A Study Based on Modified Booth Multiplier Using Wallace Tree Structure

Manas M. Ramteke[1] Prof. P. R. Indurkar[2] Prof. Mrs. D. M. Khatri[3]
[1]Student [2,3]Professor
[1,2,3]Department of Electronics & Telecommunication Engineering
[1,2,3]B.D.C.O.E., Sevagram, Wardha, India

Abstract— Multiplication is a one of the most significant operation in digital system design. The speed performance of multiplication influents overall performance in digital computer systems. Many high performance algorithms and architectures have been proposed in various literatures which improve and accelerate multiplication operations. After analysis of these literatures it may be concluded as Wallace tree structure provides less delay, less number of levels, less total number of compressors and less generation of partial products as compared to conventional multipliers. Designing a multiplier includes generation of partial products and summation of partial products. To improve the speed performance of multiplication, number of partial products has been reduced and for reducing the delay of summation of partial products Wallace Tree Structure has been used.

Key words: Wallace tree structure, VLSI circuit design and implementations, Wallace structure multiplier

I. INTRODUCTION

The various aspects of multiplier architectures have been published in the literature, during the past few decades. The multiplier is one of the basic building blocks in most of the digital and high performance systems such as digital signal processors and microprocessors. With the recent advances in modern technology, many researchers have worked on the design of increasingly more efficient multipliers. They focused on offering higher speed and lower power consumption even while occupying reduced silicon area. This makes them more compatible for various complex and portable VLSI circuit design and implementations.

However, the fact remains the same that the area and speed are two conflicting performance constraints. Hence increased speed always results in larger area. In these papers, we may arrive at a better trade-off between the two, by realizing a marginally increased speed performance through a small increase in the number of transistors. The innovative architecture enhances the speed performance of the widely acknowledged Wallace tree multiplier [6].

The Main objective of this paper is completely based on study of speed performance of multiplication in modified Booth algorithm and Wallace Structure. Compressors used in Wallace tree structure accumulate partial products. Because of these compressors, no. of levels has been decreased that also increase the speed of multiplier. In this paper, multiplier provides less delay, takes less no. of levels in Wallace tree structure and also less no. of compressor.

II. WALLACE TREE STRUCTURE

Wallace tree [4] is known for their optimal computation time, when adding multiple operands to two outputs using 3:2 or 4:2 compressors or both. Wallace tree guarantees the lowest overall delay. Figure1 shows nine operands Wallace structure, where 3:2 compressors compress the data having three multi-bit inputs and two multi-bit outputs. 4:2 compressors compress the data having four multi-bit inputs and two multi-bit outputs. Using both compressor, No. of levels has been reduced that also causes enhancing the speed of multiplier.

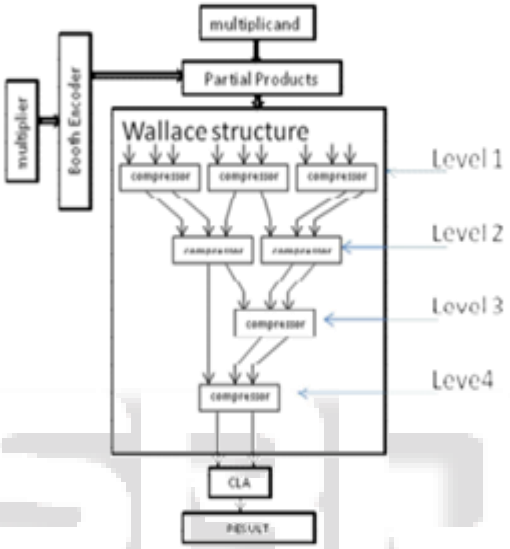

Fig. 1: Wallace structure multiplier

III. TYPES OF COMPRESSORS

These compresses the data which helps to reduce the number of levels which also causes enhancing the speed of multiplier. There are various types of compressors [5],[6] which are used in Booth multiplier.

