

Fused Add Then Multiply Implementation using Modified Booth Encoder

Chowtapalli Dayakar¹

P.Ranjith Kumar²

¹*PG Student (M.Tech), Dept. of ECE, Universal Clg. of Eng. and Tech., Guntur, AP, India*

²*Assistant professor, Dept. of ECE, Universal Clg. of Eng. and Tech., Guntur, AP, India*

Abstract— Booth recoding is widely used to reduce the number of partial products in multipliers. Different recordings exist resulting in different gate level implementations and performance. In this work the XOR-based implementation gives lowest area and delay numbers in most technologies due to the small selector size and the well-balanced signal paths. An implementation of a radix-4 butterfly has been developed. The number of stages has been reduced. This reduction comes from the fact that, to achieve a throughput comparable to that of radix-2. Therefore, the implementation of the radix-4 butterfly is suitable for high speed applications, since the hardware cost, the power consumption and the latency are reduced. To reduce the number of calculation steps for the partial products, MBA algorithm has been applied mostly where Wallace tree has taken the role of increasing the speed to add the partial product.

Keywords—Booth Encoder, Multiply, Add, Fused.

I. INTRODUCTION

Most digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) or discrete wavelet transform (DWT). Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic's determines the execution speed proceedings and performance of the entire calculation. Because the multiplier requires the longest delay among the basic operational blocks in digital system, the critical path is determined by the multiplier, in general. For high-speed multiplication, the modified radix-4 Booth's algorithm (MBA) is commonly used.

Digital signal processing (DSP) is widely used in many applications.. To overcome this problem Modified booth algorithm can be used to reduce critical delay, power consumption. To increase a performance Fused add multiply operator can be used to obtain a high throughput the logic operations involved in conventional carry select adder (CSLA) and binary to excess-1 converter (BEC)-based CSLA are analyzed to study the data dependence and to identify redundant logic operations. We have eliminated all the redundant logic operations present in the conventional CSLA and proposed a new logic formulation for CSLA. In the proposed scheme, the carry select (CS) operation is scheduled before the calculation of final-sum, which is different from the conventional approach. Bit patterns of two anticipating carry words (corresponding to $c_{in} = 0$ and 1) and fixed c_{in} bits are used for logic optimization of

CS and generation units. An efficient CSLA design is obtained using optimized logic units

Digital Signal Processing is extensively used in the domains of multimedia, signal processing, etc. Since most of the DSP applications are based on intensive kernels, a large number of arithmetic operations are carried out. Depending on the allocation and architecture of these arithmetic units, the performance of the DSP systems varies. By sharing the common data among different arithmetic operations, significant performance improvement can be achieved. This improvement can be observed from the implementation of Divide-Add Fused operation and fused floating point operations.

In most of the DSP applications, an addition operation is often successive to multiplication operation. To perform multiply-add or add-multiply or multiply-add-multiply operations, instead of using separate blocks, if a single dedicated unit is used, better performance can be achieved. This can be observed from the Multiply-Accumulator (MAC) and Multiply-Add (MAD) unit designs. Other than the MAC/MAD operations, many DSP applications depend on Add-Multiply (AM) operations. The direct design of the AM unit requires that the output of the adder to be first driven to the input of the multiplier.

