High Speed Modified Booth's Signed 64x64 Bit Multiplier Using Wallace Structure by Radix-32

Manas M. Ramteke
Deptt. of E & TC, B.D.C.O.E.,
Sevagram, Wardha, India
manasramteke87@gmail.com

Prof. P. R. Indurkar[1], Prof. Mrs. D. M. Khatri[2]
Deptt. of E & TC, B.D.C.O.E.,
Sevagram, Wardha, India
[1]*prashantindurkar@rediffmail.com*
[2]*deepali.85@rediffmail.com*

Abstract — The Main objective of the implemented work is completely based on enhancing speed performance multiplication process using radix-32 modified Booth algorithm and Wallace Tree Structure. It is designed for fixed length 64x64 bit operands. In Wallace structure, 3:2and 4:2 Compressors are used which accumulate the partial products. The implemented modified Booth multiplier is verified and advantages over the existing multiplier are quantitatively analyzed. This implemented multiplier provides less delay 0.238 ns. Many researchers had been worked and presented the modified booth multiplier with optimized delay. In this paper, it has been shown that the implemented 64 bit multiplier provides better delay in comparison with those existing papers. A VHDL code has been written and successfully synthesized and simulated using Xilinx ISE 13.1 simulator software. Also partial products which are generated are less as compared to conventional multiplier. No. of logic blocks required for fast multiplication process has been reduced in terms of no. of slices in comparison with previous ones.

Keywords*- Radix-32, 3:2 Compressor, 4:2 Compressor ,Wallace Tree*

*****

I. INTRODUCTION

Multipliers are basic elements in several applications in engineering, semiconductor technology and digital signal processing. The overall performance of digital signal processing can be improved by increasing the speed performance of multiplication. In the implemented algorithm, factors of designing multiplier are nothing but generation and summation of partial products. These partial products are summed using compressor in structure of Wallace Tree. CLA has been used for final results where CLA indicates carry look ahead adder that ahead carry of compressor. Thus to improve the speed performance of multiplication, number of partial products have been reduced by using radix-32 Booth Algorithm and Wallace structure is for delay reduction. After studying many literature survey of researchers we have concluded that the decision of using radix-32 is better than radix-16 as radix-32 produces less no. of partial products.

II. LITERATURE SURVEY

A. Wallace Tree Structure

Wallace C.S. (1964) introduced a fast technique to perform multiplication of large operands. Unlike an array multiplier, the partial product matrix for a tree-multiplier is rearranged in a tree-like format, reducing both the critical path and the number of adder cells needed. The Wallace tree multiplier belongs to a family of multipliers called column compression multipliers. The basic principle in this family of multipliers is to achieve partial product accumulation by successively reducing the number of bits of information in each column using full adders or half adders. The compressors in the Wallace structure are used to compress the data and thus produce less no. of partial products. The Wallace tree consists of numerous levels of such column compression structures until finally, only two full-width operands remain. These two operands can then be added using fast carry look ahead adder to obtain the product result. Thus the Wallace tree multiplier uses as much hardware as possible to compress the partial product matrix as quickly as possible into the final product.

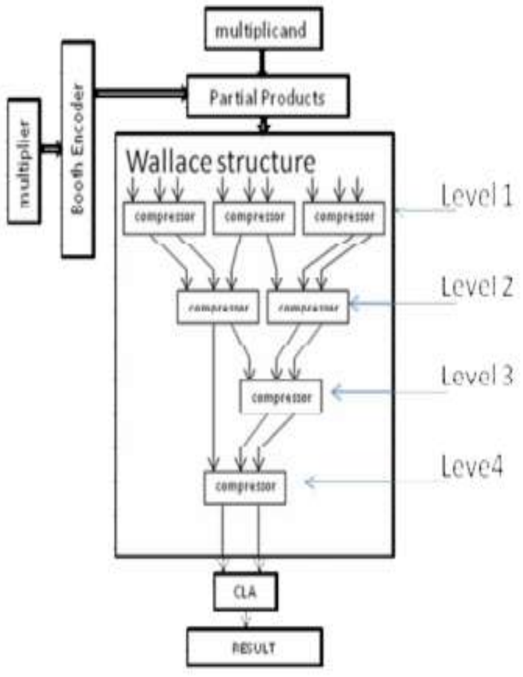


Fig 1. Wallace structure multiplier[4]

