International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 9, September 2015)

A High Speed Wallace Tree Multiplier Using Modified Booth Algorithm for Fast Arithmetic Circuits

S Sapna
Digital Communication and networking
P A College of Engineering
Mangalore, India
sapna00765@gmail.com

Vamshi U R
Digital Communication and networking
P A College of Engineering
Mangalore, India
vamshivalambra@gmail.com

***Abstract*— Designing multipliers that are of high-speed, low power, and regular in layout are of substantial research interest. Speed of the multiplier can be increased by reducing the generated partial products. Many attempts have been made to reduce the number of partial products generated in a multiplication process; one of them is Wallace tree multiplier. Wallace Tree CSA structures have been used to sum the partial products in reduced time. In this paper Wallace tree construction is investigated and evaluated. Speed of traditional Wallace tree multiplier can be improved by using compressor techniques. In this paper Wallace tree is constructed by traditional method and with the help of compressor techniques such as 4:2 compressor, 5:2 compressor, 6:2 compressor, 7:2 compressor. Therefore, minimizing the number of half adders used in a multiplier reduction will reduce the complexity.**

***Index Terms*—Component, formatting, style, styling, insert.** *(key words)*

I. INTRODUCTION

A multitude of various multiplier architectures have been published in the literature, during the past few decades. The multiplier is one of the key hardware blocks in most of the digital and high performance systems such as digital signal processors and microprocessors. 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 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 booth algorithm, 3:2, and 4:2, 5:2.

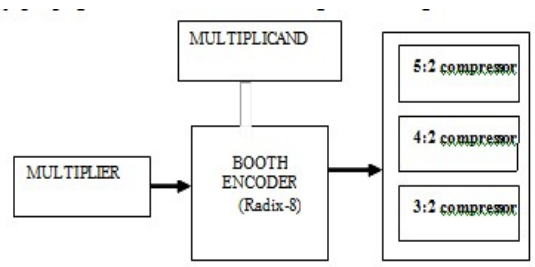

Fig 1 Proposed Architecture of Wallace tree multiplier using booth encoder

II. IMPLEMENTATION OF WALLACE TREE MULTIPLIER

A Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers, devised by an Australian Computer Scientist Chris Wallace in 1964.

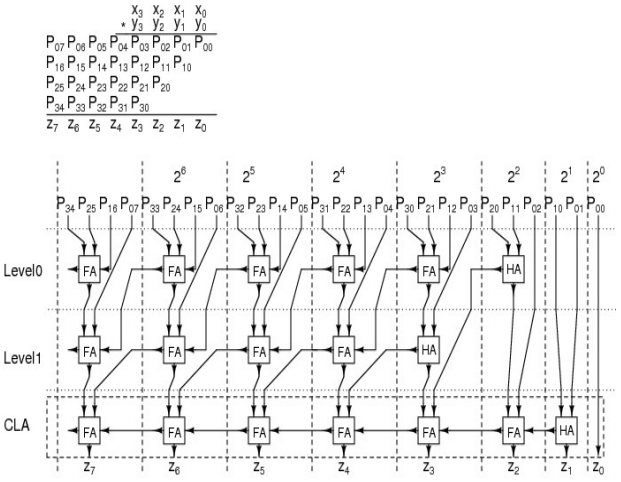

Fig 2. Implementation of Wallace tree multiplier.

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 9, September 2015)

The Wallace tree has three steps:

Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding results. Depending on position of the multiplied bits, the wires carry different weights, for example wire of bit carrying result of is 32 (see explanation of weights below).

Reduce the number of partial products to two by layers of full and half adders.

Group the wires in two numbers, and add them with a conventional adder. The second phase works as follows. As long as there are three or more wires with the same weight add a following layer:

- Take any three wires with the same weights and input them into a full adder.
- The result will be an output wire of the same weight and an output wire with a higher weight for each three input wires.

If there are two wires of the same weight left, input them into a half adder. If there is just one wire left, connect it to the next layer.

The benefit of the Wallace tree is that there are only reduction layers, and each layer has propagation delay.

As making the partial products is and the final addition is, the multiplication is only, not much slower than addition (however, much more expensive in the gate count).

Naively adding partial products with regular adders would require time. From a complexity theoretic perspective, the Wallace tree algorithm puts multiplication in the class NC1.

These computations only consider gate delays and don't deal with wire delays, which can also be very substantial. The Wallace tree can be also represented by a tree of 3/2 or 4/2 adders.

Example:

N=4 , multiplying a3a2a1a0 by b3b2b1b0

1. First we multiply every bit by every bit:

weight 1 – a0b0
weight 2 –a0b1 ,a1b0
weight 4 – a0b2,a1b1 ,a2b0
weight 8 –a0b3 ,a1b2,a2b1,a3b0
weight 16 –a1b3,a2b2,a3b1
weight 32 – a2b3,a3b2
weight 64 – a3b3.