A. 3:2Compressor Architecture:

The 3:2 compressor takes 3 inputs x1, x2, x3 and generates 2 outputs, the sum bit s, and the carry bit c as shown in Figure 2(a). The compressor is governed by the basic equation

$$x1 + x2 + x3 = Sum + 2 * Carry$$

The 3:2 compressors can also be employed as a full adder cell when the third input can be considered as the Carry input from the previous compressor block or $x3 = Cin$. Existing architectures shown in Figure 2(b) employ two XOR gates in the critical path. The equations governing the existing 3:2 compressor outputs are shown below:

$$Sum = x1 \text{ XOR } x2 \text{ XOR } x3$$

$$Carry = (x1 \text{ XOR } x2)\, x3 + (x1' \text{ XOR } x2')\, x1$$

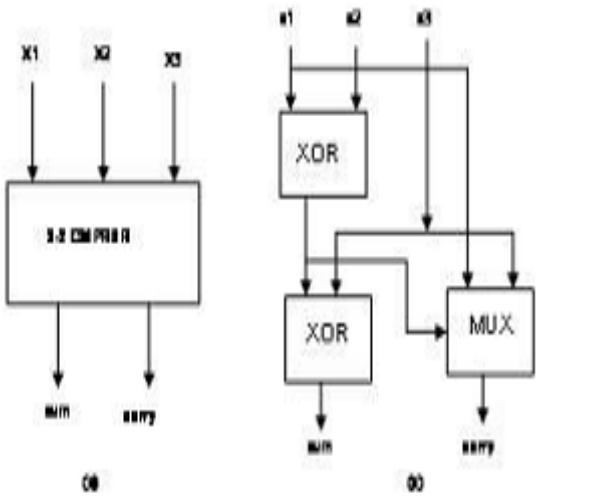

Fig. 2: 3:2 compressor block diagram and architecture

B. 4:2Compressor Architecture:

The 4-2 compressor has 4 inputs X1, X2, X3 and X4 and 2 outputs Sum and Carry along with a Carry-in (Cin) and a Carry-out (Cout) as shown in Figure 3(a). The input Cin is the output from the previous lower significant compressor. The Cout is the output to the compressor in the next significant stage.

Similar to the 3:2 compressors the 4:2 compressors in Figure 3(a) is governed by the basic equation given below:

x1+x2+x3+x4+Cin = Sum + 2(Carry + Cout)*

The standard implementation of the 4-2 compressor is done using two 3:2 compressors as shown in Figure 3(b).

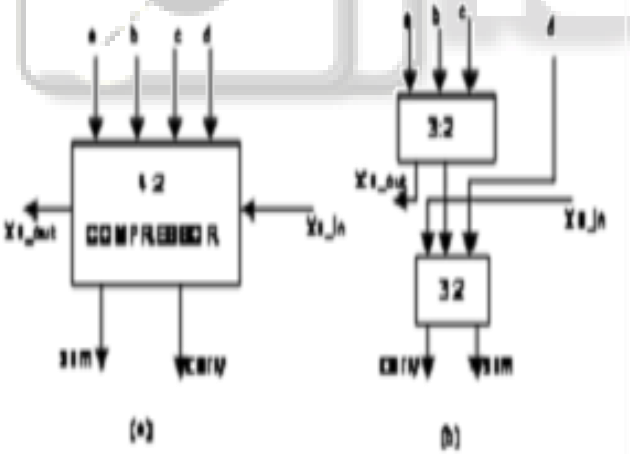

Fig. 3: 4:2 Compressor block diagram

When the individual full Adders are broken into their constituent XOR blocks, it can be observed that the overall delay is equal to 4*Δ-XOR. The block diagram in Figure 4 shows the existing architecture for the implementation of the 4:2 compressor with a delay of 3*Δ-XOR. The equations governing the outputs in the existing architecture are shown below:

Sum = x1 XOR x2 XOR x3 XOR x4 XOR Cin
Cout = (x1 XOR x2) x3 + (x1 XOR x2)' x1
Carry = (x1 XOR x2 XOR x3 XOR x4) Cin + (x1 XOR x2 XOR x3 XOR x4)'x4

