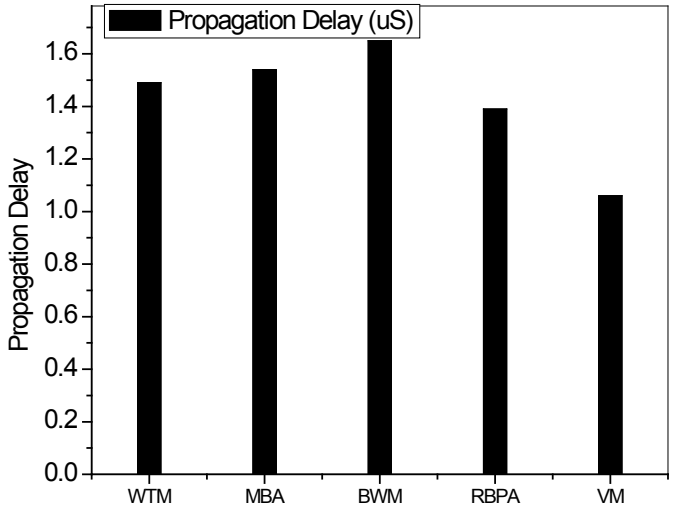
(a)



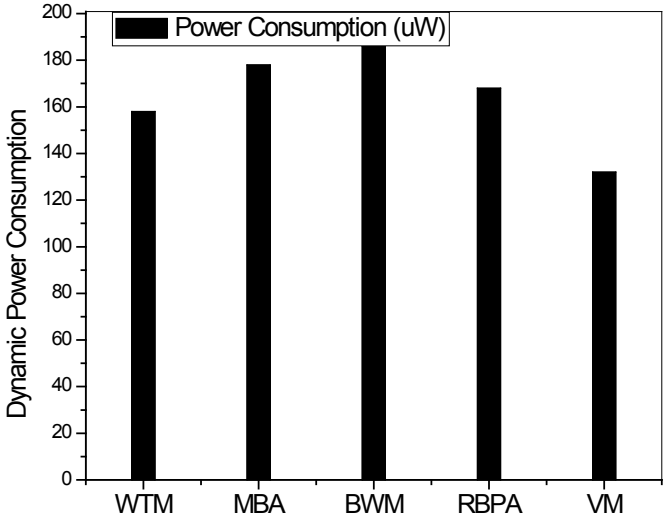
(b)

Figure 9. (a) Propagation delay (ns) analysis of different circuit modules for 32×32 bit VM implementation, (b) Dynamic Switching and Dynamic Leakage power (uW) analysis of different circuit for 32×32 bit VM implementation

A comparison between different architecture in terms of propagation delay and dynamic switching power consumption is tabulated in Figure 10. Simulation results offered 29%, 31%, 35%, and 23% improvement in terms of propagation delay compared with WTM, MBA, BWM and MRA based architecture respectively. Whereas the corresponding improvement in power is 17%, 26%, 29%, and 21% respectively compared with WTM, MBA, BWM and MRA based architecture.



(a)



(b)

Figure 10: (a) Comparison results in terms propagation delay (ns) of 32×32 bit multiplier of different architectures (b) Comparison results in terms of Dynamic Switching power (uW) of 32×32 bit multiplier.

5. CONCLUSIONS

In this paper we report on a 32×32 bit high speed low power multiplier design based on the formulas of the ancient Indian Vedic Mathematics which are having wide application in DSP processors. The implementation was done in Spice spectre and compared with the mostly used architecture like WTM, MBA, BWM and LRA based implementation. This novel architecture combines the advantages of the Vedic mathematics for multiplication which encounters the stages and partial product reduction. The architecture offered 29%, 31%, 35%, and 23% improvement in terms of propagation delay compared with WTM, MBA, BWM and RBPA based implementation respectively. Whereas the corresponding improvement in power is 17%, 26%, 29%, and 21% respectively compared with WTM, MBA, BWM and RBPA based architecture.

REFERENCES

[1] A. Asati and Chandrashekhar, "An Improved High Speed fully pipelined 500 MHz 8×8 Baugh Wooley Multiplier design using 0.6 μm CMOS TSPC Logic Design Style", Proc. IEEE, ICIINFS 2008, pp. 1-6, Dec. 8-10, 2008.

[2] A. Chandrakasan and R. Brodersen, "Low-power CMOS digital design" , IEEE Journals on Solid-State Circuits, Vol. 27, No. 4, pp. 473–484, Apr. 1992.

[3] N.-Y. Shen and O. T.-C. Chen, "Low-power multipliers by minimizing switching activities of partial products", Proc. IEEE, ISCAS 2002, vol. 4, pp. 93–96, May 2002.

[4] O. T. Chen, S. Wang, and Y.-W. Wu, "Minimization of switching activities of partial products for designing low-power multipliers," IEEE Transc. on Very Large Scale Integration (VLSI) Syst., Vol. 11, No. 3, pp. 418–433, June 2003.

[4] B. Parhami, Computer Arithmetic Algorithms and Hardware Designs, 1st ed. Oxford, U.K.: Oxford Univ. Press, 2000.

[5] C. S Wallace, "A Suggestion for a Fast Multiplier," IEEE Trans. on Computers, Vol. EC13, pp. 14-17, December 1964.

[6] J. Hu, L. Wang, and T. Xu, "A Low-Power Adiabatic Multiplier Based on Modified Booth Algorithm", Proc. IEEE, ISIC'07, pp. 489-492, Sept. 26-28, 2007.

[7] C. R. Baugh, and B.A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm", IEEE transc. on Computers, Vol. C-22, No. 12, pp. 1045-1047, December 1973

[8] K-C. Kuo, and C-W. Chou, "Low power and high speed multiplier design with row bypassing and parallel architecture, Journal of Microelectronics, 2010, doi:10.1016/j.mejo.2010.06.009

[9] K. Z. Pekmestzi, "Multiplexer-Based Array Multipliers", IEEE transc. on Computers, Vol 48, No. 1, pp. 15-23, January 1999.

[10] [11] P. K. Saha, A. Banerjee, and A. Dandapat, "High Speed Low Power Complex Multiplier Design Using Parallel Adders and Subtractors", International Journal on Electronic and Electrical Engineering, (IJEEE), vol 07, no. 11, pp 38-46, December 2009.

[11] Z. Huang, and M. D. Ercegovac, "High-Performance Low-Power Left-to-Right Array Multiplier Design," IEEE Transactions on Computers, vol 54, no. 3, pp 272-283, March 2005.

[12] P.-M. Seidel, L.D. McFearin and D.W. Matula, "Secondary radix recodings for higher radix multipliers", IEEE transc. on Computers, Vol 54, No. 2, pp. 111- 123, February 2005.

[13] P. Mehta, and D. Gawali, "Conventional versus Vedic mathematical method for Hardware implementation of a multiplier," Proc. IEEE ACT-2009, pp. 640-642, Dec. 28-29, 2009.

[14] H. D. Tiwari, G. Gankhuyag, C. M. Kim, and Y. B. Cho, "Multiplier design based on ancient Indian Vedic Mathematics," Proc. IEEE International SoC Design Conference, pp. 65-68, Nov. 24-25, 2008.

[15] P. Saha, A. Banerjee, P. Bhattacharyya, and A. Dandapat, "High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics", Proc. (Abstract) IEEE TechSym 2011, pp. 38-38,Jan 14-16.

[16] P. K. Saha, A. Banerjee, and A. Dandapat, "High Speed Low Power Factorial Design in 22nm Technology," Proc. AIP2009, pp. 294-301, 2009.