B. Types of Compressors

This 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 which are used in modified Booth multiplier.

a) 3:2 Compressor Architecture [5]

The 3:2 compressor [5] 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 architecture is shown in Figure 2. Employ two XOR gates in the critical path. The equations governing the existing 3:2 compressor outputs are shown below:

Sum = x1 XOR x2 XOR x3
Carry = (x1 XOR x2) x3 + (x1' XOR x2') x1

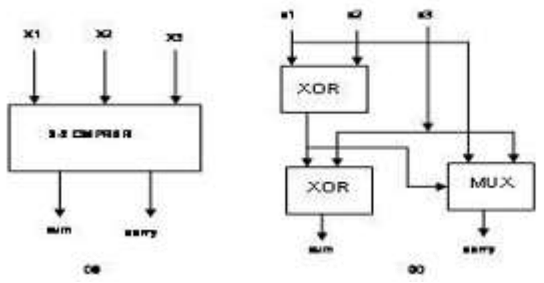


Fig 2. 3:2 Compressor block diagram and architecture

b) 4:2 Compesor Architecture

The 4:2 compressors [6] 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. 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 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.

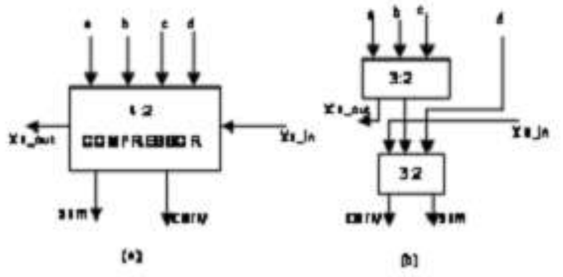


Fig 3. 4:2 compressor block diagram

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

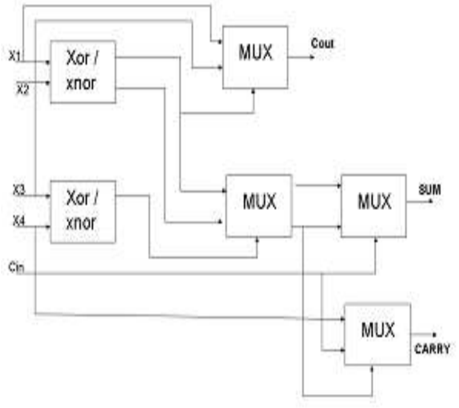


Fig 4. 4:2 compressor architecture [6]

III. DESIGN METHODOLOGY

A. Radix-32 Implemented Modified Booth Algorithm

To generate and reduce the number of partial products of multiplier, proposed modified Booth Algorithm has been used, In the proposed modified Booth Algorithm, multiplier has been divided in groups of 6 bits and each groups of 6 bits have been operationed according to modified Booth Algorithm for generation of partial products 0, ±1A, ±2A, ±3A, ±4A, ±5A, ±6A, ±7A, ,±8A, ±9A, ±10A, ±11A, ±12A, ±13A, ±14A, ,±15A, ±16A. These partial products are summed using compressor in structure of Wallace Tree. CLA has been used for final results where CLA indicates carry look ahead adder that ahead carry of compressor.

In radix-32 Booth Algorithm [4], multiplier operand B is partitioned into groups having each group of 6 bits. In first group, first bit is taken zero and other bits are least significant five bit of multiplier operand. In second group, first bit is most significant bit of first group and other bits are next five bit of multiplier operand. In third group, first bit is most significant bit of second group and other bits are next five bit of multiplier operand. This process is carried on.

For each group, Partial product is generated using multiplicand operand A. For n bit multiplier there is n/5 or [n/5 + 1] groups and partial products in proposed modified Booth Algorithm radix-32. Table is shown for Proposed radix-32 modified Booth algorithm has been designed and radix-32. So it reduces the number of partial products in comparison to radix-16, improves the computational efficiency of multiplier, reduce the calculation delay. Computation of complex multiplier and re-encoding of multiplier can be executed in parallel. Factor Fi is calculated using equation and figure 5. The recoding of multiplier by radix-32 is shown in table I [4].