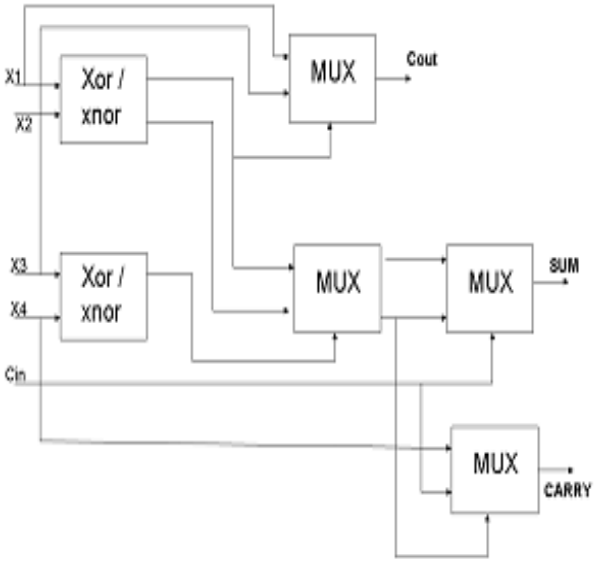

Fig. 4: 4:2Compressor architecture

C. 5:2Compressor Architecture:

The 5-2 Compressor block has 5 inputs X1,X2,X3,X4,X5 and 2 outputs, Sum and Carry, along with 2 input carry bits (Cin1, Cin2) and 2 output carry bits (Cout1,Cout2) as shown in Figure 5(a). The input carry bits are the outputs from the previous lesser significant compressor block and the output carry are passed on to the next higher significant compressor block. The basic equation that governs the function of the 5:2 compressor block shown in Figure 5 is given below:

x1+x2+x3+x4+x5+cin1+cin2 = sum+2(carry + cout1 + cout2).*

The conventional implementation of the compressor block is shown in Figure 5(b) where 3 cascaded full adder cells are used. When these full adders are replaced with their constituent blocks of XOR gates then it can be observed that the overall delay is equal to 6*Δ-XOR for the sum or carry output. The architectures have been proposed by where the delay has been reduced to 4* Δ-XOR as shown in Figure 6. The equations governing the outputs are shown below:

Sum = x1 x2 x3 x4 x5 cin1 cin2
Cout1 = (x1 + x2) x3 + x1 x2
Cout2 = (x4 x5) cin1 + (x4 x5) x4
Carry = ((x1 x2 x3) (x4 x5 cin1)) cin2 +
((x1 x2 x3) (x4 x5 cin1)) (x1 x2 x3)

The critical path delay of the proposed implementation is Δ-XOR + 3*Δ-MUX. The final stage in the Wallace tree multiplier for addition of partial products can be further reduced by the use of tree adders. But here we have used the default adder present in FPGA.

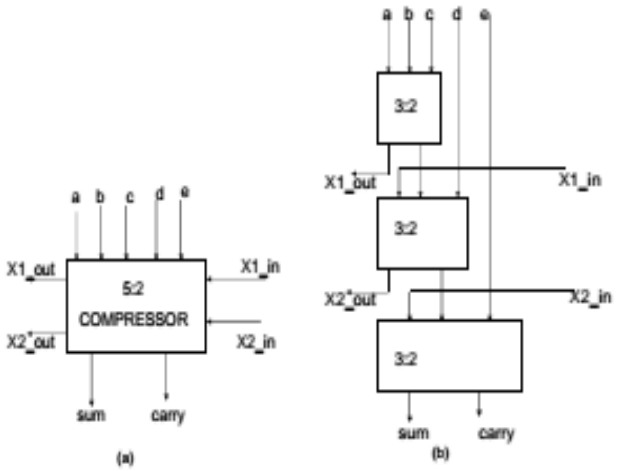

Fig. 5: 5:2 Compressor block diagram

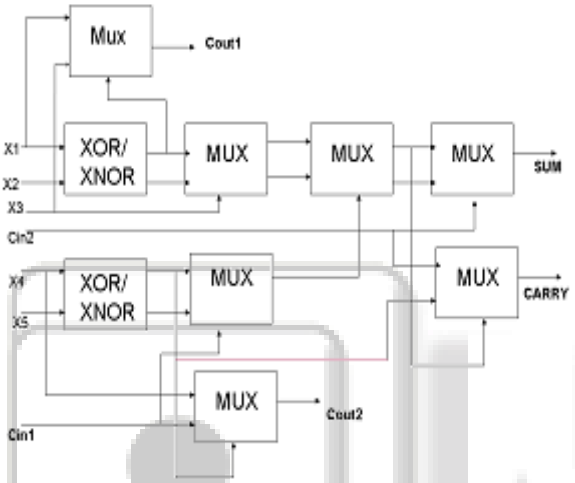

Fig. 6: 5:2Compressor architecture

IV. RADIX-4 AND RADIX-8 MULTIPLICATION

Recoding of binary numbers was first introduced by Booth four decades ago [6]. MacSorley proposed a modification of Booth's algorithm a decade after. The modified Booth's algorithm (*radix-4 recoding*) starts by appending a zero to the right of *x0* (multiplier LSB). Triplets are taken beginning at position x_{-1} and continuing to the MSB with one bit overlapping between adjacent triplets. If the number of bits in X (excluding x_{-1}) is odd, the sign (MSB) is extended one position to ensure that the last triplet contains 3 bits. In every step we will get a signed digit that will multiply the multiplicand to generate a partial product entering the Wallace reduction tree. The meaning of each triplet can be seen in figure 7 as shown below:

x_{i+2} x_{i+1} x_i	Partial product
0 0 0	0Y
0 0 1	+1Y
0 1 0	+1Y
0 1 1	+2Y
1 0 0	-2Y
1 0 1	-1Y
1 1 0	-1Y
1 1 1	0Y

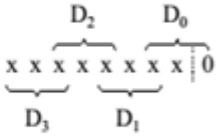

Fig. 7: Radix-4 recoding

This recoding scheme applied to a parallel multiplier halves the number of partial products so the multiplication time and the hardware requirements decrease. This gain is possible at the expense of somewhat more complex operations in every step. It should be noted, however, that the required multiples of Y {0, ±Y, ±2Y} are available by merely shifting Y to the left. Although the algorithms and operations specified above seem rather arbitrary at the first sight, they are based on meaningful number systems. If one focuses on what modifications are being done to X, then one may arrive at a different representation for the 2s-complement number X as shown in figure 8:

$$X = \sum_{i=0}^{\frac{n}{2}-1} D_i \cdot 4^i$$

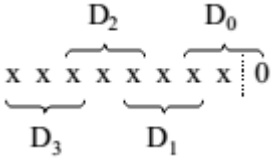

Fig. 8: Signed digit representation

Where digits Di are one of {-2, -1, 0, 1, 2} found in the table of figure 7, based on the value of triplets in the form (xi+2 xi+1 xi). Here we have a signed digit representation of X in radix-4. Signed-digit number representation allows redundancy to exist. Thanks to this we can make a parallel recodification that is, all triplets are recoded at the same time, and the value of each triplet is independent from the adjacent triplets.

Radix-8 recoding [8] applies the same algorithm as radix-4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signed-digit using the table of figure 9:

Quartet value	*Signed-digit value*
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

Fig. 9: Signed digit representation

V. WALLACE TREE MULTIPLIER USING BOOTH ENCODER

With the recent advances in technology, many researchers have worked on the design of increasingly more efficient multipliers. They aim at offering higher speed and lower power consumption even while occupying reduced silicon area. This makes them compatible for various complex and portable VLSI circuit implementations. However, the fact remains that the area and speed are two conflicting performance constraints. Hence, innovating increased speed always results in larger area. In this paper, we arrive at a better trade-off between the two, by realizing a marginally increased speed performance through a small rise in the number of transistors.

The new architecture [5] enhances the speed performance of the widely acknowledged Wallace tree

multiplier. The structural optimization is performed on the conventional Wallace multiplier, in such a way that the latency of the total circuit reduces considerably. The Wallace tree basically multiplies two unsigned integers. The conventional Wallace tree multiplier architecture comprises of an AND array for computing the partial products, a carry save adder for adding the partial products so obtained and a carry propagate adder in the final stage of addition. In the proposed architecture, partial product generation and reduction is accomplished by the use of booth algorithm, 3:2, and 4:2, 5:2 compressor structures.

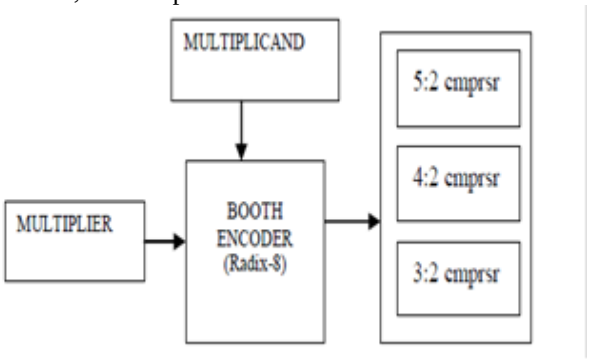

Fig. 10: Signed digit representation

In the modified Booth algorithm, multiplier has been divided in groups of 4 bits and each group of 4 bits has been operational according to modified Booth Algorithm for generation of partial products as follows:

0, $\pm$1A,$\pm$2A,$\pm$3A,$\pm$4A,$\pm$5A,$\pm$6A,$\pm$7A

Where A= Multiplicand operand

For n bit multiplier no. of groups = n/3

Consider a 32 bit multiplier operand (B) as B31B30B29……$B_2B_1B_0$. In radix-8 Booth algorithm, multiplier operand B is partitioned into 11 groups having each group of 4 bits. In first group, first bit is taken 0 and other 3 bits are LSB of multiplier operand. In second group first bit is MSB of first group and other bits are next 3 bit of multiplier operand.