2. Reduction layer 1:

Pass the only weight-1 wire through, output: 1 weight-1 wire

Add a half adder for weight 2, outputs: 1 weight-2 wire, 1 weight-4 wire

Add a full adder for weight 4, outputs: 1 weight-4 wire, 1 weight-8 wire

Add a full adder for weight 8, and pass the remaining wire through, outputs: 2 weight-8 wires, 1 weight-16 wire

Add a full adder for weight 16, outputs: 1 weight-16 wire, 1 weight-32 wire

Add a half adder for weight 32, outputs: 1 weight-32 wire, 1 weight-64 wire

Pass the only weight-64 wire through, output: 1 weight-64 wire

3. Wires at the output of reduction layer 1:

weight 1 - 1
weight 2 - 1
weight 4 - 2
weight 8 - 3
weight 16 - 2
weight 32 - 2
weight 64 - 2

4. Reduction layer 2:

Add a full adder for weight 8, and half adders for weights 4, 16, 32, 64

5. Outputs:

weight 1 - 1
weight 2 - 1
weight 4 - 1
weight 8 - 2
weight 16 - 2
weight 32 - 2
weight 64 - 2
weight 128 - 1

6. Group the wires into a pair integers and an adder to add them.

III. WALLACE TREE MULTIPLIER

A fast process for multiplication of two numbers was developed by Wallace. Using this method, a three step process is used to multiply two numbers; the bit products are formed, the bit product matrix is reduced to a two row matrix where sum of the row equals the sum of bit products, and the two resulting rows are summed with a fast adder to produce a final product.

In the Wallace Tree method, three bit signals are passed to a one bit full adder (“3W”) which is called a three input Wallace Tree circuit, and the output signal (sum signal) is supplied to the next stage full adder of the same bit, and the carry output signal thereof is passed to the next stage full adder of the same no of bit, and the carry output signal thereof is supplied to the next stage of the full adder.

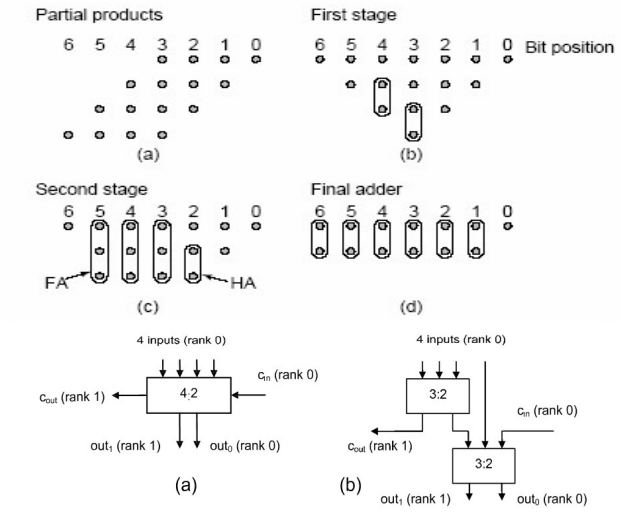

Fig 3 Multipliers

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 9, September 2015)

An 8-bit multiplier is constructed by using Wallace tree architecture .The architecture has been shown in Fig .Partial products are added in 6 steps. In the Wallace Tree method, the circuit layout is not easy although the speed of the operation is high since the circuit is quite irregular The delay generated in wallace tree multiplier can be further reduced by using modified tree structures called compressors.

IV. COMPRESSOR

Compressors are arithmetic components, similar in principle to parallel counters, but with two distinct differences: (1) they have explicit carry-in and carry-out bits; and (2) there may be some redundancy among the ranks of the sum and carry-output bits.

4.1 [4:2] Compressor

The 4:2 compressor has 4 input bits and produces 2 sum output bits (out0 and out1), it also has a carry-in (cin) and a carry-out (cout) bit (thus, the total number of input/output bits are 5 and 3); All input bits, including cin, have rank 0; the two output bits have ranks 0 and 1 respectively, while cout has rank 1 as well. Thus, the output of the 4:2 compressor is a redundant number; for example, out1 = 0 and cout = 1 is equivalent to out1 = 1 and cout = 0 in all cases.

4.2 [5:2] Compressor

The 5:2 compressor has 5 input bits and produces 2 sum output bits (sum and cout3), it also has a carry-in (cin1, cin2) and a carry-out (cout1, cout2,cout3) bit (thus, the total number of input/output bits are 7 and 4); All input bits, including cin1, have rank 0 and cin2 has rank 2 ; the two output bits have ranks 0 and 1 respectively, while cout2 has rank 1 and cout1 has rank 2.

