S.Nagaraj, R.Mallikarjuna Reddy / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 2, March -April 2013, pp.295-299

FPGA Implementation of Modified Booth Multiplier

S.Nagaraj[1], R.Mallikarjuna Reddy[2]
[1]Associate professor, Department of ECE, SVCET, Chittoor, nagarajsubramanyam@gmail.com
[2]Associate professor, Department of ECE, SVCET, Chittoor. mallikarjunareddy.r416@gmail.com

Abstract

To design a high speed multiplier with reduced error compensation technique. The fixed-width multiplier is attractive to many multimedia and digital signal processing systems which are desirable to maintain a fixed format and allow a little accuracy loss to output data. This paper presents the Design of error compensated truncation circuit and its implementation in fixed width multiplier. To reduce the truncation error, we first slightly modify the partial product matrix of Booth multiplication and then derive an effective error compensation function that makes the error distribution be more symmetric to and centralized in the error equal to zero, leading the fixed-width modified Booth multiplier to very small mean and mean-square errors. However, a huge truncation error will be introduced to this kind of fixed-width modified Booth multipliers. To overcome this problem, several error compensated truncation circuit approaches have been proposed to effectively reduce the truncation error of fixed-width modified Booth multipliers.

I.INTRODUCTION

HIGH processing performance and low power dissipation are the most important objectives in many multimedia and digital signal processing (DSP) systems, where multipliers are always the fundamental arithmetic unit and significantly influence the system's performance and power dissipation. To achieve high performance, the modified Booth encoding which reduces the number of partial products by a factor of two through performing the multiplier recoding has been widely adopted in parallel multipliers. Moreover, nxn fixed-width multipliers that generate only the most significant product bits are frequently utilized to maintain a fixed word size in these loss systems which allow a little accuracy loss to output data. Significant hardware complexity reduction and power saving can be achieved by directly removing the adder cells of standard multiplier for the computation of the least significant bits of 2n-bit output product. However, a huge truncation error will be introduced to this kind of direct-truncated fixed-width multiplier (DTFM). To effectively reduce the truncation error, various error compensation methods, which add estimated compensation value to the carry inputs of the reserved adder cells, have been proposed. The error compensation value can be produced by the constant Scheme. The constant scheme through adaptively adjusting the compensation value according to the input data at the expense of a little higher hardware complexity. However, most of the adaptive error compensation approaches are developed only for fixed-width array multipliers and cannot be applied to significantly reduce the truncation error of fixed-width modified Booth multipliers directly. To overcome this problem, several error compensation approaches [1]–[3] have been proposed to effectively reduce the truncation error of fixed-width modified Booth multipliers. In [1], the compensation value was generated by using statistical analysis and linear regression analysis. This approach can significantly decrease the mean error of fixed-width modified Booth multipliers, but the maximum absolute error and the mean-square error are still large. Cho *et al.* [2] divided the truncated part of the bit product matrix of Booth multiplication into a major group and a minor group depending on their effects on the truncation error. To obtain better error performance with a simple error compensation circuit, Booth encoded outputs are utilized to generate the error compensation value. In [3], a systematic design methodology for the low-error fixed-width modified Booth multiplier via exploring the influence of various indices in a binary threshold was developed to decrease the product error. The fixed-width modified Booth multipliers in [2] and [3] achieve better error performance in terms of the maximum absolute error and the mean-square error when compared with the previous published multiplier in [1]. However, their mean errors are much larger than that of [1]. The smaller mean error and mean-square error represent that the error distribution is more symmetric to and centralized in the error equal to zero (denoted as *zero error*). For many multimedia and DSP applications, the final output data are produced from accumulating a series of products rather than from a single multiplication operation directly.

This paper is organized as follows. In section II, the modified booth multiplier is briefly reviewed. The implementation results and outputs are showed. Section III describes the detailed comparison of booth multiplier and modified booth multiplier. Finally ,section IV concludes this paper.

