IOSR Journal of Electronics and Communication Engineering (IOSRJECE)
ISSN: 2278-2834, ISBN No: 2278-8735 Volume 3, Issue 1 (Sep-Oct 2012), PP 07-11
www.iosrjournals.org

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

Jagadeshwar Rao M[1], Sanjay Dubey[2]
[1] *PG student, Centre for VLSI Design, Padmasri Dr.B.V.Raju Institute of Technology, A.P, India*
[2] *Professors, ECE Dept., Padmasri Dr.B.V.Raju Institute of Technology, A.P, India.*

Abstract: *A Wallace tree multiplier using modified booth algorithm is proposed in this paper. It is an improved version of tree based Wallace tree multiplier [1] architecture. This paper aims at additional reduction of latency and power consumption of the Wallace tree multiplier. This is accomplished by the use of booth algorithm, 5:2, 4:2, and 3:2 compressor adders. An efficient VerilogHDL code has been written, successfully simulated and synthesized for Xilinx FPGA vertex-6 low power (Xc6vlx75tl-1Lff484) device, using Xilinx 12.2 ISE and XST. The result shows that the proposed architecture is around 67% faster than the existing Wallace-tree multiplier.*
Keywords: *Arithmetic, Booth Encoder, Compressors, Radix-8, Wallace-Tree.*

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 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.

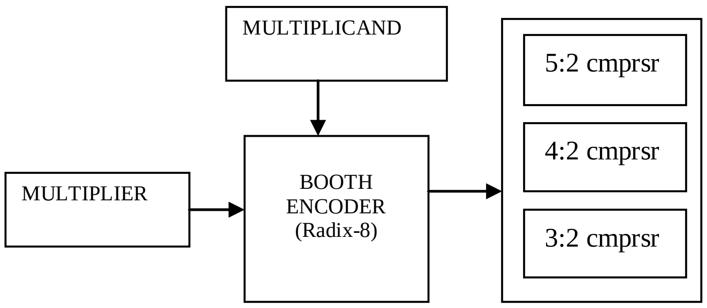


Fig 1.Proposed Architecture of Wallace tree multiplier using booth encoder

II. 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 operationed 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.

RADIX-8 BOOTH ENCODER

MULTIPLIER BITS	OPERATION FOR GROUP
0000	0
0001	+1A
0010	+2A
0011	+3A
0100	+4A
0101	+5A
0110	+6A
0111	+7A
1000	-7A
1001	-6A
1010	-5A
1011	-4A
1100	-3A
1101	-2A
1110	-1A
1111	0

Table I

III. 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 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.

A. 3:2 COMPRESSOR ARCHITECTURE

A 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 Fig.2a.

The compressor is governed by the basic equation

$x_1 + x_2 + x_3 = Sum + 2 * Carry$

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_{in}. Existing architectures shown in Fig.2 (b) employ two XOR gates in the critical path. The equations governing the existing 3-2 compressor outputs are shown below

$Sum = x_1 \oplus x_2 \oplus x_3$

$Carry = (x_1 \oplus x_2) \bullet x_3 + \overline{(x_1 \oplus x_2)} \bullet x_1$

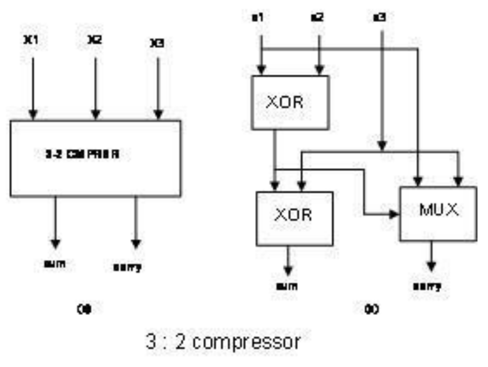


Fig.2. (a) A 3-2 Compressor
(b)Conventional Implementation of the 3-2 compressor[2]

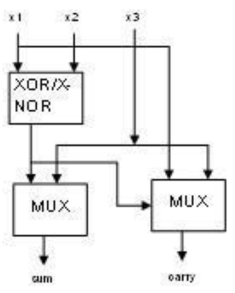
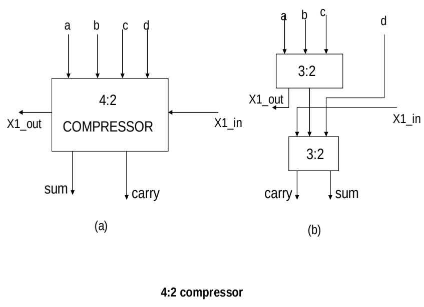