$$Fi = (k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5)$$

Where $k_1,k_2,k_3,k_4,k_5 = 0$ when 00 or 11

$k_1,k_2, k3,k4,k5 = +1$ when 01

$k_1, k2,k3,k4,k5 = -1$ when 10

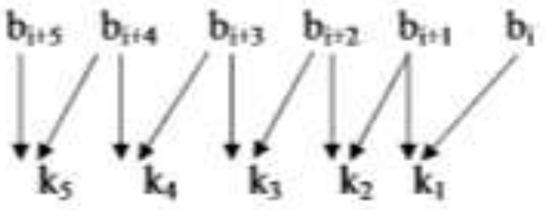
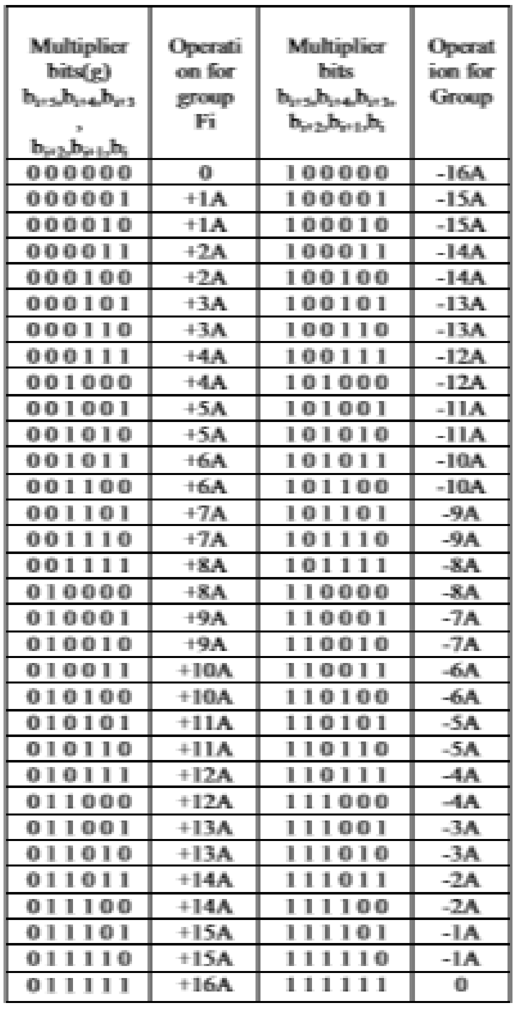


Fig 5. finding value of k_1, k_2, k_3, k_4, k_5



Multiplier bits(g) $b_{i+5},b_{i+4},b_{i+3}, b_{i+2},b_{i+1},b_i$	Operation for group Fi	Multiplier bits $b_{i+5},b_{i+4},b_{i+3}, b_{i+2},b_{i+1},b_i$	Operation for Group
000000	0	100000	-16A
000001	+1A	100001	-15A
000010	+1A	100010	-15A
000011	+2A	100011	-14A
000100	+2A	100100	-14A
000101	+3A	100101	-13A
000110	+3A	100110	-13A
000111	+4A	100111	-12A
001000	+4A	101000	-12A
001001	+5A	101001	-11A
001010	+5A	101010	-11A
001011	+6A	101011	-10A
001100	+6A	101100	-10A
001101	+7A	101101	-9A
001110	+7A	101110	-9A
001111	+8A	101111	-8A
010000	+8A	110000	-8A
010001	+9A	110001	-7A
010010	+9A	110010	-7A
010011	+10A	110011	-6A
010100	+10A	110100	-6A
010101	+11A	110101	-5A
010110	+11A	110110	-5A
010111	+12A	110111	-4A
011000	+12A	111000	-4A
011001	+13A	111001	-3A
011010	+13A	111010	-3A
011011	+14A	111011	-2A
011100	+14A	111100	-2A
011101	+15A	111101	-1A
011110	+15A	111110	-1A
011111	+16A	111111	0

Table I. Proposed radix-32 booth's algorithm [4]

B. *Design process for implemented modified multiplier*

In this paper, the design process of implementation of 64x64-bit operands multiplier has been explained. Modified Booth Algorithm is used to enhance the speed and it is based on radix-32 which produces less number of partial products. Booth encoder matches the 6 bits of multiplier with its recoded bits and provides the multiplier value has shown in table I. After generation of partial products, 3:2 and 4:2 compressors compresses the data which results in less no. of levels required for overall multiplication process. In Wallace tree structure summation delay is less because of using both 3:2 compressor and 4:2 compressor in summation operation. Again CLA has been used for final results where CLA indicates carry look ahead adder that ahead carry of compressor.

