International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:12, No:3, 2018

Analytical Comparison of Conventional Algorithms with Vedic Algorithm for Digital Multiplier

Akhilesh G. Naik, Dipankar Pal

***Abstract*—**In today's scenario, the complexity of digital signal processing (DSP) applications and various microcontroller architectures have been increasing to such an extent that the traditional approaches to multiplier design in most processors are becoming outdated for being comparatively slow. Modern processing applications require suitable pipelined approaches, and therefore, algorithms that are friendlier with pipelined architectures. Traditional algorithms like Wallace Tree, Radix-4 Booth, Radix-8 Booth, Dadda architectures have been proven to be comparatively slow for pipelined architectures. These architectures, therefore, need to be optimized or combined with other architectures amongst them to enhance its performances and to be made suitable for pipelined hardware/architectures. Recently, Vedic algorithm mathematically has proven to be efficient by appearing to be less complex and with fewer steps for its output establishment and have assumed renewed importance. This paper describes and shows how the Vedic algorithm can be better suited for pipelined architectures and also can be combined with traditional architectures and algorithms for enhancing its ability even further. In this paper, we also established that for complex applications on DSP and other microcontroller architectures, using Vedic approach for multiplication proves to be the best available and efficient option.

***Keywords*—**Wallace tree, Radix-4 Booth, Radix-8 Booth, Dadda, Vedic, Single-Stage Karatsuba, Looped Karatsuba.

I. Introduction

MULTIPLIER is an important and integral block of every microcontroller and DSP processor. Optimizing this block analytically has been significant and critical so as to support today's complex scenario of DSP applications. This makes it essential for circuit designers to rise above the stereotype and conceive new multiplication techniques and algorithms with/without the support of conventional algorithms for efficient usage [1]-[4]. This gives the motivation to study and compare different traditional architectures and use its ability to combine with a novel Vedic architecture (not used for designing multipliers before although known to exit for centuries) so that it can be best used to design pipelined techniques along with the conventional algorithms in tandem to enhance ability of the existing algorithms and architectures. Traditional algorithms like Wallace Tree, Radix-4 Booth, Radix-8 Booth, and Dadda can be compared to study its ability and advantageous aspects so that their ability can be combined with the Vedic algorithms to suit and enhance the present endeavour of pipelined architectures. The fact that modern DSP applications with complex analytics require complex calculations and hence prefer microcontroller architectures with pipelined stages to be exploited for throughput and speed, gives a boost for exploring the internal techniques of Vedic algorithms [11], [12], which have genetics of functioning with parallel mathematical operation and reduced computations. On one side, it has been proved that the Wallace-Tree simplifies and thereby speeds up the partial product compression with its tree like structure. On the other hand, it is also true that the Booth (Radix-4, Radix-8) algorithm [3], [9], depending on their encoding techniques reduce the number of partial products based on the Radix phenomena that they employ. Booth combined with Wallace Tree [2], [10], [13], [14] can speed up the overall throughput scenario by utilizing the ability to reduce the partial products on one end and to simplify the compression of partial products utilizing the parallel compression based on Wallace Tree algorithm [5] and its optimization architectures. Similar to Wallace-Tree, the Dadda algorithm also has a tradition of compressing the partial products, but in a limited manner, as each step-height is based on a certain factor of multiplication to its successor. Hence, amongst these, Booth (Radix-4, Radix-8) algorithm [1], [14] is the only algorithm which utilizes the method of reducing the number of partial product terms, whereas the Wallace Tree [5], [8] and Dadda algorithms [6], [7] completely rely on the advantage of compressing the partial product terms in a parallel manner reducing the number of steps with a particular factor. Section II of the paper explores the existing multiplier algorithms with examples and particularly explains the characteristics of the Wallace-Tree, Dadda and Booth (Radix-4, Radix-8) algorithms followed by Section III, which provides the explanation of the Vedic algorithms and its technique of optimizations to enhance its ability to deal with various current pipelined scenarios and also will provide a glimpse of how to combine the traditional algorithms with Vedic methods to increase the overall ability in tandem even further. Section IV provides a new idea proposed by these authors of further optimization of the Vedic algorithm in combination with the Wallace Tree algorithm/architecture. Section V concludes the paper describing the novelty added to the Vedic algorithm by the proposed architectural optimization.

Akhilesh G. Naik is with Birla Institute of Technology and Science, Pilani, Goa Campus, India (e-mail: p20160015@goa.bits-pilani.ac.in).

Dipankar Pal is with Birla Institute of Technology and Science, Pilani, Goa Campus, India (corresponding author; e-mail: dipankarp@goa.bits-pilani.ac.in).