II. PROPOSED LOGIC

Here booth multiplier is going to modified as Multiplier, partial product, partial product shifter, adder blocks are shown in below figure 1

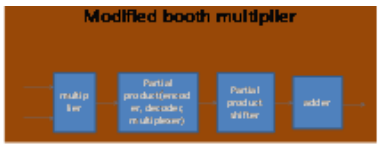


Fig 2.1 Block diagram of modified booth multiplier

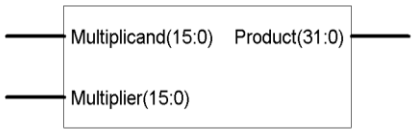


Fig 2.1.1.Block diagram of multiplier

For example:
Multiplicand: 0110010110101001
Multiplier : 0000111101010101
Product : 00000110000101101010100000001101

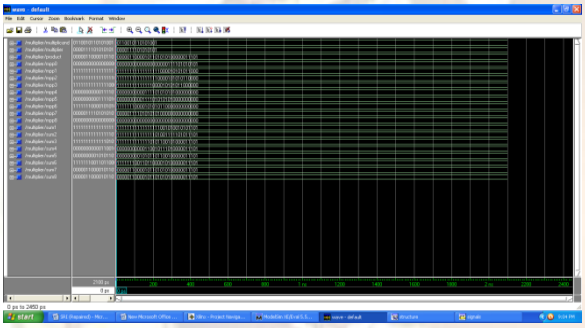


Fig2.1.2.Output waveform of multiplier

2.1 MODIFIED BOOTH ENCODER (MBE)

Modified Booth encoding is most often used to avoid variable size partial product arrays. Before designing a MBE, the multiplier B has to be converted into a Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier. The figure above shows that the multiplier has been divided into four partitions and hence that mean four partial products will be generated using booth multiplier approach Instead of eight partial products being generated using conventional multiplier.

Zn = -2* Bn+1 + Bn + Bn-1

Let's take an example of converting an 8-bit number into a Radix-4 number. Let the number be **-36 = 11011100**. Now we have to append a '**0**' to the LSB. Hence the new number becomes a 9-digit number that is **110111000**. This is now further encoded into Radix-4 numbers according to the following given table.

Bits A	Bits B	Bits C	operation	X1	X2	Add	Sub
0	0	0	+0	0	0	0	1
0	0	1	+a	0	1	0	1
0	1	0	+a	0	1	0	1
0	1	1	+2a	1	0	0	1
1	0	0	-2a	1	0	1	0
1	0	1	-a	0	1	1	0
1	1	0	-a	0	1	1	0
1	1	1	-0	0	0	1	0

Table 1: Modified booth encoder

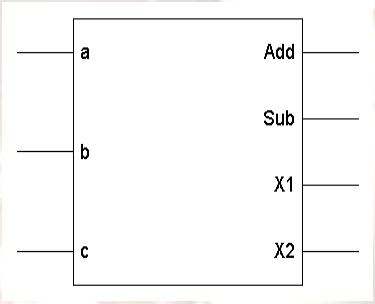


Fig 2.1.3.Block diagram of booth encoder

The encoder block generates the selector signals for each 3 bits of multiplicand. This is the logic for the encoder block:

X1= (a xor b)(a xor c)(not(b xor c))

