Design and Performance Analysis of Multiplier using Wallace-Booth Algorithm

Narsampalli Bhargavi[1], Ms. Shatabdi Nandi[2], D. Devi Lavanya [3]
M.E Student(ES), Stanley College of Engineering and Technology, Hyderabad, India[1,3]
Assistant Professor, Stanley College of Engineering and Technology, Hyderabad, India[2]
bhargavi90.palli@gmail.com[1], shatabdinandi07@gmail.com[2], ddlavanya@gmail.com[3]

Abstract— This paper presents the Radix-4 Booth Algorithm with 3:2 compressors, the Radix-8 Booth algorithm with 4:2 compressors and the Radix-8 Booth algorithm with Asynchronous counters. The design is structured for m × n multiplication where m and n can reach up to 126 bits. Carry Look ahead Adder is used as the final adder to enhance the speed of operation. Finally the performance improvement of multipliers is validated by implementing a higher order FIR filter. Adder structure with counters is faster compared to Wallace tree and requires fewer gates.

Index Terms— Booth Encoding, Wallace Tree, Compressor, Carry Look Ahead Adder, Carry Save Adder, Asynchronous counters, FIR Filter.

1 INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas.

As a result, a whole spectrum of multipliers with different area-speed constraints has been designed with fully parallel multipliers at one end of the spectrum and fully serial multipliers at the other end. In between are digit serial multipliers where single digits consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. However, existing digit serial multipliers have been plagued by complicated switching systems and/or irregularities in design. Radix 2^n multipliers which operate on digits in a parallel fashion instead of bits bring the pipelining to the digit level and avoid most of 'the above problems. They were introduced by M. K. Ibrahim in 1993. These structures are iterative and modular. The pipelining done at the digit level brings the benefit of constant operation speed irrespective of the size of' the multiplier. The clock speed is only determined by the digit size which is already fixed before the design is implemented. Booth and Booth Wallace multiplier is used for implementing digital signal processing algorithms in hearing aids.

2 BOOTH ENCODING ALGORITHM

Booth algorithm is a powerful algorithm for signed number multiplication, which treats both positive and negative numbers uniformly.

2.1 Radix-4 Booth Encoding

The major disadvantage of the Radix-2 algorithm was that the process required n shifts and an average of n/2 additions for an n bit multiplier. This variable number of shift and add operations is inconvenient for designing parallel multipliers. Also the Radix-2 algorithm becomes inefficient when there are isolated 1's. The Radix-4 modified Booth algorithm overcomes all these limitations of Radix-2 algorithm. For operands equal to or greater than 16 bits, the modified Radix-4 Booth algorithm has been widely used. It is based on encoding the two's complement multiplier in order to reduce the number of partial products to be added to n/2. The multiplier, Y in two's complement form can be written as in

$$Y = -Y_{n-1}\, 2^{n-1} + \sum Y_i 2^i ; \qquad 0 \le i \le n-2$$

It can be written as

$$Y = \sum (-2\, Y_{2i+1} + Y_{2i} + Y_{2i-1})\, 2^{2i} ; \quad 0 \le i \le n-2$$

Table 1 shows the encoding of the signed multiplier Y, using the Radix-4 Booth algorithm. Radix-4 Booth recoding encodes multiplier bits into [-2, 2]. Here we consider the multiplier bits in blocks of three, such that each block overlaps the previous block by one bit. It is advantageous to begin the examination of the multiplier with the least significant bit.

Table 1: Radix-4 Multiplication

Multiplier Bits			Recoded Operation on multiplicand, X
Y_{2i+1}	Y_{2i}	Y_{2i-1}	
0	0	0	0 X
0	0	1	+1 X
0	1	0	+1 X
0	1	1	+2 X
1	0	0	-2 X
1	0	1	-1 X
1	1	0	-1 X
1	1	1	0 X

The overlap is necessary so that we know what happened in the last block, as the most significant bit of the block acts like a sign bit.

2.2 Radix-8 Booth Encoding

Radix-8 Booth recoding applies the same algorithm as that of Radix-4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signed digit using Table 2. Radix-8 algorithm reduces the number of partial products to n/3, where n is the number of multiplier bits. Thus it allows a time gain in the partial products summation.