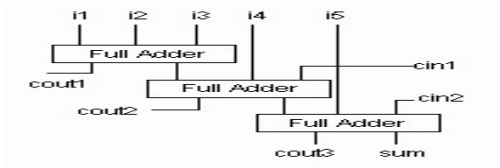

Fig 4. [5:2] compressor I/O diagram.

4.3 [6:2] Compressor

The 6:2 compressor has 6 input bits and produces 2 sum output bits (out0 and out1), it also has a carry-in (Cin0, Cin1) and a carry-out (Cout0, Cout1) bit (thus, the total number of input/output bits are 8 and 4); All input bits, including Cin0, have rank 0 and Cin1 has rank 1 ; the two output bits have ranks 0 and 1 respectively, while Cout0 has rank 1 and Cout1 has rank 2 as shown in Fig 4.2.

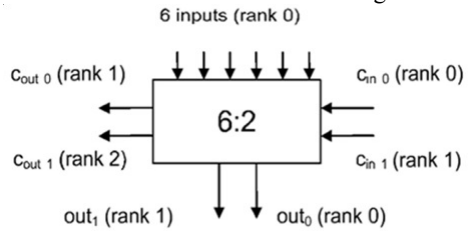

Fig 5. [6:2] compressor I/O diagram.

4.4 [7:2] Compressor

The 7:2 compressor has 7 input bits and produces 2 sum output bits (out0 and out1), it also has a carry-in (Cin0, Cin1) and a carry-out (Cout0, Cout1) bit (thus, the total number of input/output bits are 9 and 4); All input bits, including Cin0, have rank 0 and Cin1 has rank 1 ; the two output bits have ranks 0 and 1 respectively, while Cout0 has rank 1 and Cout1 has rank 2 as shown in Fig 4.3.

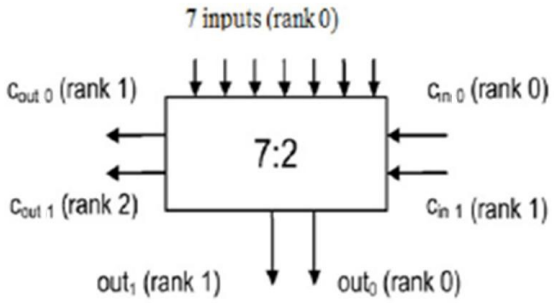

Fig 6. [7:2] Compressor.

V. BOOTH ALGORITHM FOR PARTIAL PRODUCTS GENERATION

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 4 bits and each groups of 4 bits have been operational according to modified Booth Algorithm for generation of partial products 0±1A, ±2A, ±3A, ±4A, ±5A, ±6A, ±7A . These partial products are summed using compressors in structure of Wallace Tree .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 zero and other bits are least Significant three bit of multiplier operand. In second group, first bit is most significant bit of first group and other bits are next three bit of multiplier operand. In third group, first bit is most significant bit of second group and other bits are next three bits 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/3 or [n/3 + 1] groups and partial products in proposed modified Booth Algorithm radix-8. Table I for Proposed radix-8 modified Booth algorithm has been designed.

VI. COMPRESSOR FOR PARTIAL PRODUCTS REDUCTION

The latency in the Wallace tree multiplier can be reduced by decreasing the number of adders in the partial products reduction stage. In the proposed architecture, multi bit compressors are used for realizing the reduction in the number of partial product addition stages. The combined factors of low power, low transistor count and minimum delay makes the 5:2 , 4:2 and 3:2 compressors, the appropriate choice. In these compressors, the outputs. generated at each stage are efficiently used by replacing the

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 9, September 2015)

XOR blocks with multiplexer blocks so that the critical path delay is minimized. The various adder structures in the conventional architecture are replaced by compressors.

VII. COMPRESSOR ARCHITECTURE

A 3-2 compressor takes 3 inputs x1, x2, x3 and generates 2 outputs, the sum bit s, and the carry bit .The compressor is governed by the basic equation

$x + x + x = Sum + 2 * Carry$

The 3-2 compressor can also be employed as a full adder cell when the third input is considered as the Carry input from the previous compressor block or X3 = C . Existing architectures shown in Fig.2 (b) employ two XOR gates in the critical path. The fig shows governing the existing 3-2 compressor outputs are shown below,