X2= b xor c

Add=not a

Sub= a

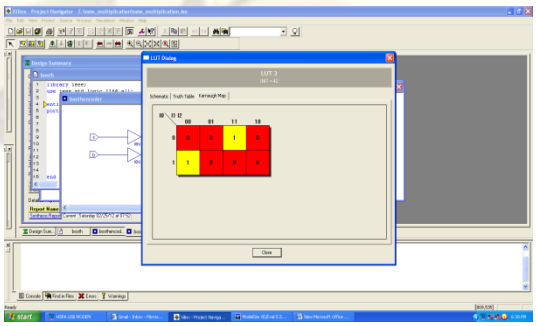


Fig 2.1.4.k-map of booth encoder

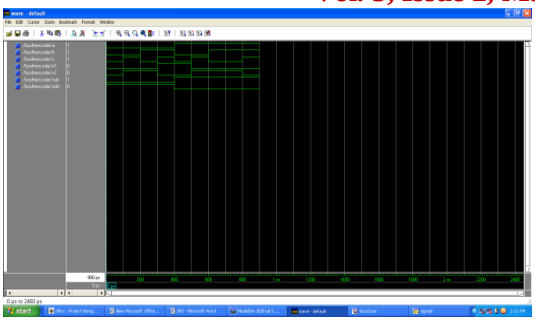


Fig 2.1.5.Booth encoder output Wave form

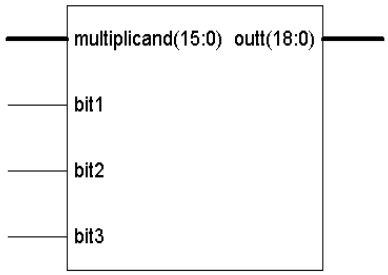


Fig 2.1.6.Block diagram of booth decoder

The decoder block generates the partial product from the selector signals that they are generated in encoder block.

Example:

Multiplicand = 0110010110101001

Bits = 0110

Out = 1111001101001010

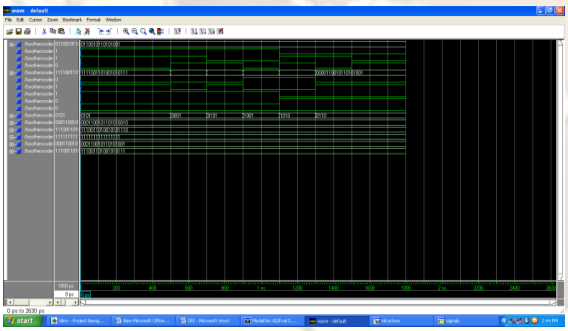


Fig 2.1.7.Booth decoder output waveform

2.2 PARTIAL PRODUCT

Partial product generator is the combination circuit of the product generator and the 5 to 1 MUX circuit. Product generator is designed to produce the product by multiplying the multiplicand A by 0, 1, -1, 2 or -2. A 5 to 1 MUX is designed to determine which product is chosen depending on the M, 2M, 3M control signal which is generated from the MBE. For product generator, multiply by zero means the multiplicand is multiplied by "0".Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place.

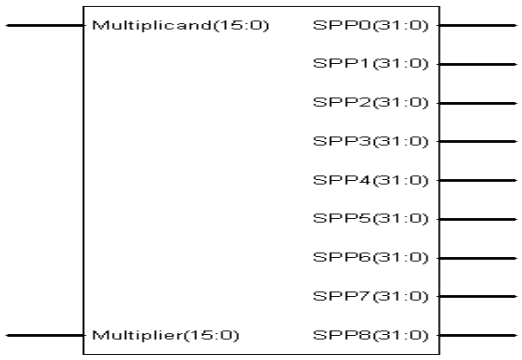


Fig 2.2.1.Block diagram of partial product

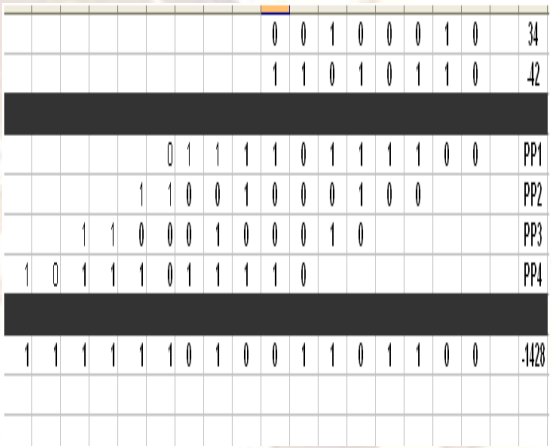


Fig 2.2.2.Example of showing partial product (6-bit)

method showing how partial products should be Added

To prove the output result is correct:

11111101001101100 =

20(0) + 21(0) + 22(1) +

23(1) + 24(0) + 25(1) +

26(1) + 27(0) + 29(1) +

210(0) + 211(-1) =

4+8+32+64+512-2048= -1428

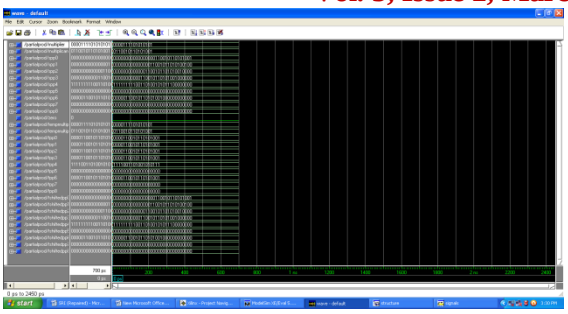


Fig 2.2.3.Partial product output waveform

2.3 PARTIAL PRODUCT SHIFTER

Partial product shifter is used to know when numbers of bits are shifted after every operation of multiplier.

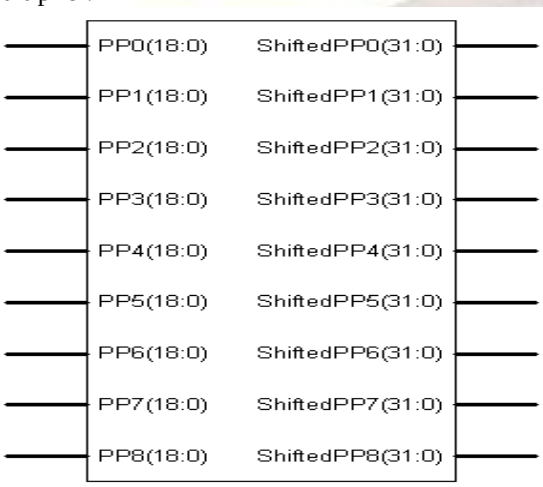


Fig 2.3.1.Block diagram of partial product shifter

For example:
Multiplier - 0000111101010101
Multiplicand - 0110010110101001
pp0 - 00000000000000000110010110101001
pp1 - 00000000000000001100101101010100
pp2 - 00000000001100101101010010010000

Like this bits are shifted for every operation of multiplier.

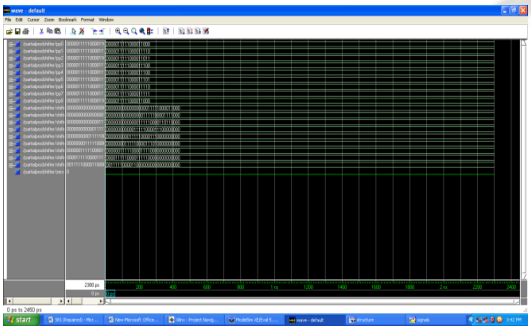
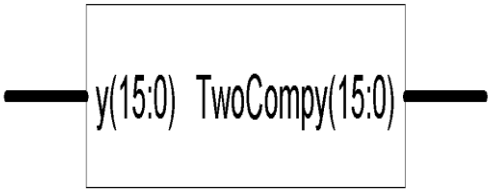


Fig 2.3.2.Partial product shifter output waveform

Two's complement

Here two's complement is implemented in new using xor & or gates.



.

Fig 2.3.3.Block diagram of two's complement

For example:

X: 0000111101010101
Y: 1111000010101010
1
Z: 1111000010101011

Or vector is used to put zeros or ones:

1. If MSB of the two's complement result is one then or vector is one.
2. If MSB of the two's complement result is zero then or vector is zero.