Fig 3. Modified 3:2 compressor [2]

In the architecture shown in Fig. 3, the fact that both the XOR and XNOR values are computed is efficiently used to reduce the delay by replacing the second XOR with a MUX.
The equations governing the modified 3-2 compressor outputs are shown below
$Sum = (x_1 \oplus x_2) \bullet \overline{x_3} + \overline{(x_1 \oplus x_2)} \bullet x_3$
$Carry = (x_1 \oplus x_2) \bullet x_3 + \overline{(x_1 \oplus x_2)} \bullet x_1$
It can be seen that in this implementation the overall delay is Δ-XOR + Δ-MUX (where Δ refers to delay).

B. 4-2 COMPRESSOR 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 Fig 5. 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.



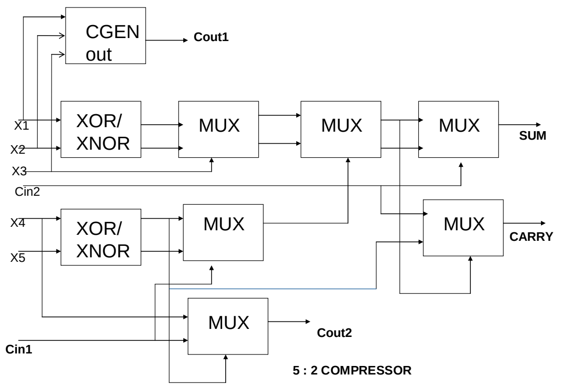


Fig 4. a) 4:2 compressor block
b) Conventional 4:2 compressor [2]

Fig 5. 4:2 Compressor Architecture [1]

Similar to the 3-2 compressor the 4-2 compressor in fig 4(a) is governed by the basic equation
$x_1+x_2+x_3+x_4+C_{in} = Sum + 2*(Carry + C_{out})$

The standard implementation of the 4-2 compressor is done using 2 Full Adder cells as shown in Fig 4(b). 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 Fig. 4(b) 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
$Sum = x_1 \oplus x_2 \oplus x_3 \oplus_4 \oplus_{in}$
$Cout = (x_1 \oplus x_2) \bullet x_3 + \underline{(x_1 \oplus x_2)} \bullet x_1$
$Carry = (x_1 \oplus x_2 \oplus x_3 \oplus x_4) \bullet c_{in} + \overline{(x_1 \oplus x_2 \oplus x_3 \oplus x_4)} \bullet x_4$

However, like in the case of 3-2 compressor, the fact that both the output and its complement are available at every stage is neglected. Thus replacing some XOR blocks with multiplexers results in a significant improvement in delay. This is shown in Fig. 5.

The equations governing the outputs in the proposed architecture are shown below

$Sum = (x_1 \oplus x_2) \bullet \overline{(x_3 \oplus x_4)} + \overline{(x_1 \oplus x_2)} \bullet (x3 \oplus x4) \bullet \overline{cin} + \overline{(x1 \oplus x2) \bullet \overline{(x_3 \oplus x_4)} + \overline{(x_1 \oplus x_2)} \bullet (x_3 \oplus x_4)} \bullet cin$
$C_{out} = (x_1 \oplus x_2) \bullet x_3 + \overline{(x_1 \oplus x_2)} \bullet x_1$
$Carry = (x_1 \oplus x_2 \oplus x_3 \oplus x_4) \bullet c_{in} + (\overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4}) \bullet x_4$
The critical path delay of the proposed implementation is Δ-XOR + 2*Δ-MUX.

C. 5-2 COMPRESSOR 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 Fig.6a. 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.

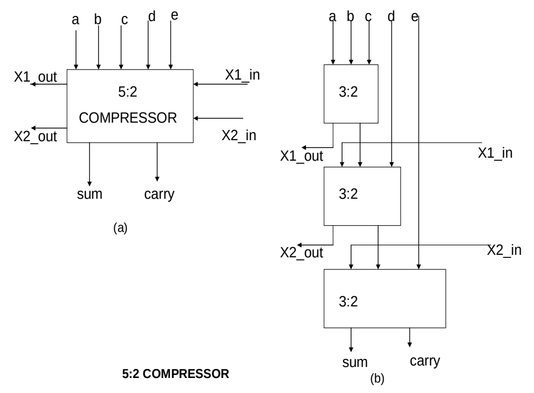


Fig.6. (a) A 5-2 compressor block
(b) Conventional implementation of a 5-2 compressor block [2]