C. *Simulation result*

For VHDL code of High speed 64x64 bit multiplier using radix-32, Xilinx ISE 13.1 has been used for synthesizing and simulation. By using this tool, VHDL code has been successfully synthesized and simulated..The result of simulation has been shown in figure 6.

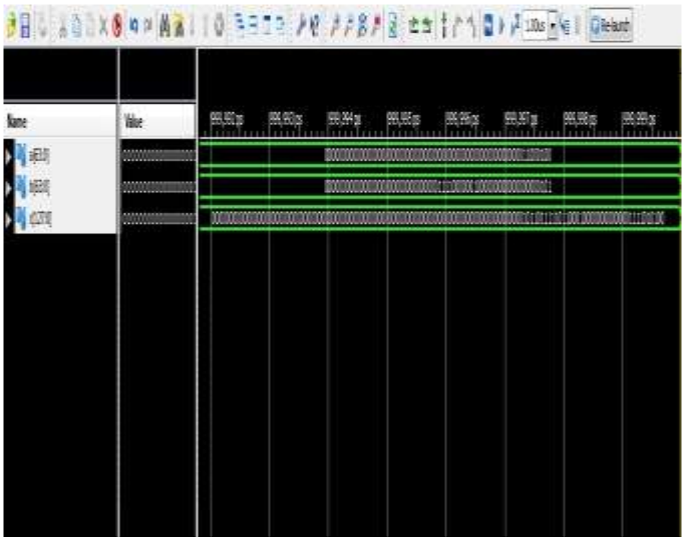


Fig 6. Simulation result for 64X64 bit modified booth's multiplier using radix-32.

D. *RTL schematic view for proposed algorithm*

After successful simulation, the rtl view for implemented algorithm is shown below.

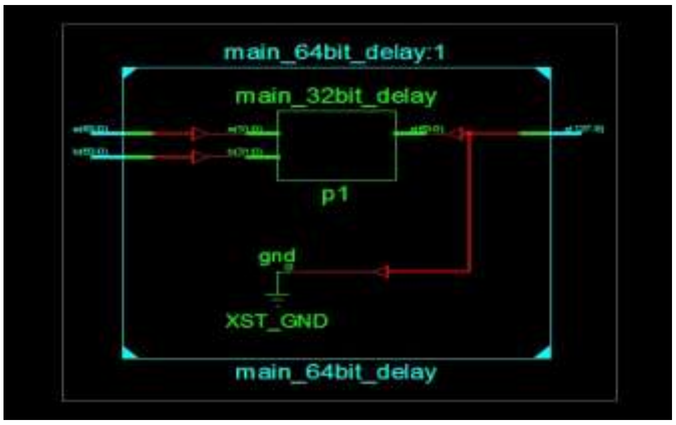


Fig 7. RTL view for implemented algorithm.

E. Comparison Table

Comparison among various multipliers and proposed multiplier is shown in table II below [4].

Multiplier	Delay in ns	Slices used
64X64 bit multiplier using radix-4 and array structure	28.96	14271/69120 (20%)
High speed parallel 32X32 bit multiplier using radix-16 booth encoder	7.55	182/2352 (7%)
64X64 bit multiplier	4.88	-
Signed 64X64 bit multiplier using radix 16 booth algorithm	1.8	3347/69120 (3%)
Signed 64X64 bit multiplier using radix-32 booth algorithm.	1.4	3677/69120 (5.31%)
Proposed high speed signed 64X64 bit multiplier using radix-32 booth algorithm.	0.284	1704/69120 (2%)

Table II . Comparison among multipliers[4]

IV. CONCLUSION

In this paper we have designed 64x64 bit multiplier using radix-32 Modified Booth Algorithm and Wallace structure. It is observed that compressors in the Wallace tree improves the speed of implemented multiplier and radix-32 reduces no. of partial products.

Implemented design provides less delay 0.284 ns and requires less number of levels of Wallace tree structure. No. of logic blocks required for fast multiplication process has been reduced in terms of no. of slices in comparison with previous ones.

Since delay and area has been reduced hence it can be concluded that the overall power consumption will get reduced when implemented for low power VLSI design.

ACKNOWLEDGMENT

I wish to thank Prof. P. R. Indurkar and Prof. Mrs. D. M. Khatri for their guidance and support. I also thankful to Manish Bansal, Sangeeta Nakhate and Ajay Somkuwar for their paper in IEEE 2011 which help me in my implemented work and for successful result.

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.