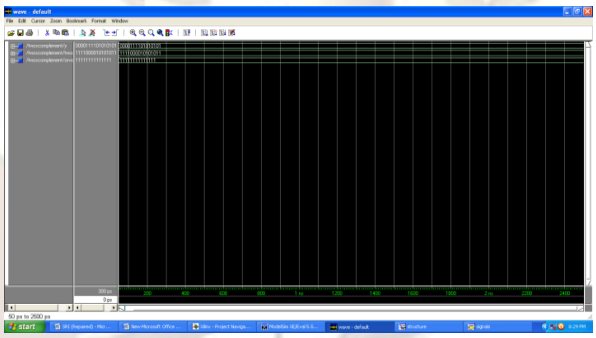


Fig 2.3.4.Two's complement output waveform

2.4 ADDER

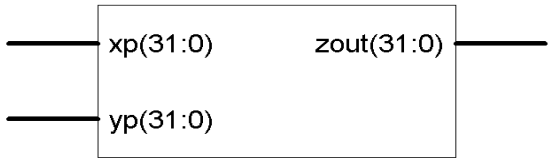


Fig 2.4.1.Block diagram of adder

Adder takes the inputs performs addition operation and generates sum, carry outputs
For example:
X: 00001111000011110101010101010101
Y : 00001111000011110101010101010100
Z:00001111000011110101010101010001

Fig 2.4.2.Adder Output waveform

III. PARAMETER COMPARISONS

After synthesis	Booth multiplier	Modified booth multiplier
Adders-16	16	-
Subtract-16	15	-
4x1 mux	240	-
No. of slices	500	366
4 i/p LUT	977	644
IOBS	64	64
Combinational delay path	86.70ns	65.96ns
<u>After map</u>		
No. of occupied slices	496	375
4-i/p LUT	992	642
Equivalent gates	9,456	3939
No. of fan-out	6	24
<u>Place &route</u>	64	32
external IOB	496	357
No. of slices		
Power consumed	25Mw	7mW

Table 2: Parameters comparison

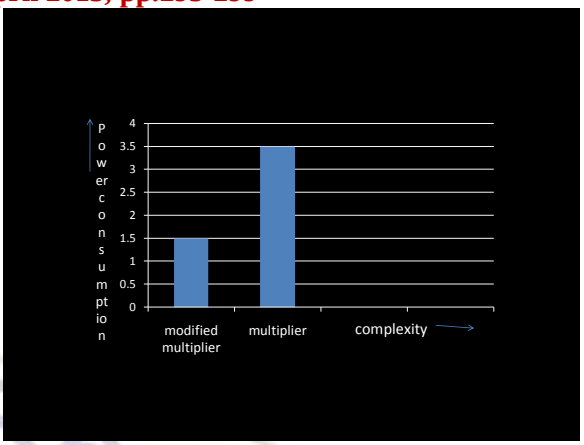


Fig 2 Graph representation of modified multiplier and multiplier

In this graph vertical axis is power consumption, horizontal axis is complexity. We know from this graph complexity and power consumption is less in modified booth multiplier, when compared to multiplier. So, modified multiplier is used to save power, complexity is reduced, speed increment can be performed.

IV. CONCLUSION

In this paper, FPGA implementation of modified Booth multiplier has been proposed. In the proposed multiplier, the Partial product matrix of Booth multiplication was slightly modified as booth encoder, decoder, and mux. In booth encoder, encoding table is derived from the booth multiplier, according to this table we perform shifting, two's complement in new way. So, modified multiplier is used to save power, complexity is reduced, speed increment can be achieved. When booth multiplier and modified booth multiplier we can save the power up to 40% respectively.

REFERENCES

[1] S. J. Jou, M.-H. Tsai and Y.-L. Tsao, "Low-error reduced-width BoothMultipliers for DSP applications," *IEEE Trans. Circuits Syst. I, Fudam.Theory Appl.*, vol. 50, no. 11, pp. 1470–1474, Nov. 2003.

[2] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low errorFixed-width modified Booth multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 522–531, May 2004.

[3] M.-A. Song, L.-D. Van and S.-Y. Kuo, "Adaptive low-error fixed widthBooth multipliers," *IEICE Trans. Fundamentals*, vol. E90-A, no.6, pp. 1180–1187, Jun. 2007.