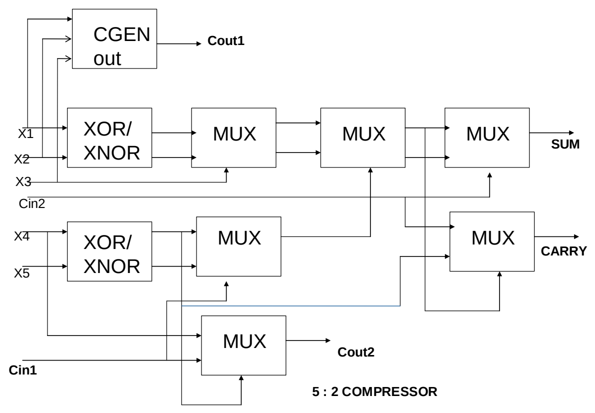


Fig.7. Architecture of the 5-2 compressor [1]

The basic equation that governs the function of the 5 -2 compressor block (fig 6) is given below
$x_1+x_2+x_3+x_4+x_5+cin_1+cin_2 = sum+2*(carry + cout_1 + cout_2)$

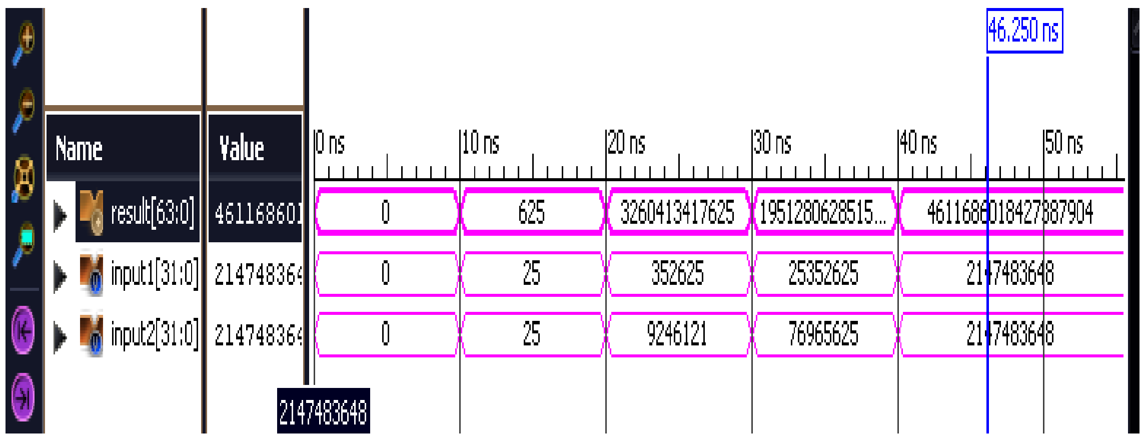
The conventional implementation of the compressor block is shown in Fig.6 (b) where 3 cascaded full adder cells are used. When these full adders are replaced with their constituent blocks of XOR gates then it can be observed that the overall delay is equal to 6*Δ-XOR for the sum or carry output. Many architectures have been proposed where the delay has been reduced to 4* Δ-XOR (Fig.7).

As mentioned before, in all the general implementations of the XOR or MUX block, in particular CMOS implementation, the output and its complement are generated. But in the existing architectures this advantage is not being utilized at all. In the proposed architecture these outputs are utilized efficiently by using multiplexers at select stages in the circuit. Also additional inverter stages are eliminated. This in turn contributes to the reduction of delay, power consumption and transistor count (area).

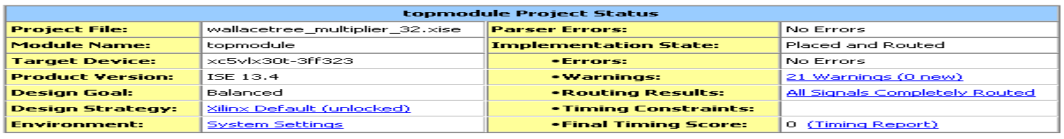
The equations governing the outputs are shown below:
$\text{Sum} = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus cin1 \oplus cin2$
$\text{Cout1} = (x_1 + x_2) \bullet x_3 + x_1 \bullet x_2$
$\text{Cout2} = (x_4 \oplus x_5) \bullet cin_1 + \overline{(x_4 \oplus x_5)} \bullet x_4$
$\text{Carry} = ((x_1 \oplus x_2 \oplus x_3) \oplus (x_4 \oplus x_5 \oplus cin_1)) \bullet cin_2 + (\overline{(x_1 \oplus x_2 \oplus x_3) \oplus (x_4 \oplus x_5 \oplus cin_1)}) \bullet (x_1 \oplus x_2 \oplus x_3)$

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 .