II. Characteristics of Traditional Algorithms

The traditional algorithms under investigation in this paper are Wallace Tree, Booth (Radix-4, Radix-8), and Dadda. The reason behind investigating these particular algorithms is the

fact that they can be used in combination with the Vedic algorithms to enhance or further optimize the overall ability. The novelty is thus obtained in the overall algorithm and overall architectural combination.

A. Wallace Tree

Wallace Tree [2], [5] is based on the concept of optimizing the architectures with compressors employed in parallel for speeding up the compression of the partial products and hence reducing the number of steps required to establish the output.

Fig. 1 shows the Wallace Tree architecture in which the partial products are arranged in a tree-like triangular structure which reduces the complexity to log N as compared to $(\text{Log } N)^2$, if the partial products had been added normally, where N is the number of bits. This type of structural mapping can be combined with other algorithms, which work on reducing the number of partial terms.

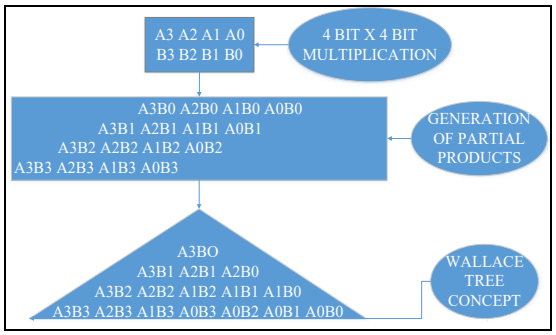

Fig. 1 Wallace Tree Structural Mapping

TABLE I
RADIX-4 ENCODING

Multiplier $Y_{2i-1}\ Y_{2i}\ Y_{2i+1}$	Encoded Operation on Multiplicand, X
0 0 0	0X
0 0 1	+X
0 1 0	+X
0 1 1	+2X
1 0 0	-2X
1 0 1	-X
1 1 0	-X
1 1 1	0X

B. Dadda Algorithm

Similar, to Wallace Tree, Dadda algorithm [6], [7] also utilizes the concept of reducing the steps for compressing of partial products, but in a limited manner. In this, each step-size is dependent on a factor of multiplication of its successor. Hence, the Dadda algorithm will have each step stage dependent on a factor of its successor for its height reduction. The algorithm can also be termed as the reduced form of Wallace Tree, as Wallace Tree utilizes the total possible reduction as compared to the height dependent one as in Dadda. Fig. 2 depicts the structural view of Dadda algorithm wherein each step height (d_j) is 1.5 times of its successor. This algorithm provides its best feature in reducing number of steps in partial-product-compression when the height of successive partial products reduces by the maximum value (typically 2/3=0.67).

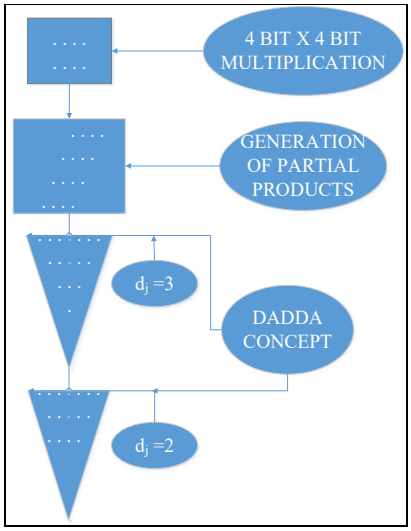

Fig. 2 Dadda Concept

C. Booth Algorithm (Radix-4, Radix-8)

These algorithms [3] basically deal with encoding of the bits based on predefined encoding operation techniques. The algorithm provides the advantage in terms of reducing the number of partial product terms based on the Radix technique that it uses. The Radix-4 technique [4], [10] reduces the partial product terms to $N^2/2$, whereas Radix-8 [1] technique reduces the partial product terms to $N^2/3$, where N is the number of bits. In this algorithm the bits-encoding functions based on the equation given as:

$$N = 1+\log_2 R \quad (1)$$

A brief outline of the algorithm is given here. As per (1), in the Radix-4 encoding technique, the number of bits encoded at a time is 3, whereas in Radix-8 technique, the number of bits encoded at a time is 4. The encoding operation for its corresponding bits pattern is as depicted in Tables I and II for Booth Radix-4 and Radix-8, respectively [1], [4], [10].