After performing said algorithm the pattern of groups result as

$B_5B_4B_3B_2$ $B_2B_1B_00$

Similarly the third group is formed which is shown below

$\mathbf{B_8B_7B_6B_5}$ $B_5B_4B_3B_2$ $B_2B_1B_00$

And finally the complete pattern of groups will be as follows:

$0B_{31}B_{30}B_{29}$ ……$\mathbf{B_8B_7B_6B_5}$ $B_5B_4B_3B_2$ $B_2B_1B_00$

Wallace tree using 3:2 and 4:2 compressors, radix-32 modified Booth Algorithm improve the speed of the proposed multiplier because radix-32 reduces no. of partial products, both 3:2 and 4:2 compressor reduces no. of levels in Wallace structure.

Wallace tree multiplier using booth algorithm is very a good technique for high speed applications, its implementation with different logics in VLSI [1]. Further the work can be extended for optimization of said multiplier to improve the power. It can be concluded that the use of Radix-4 modified Booth algorithm and Wallace tree structure lead to a better performance in throughput speed and area. The utilizations of CSA adders in performing addition process for partial product rows also have been shown to greatly influence the speed of the system.

VI. Conclusion

Wallace tree multiplier using booth algorithm is very a good technique for high speed applications, its implementation with different logics in VLSI. Further the work can be extended for optimization of said multiplier to improve the power. From overall analysis of literature review we realize that improvement can be done by modifying the structure using Modified Booth algorithm and Wallace structure for radix 32 multiplications.

Modifications can be done to obtain following purposes:

(1) Number of levels should be reduced.
(2) Number of compressors should be reduced.
(3) Generation of partial products should be reduced.
(4) Speed of multiplication process should be increased.

VII. Acknowledgment

I wish to thank Prof. P. R. Indurkar and Prof. Mrs. D. M. Khatri for their guidance and support.

References

[1] High Performance Complex Number Multiplier Using Booth-Wallace Algorithm by Rizalafande Che Ismail and Razaidi Hussin, ICSE2006 Proc. 2006, Kuala Lumpur, Malaysia.
[2] Power Aware and High Speed Reconfigurable Modified Booth Multiplier by S. Sri Sakthi and N. Kayalvizhi, IEEE 2011.
[3] Disposition (reduction) of (negative) partial product for Radix 4 Booth's Algorithm by Manoj Sharma and Richa Verma, IEEE 2011.
[4] High performance pipelined signed 64x64-bit multiplier using radix-32 modified Booth algorithm and Wallace structure by Manish Bansal, Sangeeta Nakhate, and Ajay Somkuwar IEEE 2011.
[5] A High Speed Wallace Tree Multiplier Using Modified Booth Algorithm for Fast Arithmetic Circuits Jagadeshwar Rao M and Sanjay Dubey, IOSR Journal of Electronics and Communication Engineering (IOSRJECE) Sept. 2012.
[6] A High Speed and Area Efficient Booth Recoded Wallace Tree Multiplier for fast Arithmetic Circuits by Jagadeshwar Rao M and Sanjay Dubey, 2012 Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PRIMEASIA).
[7] High speed Modified Booth Encoder multiplier for signed and unsigned numbers by Ravindra P. Rajput and M. N. Shanmukha Swamy, 2012 14th International Conference on Modelling and Simulation.
[8] Design of a Low-Error Fixed-Width Radix-8 Booth Multiplier by Saroja S. Bhusare and V. S. Kanchana Bhaaskaran, IEEE 2013.
[9] Implementation of Pipelined Booth Encoded Wallace Tree Multiplier Architecture by Rahul D. Kshirsagar, Aishwarya E. V., Ahire Shashank, Vishwanath and P. Jayakrishnan, IEEE 2013.