Table 2: Radix-8 Multiplication

Multiplier Bits				Recoded Operation on multiplicand, X
Y_{i+2}	Y_{i+1}	Y_i	Y_{i-1}	
0	0	0	0	0 X
0	0	0	1	+1 X
0	0	1	0	+1 X
0	0	1	1	+2 X
0	1	0	0	+2 X
0	1	0	1	+3 X
0	1	1	0	+3 X
0	1	1	1	+4 X
1	0	0	0	-4 X
1	0	0	1	-3 X
1	0	1	0	-3 X
1	0	1	1	-2 X
1	1	0	0	-2 X
1	1	0	1	-1 X
1	1	1	0	-1 X
1	1	1	1	0 X

3 WALLACE TREE

The Wallace tree method is used in high speed designs in order to produce two rows of partial products that can be added in the last stage. Also critical path and the number of adders get reduced when compared to the conventional parallel adders. Here the Wallace tree has taken the role of accelerating the accumulation of the partial products. Its advantage becomes more pronounced for multipliers of greater than 16 bits .The speed, area and power consumption of the multipliers will be in direct proportion to the efficiency of the compressors. The 3:2 compressors and 4:2 compressors are shown in Fig.1 and Fig.2 respectively. In this regard, we can expect a significant reduction in computing multiplications.

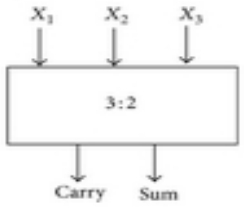

Figure 1: Compressor 3:2

The 3:2 compressors make use of a carry save adder .The carry save adder outputs two numbers of the same dimensions as the inputs, one is a sequence of partial sum bits and other is a sequence of carry bits. In carry save adder, the carry digit is taken from the right and passed to the left, just as in conventional addition; but the carry digit passed to the left is the result of the previous calculation and not the current one. So in each clock cycle, carries only have to move one step along and the clock can tick much faster. Also the carry-save adder produces all of its output values in parallel, and thus has the same delay as a single full-adder. The 4:2 compressors have been widely employed in the high speed multipliers to lower the latency of the partial product accumulation stage. A 4:2compressor can be built using two 3:2 compressors. Owing to its regular interconnection, the 4:2 compressors is ideal for the construction of regularly structured Wallace Tree with low complexity.

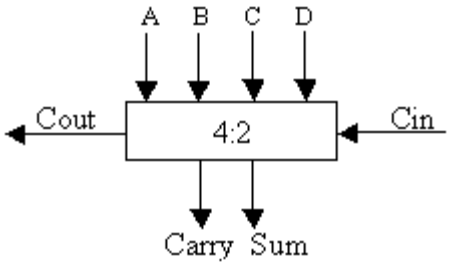

Figure 2: Compressor 4:2

The final results obtained at the output of the Wallace tree are added using a Carry Look-ahead Adder (CLA) which is independent of the number of bits of the two operands.

In Carry Look-ahead Adder, for every bit the carry and sum outputs are independent of the previous bits and thus

the rippling effect has completely been eliminated. It works by creating two signals, propagate and generate for each bit position, based on whether a carry is propagated through from a less significant bit position, a carry is generated in that bit position, or if a carry is killed in that bit position.

4 ARCHITECTURE

This section deals with architecture of the proposed methods.

4.1 Radix-4 using 3:2 Compressors

Booth Encoder encodes the multiplier using Radix-4 algorithm. The encoded output is given to the partial product generator which produces partial products and number of partial products generated are (n/2+1).These partial products are passed to Wallace Tree (with 3:2 compressors).Wallace tree reduces (n/2+1) partial products to 2 partial products .Final stage used is Carry Look Ahead Adder which adds the sum and carry (2 partial products).