TABLE II
RADIX-8 ENCODING

Multiplier $Y_{i+2}\ Y_{2i-1}\ Y_{2i}\ Y_{2i+1}$	Encoded Operation on multiplicand, X
0 0 0 0	0X
0 0 0 1	+X
0 0 1 0	+X
0 0 1 1	+2X
0 1 0 0	+2X
0 1 0 1	+3X
0 1 1 0	+3X
0 1 1 1	+4X
1 0 0 0	-4X
1 0 0 1	-3X
1 0 1 0	-3X
1 0 1 1	-2X
1 1 0 0	-2X
1 1 0 1	-X
1 1 1 0	-X
1 1 1 1	0X

Since this algorithm gives an advantage on reducing the number of partial products, the steps required to generate the partial products are lesser as compared to the Wallace Tree and Dadda algorithms. To make the best use in an optimized manner, the advantage of reduction in the number of partial products can be used along with the advantage of reduction in the number of steps required to compress the partial products. In other words, Booth (Radix-4, Radix-8) [4], [1], can be combined with the Wallace Tree [5] and Dadda algorithms [6], [7] to gain the overall reduction in complexity and to achieve least delay in obtaining the multiplication output. The same concept can be applied to Vedic (Karatsuba) algorithm [12] too, as we shall see next.

The Karatsuba algorithm [11], [12] itself can be looped to achieve its best delay for gaining recursive enhancement in the speed of the multiplier. This concept, which is the contribution of these authors, is explained in the following section.

III. Single-Stage Karatsuba and Looped Karatsuba

Normal Karatsuba algorithm for decimal numbers is based on the technique where that the digits are divided into half, and then performing the trick of multiplication by step reduction using certain mathematical analysis. Table III depicts the N digits X and Y being split each into 2× N/2 digits having denoted by A, B and C, D, respectively. Combining them as shown in the Table III will reversely lead to X and Y having N digits. Multiplying X and Y will lead to (2) resulting in four product terms (AC, BD, AD, BC) as usual.

TABLE III
Bit Division in Recursive Karatsuba

N N/2 N/2	Combining X_1 and Y_1 to form X and Y
X A B	X= (A*($10^{(N/2)}$)) + B
Y C D	Y= (C*($10^{(N/2)}$)) + D

$$X*Y=(AC*10^{(N)})+BD+((AD + BC)*10^{(N/2)}) \quad (2)$$

But Karatsuba utilizes a trick, wherein these four product terms can be reduced to three product terms. Equations (3) and (4) gives the trick of Karatsuba as follows:

$$(A+B) * (C+D) = AC+BD+(AD+BC) \quad (3)$$

$$(AD+BC)=((A+B)*(C+D)) –(AC+BD) \quad (4)$$

Having the repetitive terms in (4) and just by analyzing the term (AD+BC) with the terms that are already calculated or available leads to only three product terms AC, BD being already calculated, and the third product term ((A+B)*(C+D)) which is required to be calculated totally leading to only three unique product terms in the multiplication process. The same concept can be applied in bit/binary form to employ Karatsuba digitally having the power of '2' instead of power of '10', which in turn means only shifting the number, bit-wise or digit-wise to the left, respectively.

Mathematically, Looped Karatsuba is depicted in Fig. 3.

Recurrence equation [11] for the number of steps is:

$$O(N) = 3*(O(N/2)) \quad (5)$$

where O(N) gives the number of steps, N is the number of bits. O(N/2) is multiplied by the number of steps recursively.

BIT MULTIPLICATION USING LOOPED KARATSUBA

1001
x 1000
0100 1000

$(10x10)x2^4+(01x00)+[[(10+01)x(10+00)]-[(10x10)+(01x00)]]x2^2$
$=(10x10)x2^4+(01x00)+[[(11)x(10)]-[(10x10)+(01x00)]]x2^2$
$=((0100)x2^4)+000000+((0110-0100)x2^2)$
$=01001000$

(10x10) – Karatsuba Algorithm
$(1x1)x2^2+(0x0)+[[(1+0)x(1+0)]-[(1x1)+(0x0)]]x2^1$
$=(1x1)x2^2+(0x0)+[[(1+0)x(1+0)]-[(1x1)+(0x0)]]x2^1$
$=100+0+0000$
$=0100$

(01x00) – Karatsuba Algorithm
$(0x0)x2^2+(1x0)+[[(0+1)x(0+0)]-[(0x0)+(1x0)]]x2^1$
$=(0x0)x2^2+(1x0)+[[(0+1)x(0+0)]-[(0x0)+(1x0)]]x2^1$
$=000+0+00$
$=000$