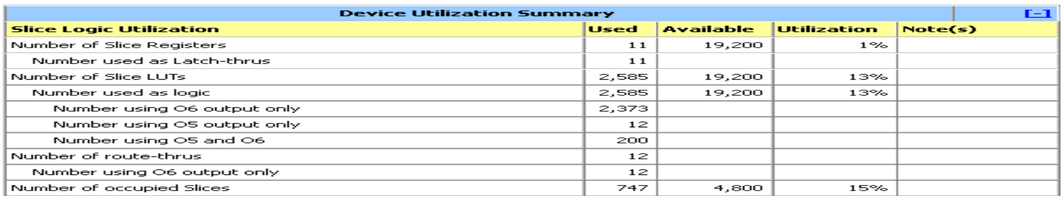
IV. Simulation Results



V. DESIGN SUMMARY

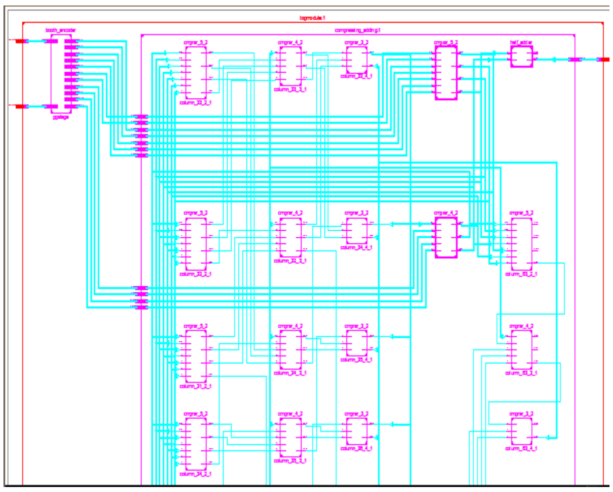


topmodule Project Status			
Project File:	wallacetree_multiplier_32.xise	**Parser Errors:**	No Errors
Module Name:	topmodule	**Implementation State:**	Placed and Routed
Target Device:	xc5vlx30t-3ff323	**•Errors:**	No Errors
Product Version:	ISE 13.4	**•Warnings:**	21 Warnings (0 new)
Design Goal:	Balanced	**•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%	

Vi. RTL SCHEMATIC



VII. COMPARISONS

TYPE OF MULTIPLIER	WIDTH	DELAY (ns)
Wallace tree Multiplier	32-bit	28.672
Multiplier using Vedic mathematics	32-bit	20.249
Modified Booth multiplier	32-bit	12.081
Fpga Xc6vlx75tl-1Lff484 Default multiplier	32-bit	11.238
Proposed Booth encoded-Wallace-tree multiplier	32-bit	9.536

Table II

VIII. 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] C.Vinoth1, V. S. Kanchana Bhaaskaran2, B. Brindha, S. Sakthikumaran, V.Kavinilavu, B.Bhaskar, M. Kanagasabapathy and B. Sharath," A Novel low power and high speed Wallace tree multiplier for risc processor",C 978-1-4244-8679-3/11/$26.00 ©2011 IEEE
[2] Sreehari veeramanchaneni, Kirthi Krishna M, Lingamneni Avinash, Sreekanth Reddy Puppala, and M.B. Srinivas," Novel Architectures for High Speed and Low power 3-2, 4-2 and 5-2 compressors"20th international conference on VLSI Design , jan 2007 , pp. 324-329.
[3] Karuna Prasad and keshab K.Parhi ,"Low Power 4-2 and 5-2 compressors" in proc. of the 35th asilomar conf. on signals, systems and computers,2001, vol. 1, pp.129-133.
[4] Chen Ping-Hua and Zhao Juan , "High speed Parallel 32x32-b Multiplier Using a Radix-16 Booth Encoder".
[5] Weinan Ma, Shuguo Li, "A New High Compression Compressor for Large Multiplier", Institute of Microelectronics,
[6] Tsinghua University, Beijing 100084,P.R. China, 2008 IEEE..
[7] S. F. Hsiao, M. R. Jiang, and J. S. Yeh, "Design of highspeed low-power 3-2 counter and 4-2 compressor for fast multipliers," Electron. Lett, vol. 34, no. 4, pp. 341–343, 1998
[8] Z. Wang, G. A. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," IEEE Trans. Comput., vol. 44, pp. 962–970, Aug. 1995.