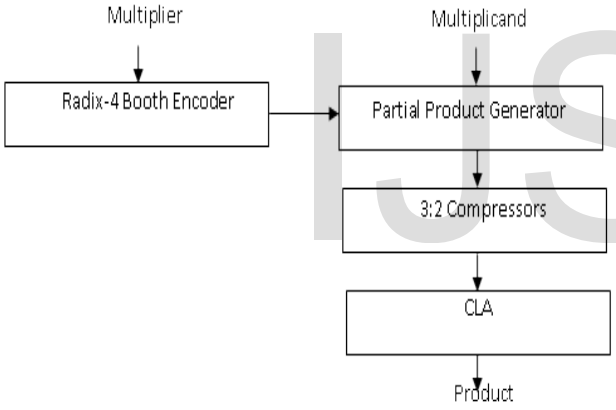

Figure 3: Radix-4 Booth Algorithm with 3:2 Compressors

4.2 Radix-8 using 4:2 Compressors

Booth Encoder encodes the multiplier using Radix-8 algorithm. The encoded output is given to the partial product generator which produces partial products and number of partial products generated is (n/3+1). These partial products are passed to Wallace Tree (with 4:2 compressors).Wallace tree reduces (n/3+1) partial products to 2 partial products. Final stage used is Carry Look Ahead Adder which adds the sum and carries (2 partial products).

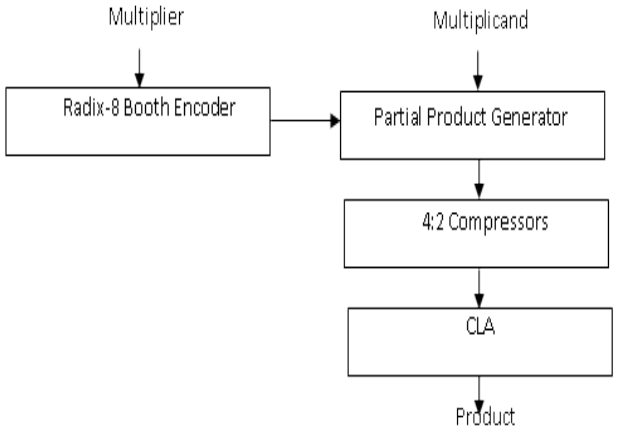

Figure 4: Radix-8 Booth Algorithm with 4:2 Compressors

4.3 Radix-8 using Asynchronous Counters

In Radix-8 multiplier 4:2 compressors is replaced by asynchronous counters. The counters are used to count the number of 1's in a column. Each of them is a simple DFF based ripple counter. The clock input is synchronized with the input data rate and thus the operands can be accumulated with a high frequency defined by the setup time and propagation delay of a DFF. Moreover, the counters change states only when the input is "1," which leads to low switching power. This simple and efficient bit accumulation technique is used to design this multiplier. Advantages of this method are low complexity and speed increase.

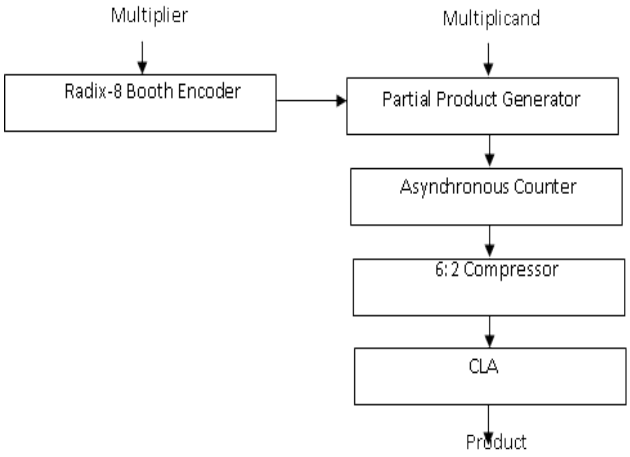

Figure 5: Radix-8 with Asynchronous Counters

5 IMPLEMENTATION RESULTS

The design entry of 126×126 bit multipliers using Radix-4 Booth algorithm with 3:2 compressors, Radix-8 Booth algorithm with 4:2 compressors and Radix-8 Booth algorithm with Asynchronous counters are done using VHDL and simulated using ModelSim SE 6.4 design suite from Mentor Graphics. It is then synthesized and implemented in a Xilinx XC3S5000 fg1156 -4 FPGA using the Xilinx ISE 9.2i design suite. Figure 6 presents a snapshot of simulation waveforms for 126×126 bit multiplier. Table 3 summarizes the FPGA resource utilization of these three multipliers.