(11x10) – Karatsuba Algorithm
$(1x1)x2^2+(1x0)+[[(1+1)x(1+0)]-[(1x1)+(1x0)]]x2^1$
$=(1x1)x2^2+(1x0)+[[(10)x(01)]-[(1)+(0)]]x2^1$
$=100+0+(10)$
$=0110$

Fig. 3 Looped Karatsuba (LK) demonstrated mathematically

This enhances the complexity to $N^{log_2(3)}$, which is approximately equal to $N^{1.585}$. Table IV depicts the comparison of different existing conventional algorithms (Wallace Tree, Dadda, Booth (Radix-4, Radix-8)) with the

Vedic algorithms (Karatsuba) like Single-Stage Karatsuba (SSK) and Looped Karatsuba (LK) optimization on the complexity in terms of partial product generation.

TABLE IV
ALGORITHM COMPLEXITY FOR PARTIAL PRODUCTS GENERATION

Algorithms	Complexity for Partial Products Generation
Wallace Tree, Dadda	N^2
Radix-4 Booth	$N^2/2$
Radix-8 Booth	$N^2/3$
Single-Stage Karatsuba	$(N^{1.58})^2$
Looped Karatsuba (LK)	$(N^{1.58})^{1.58}$

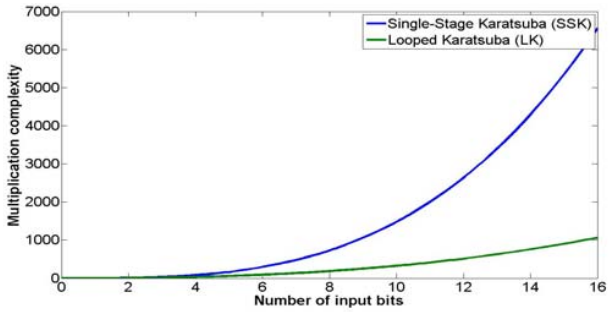

Fig. 4 Graph depicting the least complexity achieved in Looped Karatsuba (LK) as compared to Single-Stage Karatsuba (SSK)

Single-Stage Karatsuba gives the complexity of $(N^{1.58})^2$ as compared to Looped Karatsuba (LK), which will further reduce the complexity for same bit-length to only $(N^{1.58})^{1.58}$, reporting thereby an improvement.

Looped Karatsuba (LK) plays an important role for increased number of bits as compared to Single-Stage Karatsuba (SSK), which is evident from the graph shown in the Fig. 4.

Single-Stage Karatsuba (SSK) concept is based on application of the Karatsuba algorithm only at the first stage following normal multiplication using gates/MUX. Thus in SSK, the digits are split into two halves using Karatsuba at the first stage and then the split-digits are multiplied directly. In Looped Karatsuba (LK) however, the split numbers (from the first stage) are again subjected to the same algorithm. This further continues repetitively or recursively until the numbers can be directly multiplied. It can be pictorially shown (avoided here for brevity).

IV. LOOPED KARATSUBA COMBINED WITH WALLACE-TREE

Since it is evident from Section III that Looped Karatsuba gives least complexity for higher number of bits, it can be combined further with Wallace Tree structure [2], [10] (described in Section II) for compressing of the three multiplication product terms which can optimize the algorithm even further making it an ideal candidate for pipelined architectures.

Without using Wallace Tree structure, this algorithm would compress employing full adders and half adders normally, giving the complexity of $(\text{Log } N)^2$. But employing or combining with Wallace Tree [2], [10], it can optimized further to achieve a complexity reduced to log N as depicted in Fig. 5, which can reduce the delay greatly thereby increasing the speed in a given pipelined architecture.

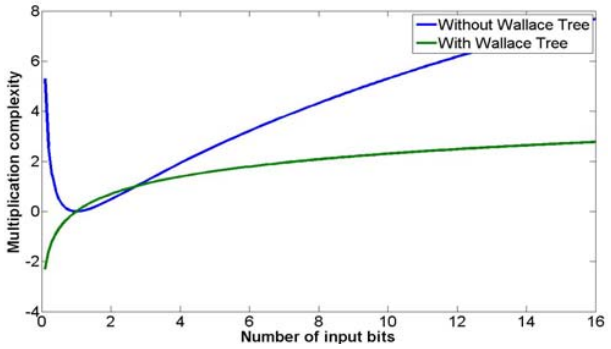

Fig. 5 Graph depicting the least complexity achieved when Wallace Tree is being used

From Figs. 4 and 5, it is evident that with Wallace Tree utilized for the compression of the product terms generated from Looped Karatsuba (LK), the Looped Karatsuba (LK) algorithm is improved and optimized further for higher number of bits and can be the best candidate available to date for pipelined architectures by enhancing the speed optimizing the throughput.