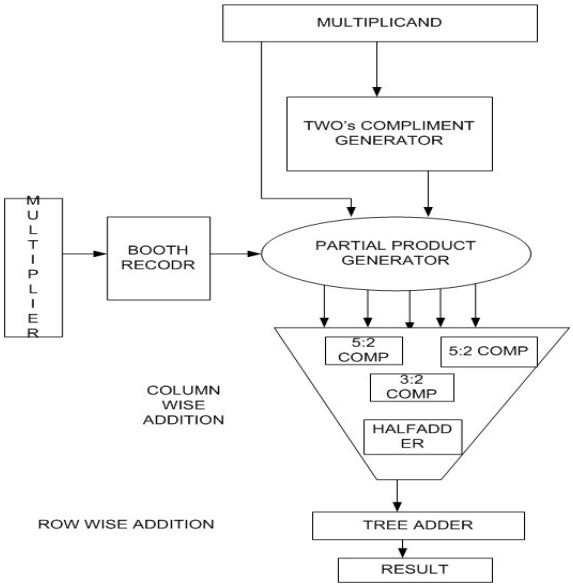

Fig 7. Flow chart

VIII. EXPERIMENTAL & RESULT

The above Wallace tree multiplier with booth recoding logic has been implemented by coding in verilog HDL and synthesis is performed by verilog. The performance is compared with booth multiplier of radix-8 and with normal wall ace multiplier. By using the proposed multiplier of 32-bit length we achieved the delay which booth recoding logic and also by compressor circuits. The following table 2 shows the comparisons of different channel is less than using normal Wallace tree multiplier. This is achieved by combined use of multipliers based on timing analysis.

topmodule Project Status			
Project File:	wallacetree_multiplier_32.xise	**Parser Errors:**	No Errors
Module Name:	topmodule	**Implementation State:**	Placed and Routed
Target Device:	xc5vlx30t-3ff323	**•Errors:**	No Errors
Product Version:	ISE 13.4	**•Warnings:**	21 Warnings (0 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	11	19,200	1%	
Number used as Latch-thrus	11			
Number of Slice LUTs	2,585	19,200	13%	
Number used as logic	2,585	19,200	13%	
Number using O6 output only	2,373			
Number using O5 output only	12			
Number using O5 and O6	200			
Number of route-thrus	12			
Number using O6 output only	12			
Number of occupied Slices	747	4,800	15%	

IX. CONCLUSION

The proposed 32x32 bit Booth encoded ± Wallace tree multiplier has been designed .and the comparison of proposed multiplier with existing Wallace tree multiplier, multiplier designed using Vedic mathematics, booth multiplier, default multiplier present in Xilinx fpga vertex-6 low power has been shown in table II. Wallace tree using 5:2, 4:2 and 3:2 compressors, radix-8 modified Booth Algorithm improve the speed of the proposed multiplier because radix-8 reduces no. of partial products, and 5:2, 4:2 and 3:2 compressor reduces no. of levels in Wallace structure. It provides less delay 9.536 ns as compared to existing Wallace tree multiplier. The results prove that the proposed architecture is more efficient than the existing one in terms of delay. This approach may be well suited for multiplication of numbers with more than 16 bit size for high speed applications. The power of the proposed multiplier can be explored to implement high performance multiplier in VLSI applications. 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.

REFERENCES

[1]. Vinoth, C.; Bhaaskaran, V.S.K.; Brindha, B.; Sakthikumaran, S.;Kavinilavu, V.; Bhaskar, B.; Kanagasabapathy, M.; and Sharath, B.;"A novel low power and high speed Wallace tree multiplier for RISC processor," *3rd International Conference on Electronics Computer Technology (ICECT), 2011*, vol.1, pp.330-334, 8-10 April 2011.

[2]. Sreehari Veeramachaneni; Kirthi M Krishna; Lingamneni Avinash;Sreekanth Reddy Puppala and M.B. Srinivas; "Novel Architectures for *International Conference on VLSI Design, 2007,* pp.324-329, Jan.2007.

[3]. Prasad, K. and Parhi, K.K.; "Low-power 4-2 and 5-2 compressors," Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, 2001, vol.1, no.,pp.129-133 vol.1, 4-7 Nov. 2001.

[4]. Chen Ping-hua and Zhao Juan; "High-speed Parallel 32×32-b Multiplier Using a Radix-16 Booth algorithm . Encoder," Third International Symposium on Intelligent Information Technology Application Workshops, 2009. IITAW '09, pp.406-409, 21-22 Nov. 2009.

[5]. Weinan Ma and Shuguo Li; "A new high compression compressor for large multiplier," Solid-State and Integrated-Circuit. Technology,2008. ICSICT 2008. 9th International Conference on , vol., no., pp.1877-1880, 20-23 Oct. 2008.

[6]. Shen-Fu Hsiao; Ming-Roun Jiang; Jia-Sien Yeh; , "Design of high speed low-power 3-2 counter and 4-2 compressor for fast multipliers," Electronics Letters , vol.34, no.4, pp.341-343, 19 Feb 1998.