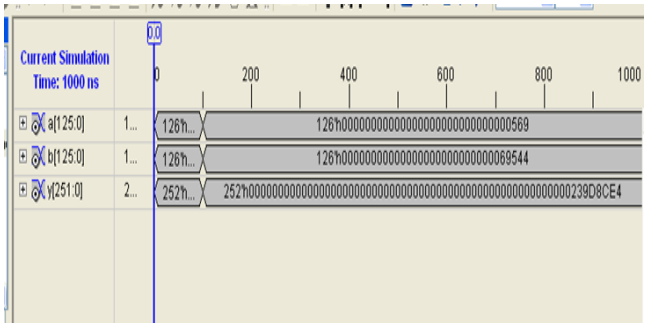

Figure 6: Simulation of 126 × 126 bit multiplier

Table 3: Device Utilization Summary of Multipliers

Logic Utilization (XC3S5000 fg1156 -4)	Radix-4 using 3:2 Compressors	Radix-8 using 4:2 Compressors	Radix-8 using Asynchronous Counters
No. of four i/p LUTs	29,990	43,520	34981
No. of Slices	16,535	25,100	69770
No. of IOBs	503	503	506
Maximum combinational path delay(ns)	448.270	448.082	No Combinational delay
Average pin delay	5.514	4.781	4.531
Maximum pin delay	27.603	28.171	28.051

Finally the performance improvement of first two methods is validated by implementing a higher order FIR filter using these multipliers. Table 4 summarizes the FPGA resource utilization for FIR filters using these multipliers. This shows that the multiplier using Radix-8 Booth multiplier with Asynchronous counters gives better speed and the number of occupied slices is lower for the multiplier using Radix-4 Booth algorithm with 3:2 compressors. The FIR filters are implemented in Xilinx XC3S1500fg676-4 FPGA.

Table 4: Device Utilization Summary of FIR Filters

Logic Utilization (XC3S5000fg1156 -4)	Radix-4 using 3:2 Compressors	Radix-8 using 4:2 Compressors
Number of four input LUTs	9,870	11,099
Number of occupied Slices	5,445	5,910
Number of bonded IOBs	311	311
Maximum combinational path delay (ns)	100.241	75.502

6 CONCLUSION

In this paper, the design and implementation of three high performance parallel multipliers is proposed. The first multiplier makes use of the Radix-4 Booth Algorithm with 3:2 compressors, the second multiplier uses the Radix-8 Booth algorithm with 4:2 compressors, and the third multiplier uses the Radix-8 Booth algorithm with asynchronous counters. All the designs were implemented on Spartan 3 FPGA. The multiplier using Radix-8 Booth algorithm with 4:2 compressors shows more reduction in device utilization as compared to the multiplier using Radix-4 Booth algorithm with 3:2 compressors. Meanwhile the multiplier using Radix-8 Booth algorithm with 4:2 compressors are found to be faster than the other. The first two multipliers have been implemented in FIR filters and found that the FIR filter with radix 8 booth encoding and 4:2 compressors is fastest.

7 REFERENCES

[1] C. S. Wallace, "A Suggestion for a Fast Multiplier", Electronic Computers, IEEE Transactions, vol.13, Page(s): 14-17, Feb. 1964.

[2] Dong-Wook Kim, Young-Ho Seo, "A New VLSI Architecture of Parallel Multiplier-Accumulator based on Radix-2 Modified Booth Algorithm", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol.18, pp.: 201-208, 04 Feb. 2010.

[3] Hussin R et al , "An Efficient Modified Booth Multiplier Architecture", IEEE International Conference, pp.:1-4,2008.

[4] Lakshmanan, Masuri Othman and Mohamad Alauddin Mohd.Ali, "High Performance Parallel Multiplier using Wallace-Booth Algorithm", Semiconductor Electronics, IEEE International Conference , pp.: 433-436, Dec. 2002.

[5] Louis P. Rubinfield, "A Proof of the Modified Booth's Algorithm for Multiplication", Computers, IEEE Transactions,vol.24, pp.: 1014-1015, Oct. 1975.

[6] Nilay Nagdeve, "A Simulation Based Evaluation of Different Compressors For Fast Multiplication" International Journal of Scientific & Engineering Research Volume 3, Issue 6, June-2012.

[7] Prasanna Raj P, Rao, Ravi, "VLSI Design and Analysis of Multipliers for Low Power", Intelligent Information Hiding and Multimedia Signal Processing, Fifth International Conference, pp.: 1354-1357, Sept. 2009.

[8] Rajendra Katti, "A Modified Booth Algorithm for High Radix Fixedpoint Multiplication", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol. 2, pp.: 522-524, Dec. 1994.