V. CONCLUSION

This paper compares and describes the traditional algorithms with Vedic algorithm (Single-Stage Karatsuba). The authors have further proposed the Looped Karatsuba by extending the known Karatsuba algorithm recursively to establish how the modification of the age-old Vedic algorithm optimized the delay further. In this paper, the other novelty included involves using a Wallace-Tree Structure in tandem with Looped Karatsuba (LK) concept which provides further enhancement in speed and can best be best suited among available techniques for pipelined architectures and multi-core processor architectures. This novel architecture (Vedic (Looped Karatsuba) in combination with Wallace-Tree Structure) can enhance the speed of modern DSP applications involving high-end computational complexities, which can boost present day electronic and communication technologies and system architectures.

REFERENCES

[1] H. Jiang, J. Han, F. Qiao and F. Lombardi, "Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation", *IEEE Trans. Computers*, vol. 65, no. 8, pp. 2638-2644, Aug. 2016.

[2] R. S. Waters and E. E. Swartzlander, "A Reduced Complexity Wallace Multiplier Reduction," *IEEE Trans. Computers*, vol. 59, no. 9, pp. 1134-1137, Aug. 2010.

[3] K. Tsoumanis, S. Xydis, C. Efstathiou, N. Moschopoulos and K. Pekmestzi, "An Optimized Modified Booth Recoder for Efficient Design of the Add-Multiply Operator", *IEEE Trans. Circuits and Systems* I, vol. 61, no. 4, pp.1133-1143, 2014.

[4] W. Liu, L. Qian, C. Wang. H. Jiang, J. Han, F. Lombardi, "Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant

Computing", *IEEE Trans. Computers*, vol. 66, no. 8, pp.1435-1441, 2017.

[5] N. Sureka, R. Porselvi and K. Kumuthapriya, "An Efficient High Speed Wallace Tree Multiplier", *IEEE International Conference on Information Communication and Embedded Systems (ICICES)*, pp. 1023-1026, 2013.

[6] B. Jeevan, S. Narendar, Dr. C.V. Krishna Reddy, Dr. K. Sivani, "A High Speed Binary Floating Point Multiplier Using Dadda Algorithm", 2013 *International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing(iMac4s)*, pp. 455-460, 2013.

[7] Vinod Budhe, Prasanna Palsodkar, Prachi Palsodakar, "Design and Verification of Dadda Algorithm Based Binary Floating Point Multiplier", 2014 *International Conference on Communication and Signal Processing*, pp. 1073-1077, 2014.

[8] K. B. Jaiswal, N. Kumar V, and P. Seshadri, Lakshminarayanan G., "Low Power Wallace Tree Multiplier Using Modified Full Adder", *IEEE International Conference on Signal Processing, Communication and Networking (ICSCN),* pp. 1-4, 2015.

[9] Manjunath, V. Harikaran, K. Manikanta, Sivananthan S. and Sivasankaran K., "Design and Implementation of 16x16 Modified Booth Multiplier", *IEEE Online International Conference on Green Engineering and Technologies (IC-GET),* pp. 1-5, 2015.

[10] S. Asif and Y. Kong, "Performance Analysis of Wallace and Radix-4 Booth-Wallace Multipliers", *IEEE Electronic System Level Synthesis Conference (ESLsyn),* pp. 17-22, 2015.

[11] Srini Devadas, "Introduction to Algorithms," Lecture 11: Integer Arithmetic, Karatsuba Multiplication, MIT Open Course Ware, Massachusetts Institute of Technology, 6.006, Fall 2011. Available at: https://www.youtube.com/watch?v=eCaXlAaN2uE.

[12] A. Mehta, C. B. Bidhul, S. Joseph and Jayakrishnan P., "Implementation of Single Precision Floating Point Multiplier using Karatsuba Algorithm", *International Conference on Green Computing, Communication and Conservation of Energy (ICGCE),* pp. 254-256, 2013.

[13] K. Shruthilaya and M. Vinoth, "Power Estimation of Modified Booth Recoder for Efficient Add-Multiply Operator", *IEEE Conference on Computing for Sustainable Global Development (INDIACom),* pp. 1684-1689, 2015.

[14] R. Pratibha, P. Sandhya, and R. Varun, "Design of High Performance and Low Power Multiplier using Modified Booth Encoder", *IEEE International Conference on Electrical, Electronics and Optimization Techniques (ICEEOT),* pp. 794-798, 2016.