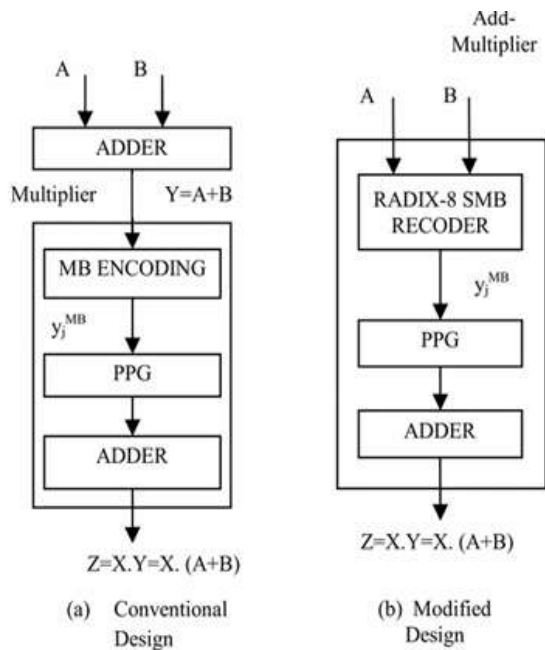


Figure 1: Add multiply unit (a) conventional design (b) modified design

The use of separate adder and multiplier increases both the critical path delay and area. To minimize the carry propagation delay, Carry Look Ahead (CLA) adder or some other efficient adder can be used. But this increases the area. So to improve the performance of the AM unit, designs that share data were implemented based on the fusion technique and Carry free addition. The fusion technique is employed based on the direct recoding of the addition of two numbers to its Modified Booth (MB) form (equivalent to Carry Save form). The use of constant-time addition ensures that the execution delay is independent of the input bit widths. The authors Lyu and Matula introduced a novel signed-bit recoder that transforms the redundant numbers to MB recoding form.

A two-stage recoder has been proposed that converts a carry save form number to its MB form. In this, the first stage converts an input number in the carry save form to signed digit form, and then in the second stage, recoding is done to match with the form that MB digits request. Zimmerman and Tran presented an improved design of, which yields optimized design in terms of both critical path and area. The authors Daumas and Matula, had proposed a recoder that transforms input in carry save form to the respective borrow-save form with the critical path fixed.

II. BOOTH MULTIPLIER

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ± 1 , ± 2 , or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Figure 2.3.1 shows the grouping of bits from the multiplier term for use in modified booth encoding.

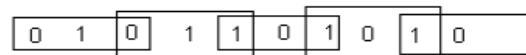


Figure 2: grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 2.3.2

Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

Table 2.3.2 Operation on the Multiplicand

For the partial product generation, we adopt Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half. For multiplication of 2’s complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits.

Figure 4, shows a computing example of Booth multiplying two numbers "2AC9" and "006A". The shadow denotes that the numbers in this part of Booth multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals.

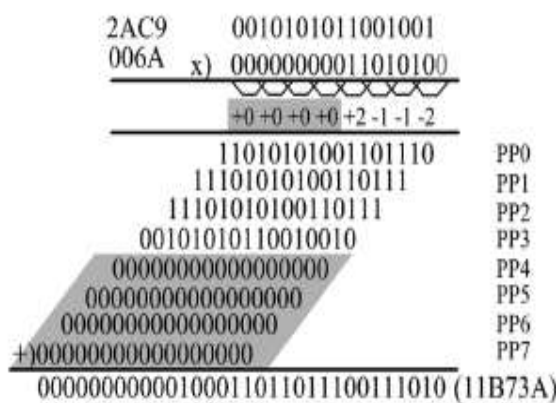


Figure 3: Illustration of multiplication using modified Booth encoding

The PP generator generates five candidates of the partial products, i.e., $\{-2A, -A, 0, A, 2A\}$. These are then selected according to the Booth encoding results of the operand B. When the operand besides the Booth encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree.

III. PROPOSED SYSTEM

Number of promising technologies shows an enormous advancement of multiplier over the past few decades. The array multiplier was an earliest reported multiplier that employs a series of ripple carry adders to compute the product by repetitive addition. It has regular structure but the speed of this multiplier is relatively slow. The shortcomings of array multiplier are resolved by Wallace tree multiplier. The Wallace tree construction method is used to accelerate the multiplication by compressing the number of partial products in a tree-like fashion and produce two rows of partial products that can be added by utilizing the suitable adder in the last stage. Generally, Wallace tree multiplier is used to reduce the time complexity and the depth of the adder chain. In high speed multipliers, 4:2 compressors are used extensively to curb the time taken at the partial product accumulation stage. By virtue of its regular interconnection, 4:2 compressor is used to construct regularly structured Wallace tree multiplier with reduced complexity.

In the S-MB2 recoding mechanism, the sum of two continuous bits of two inputs $A(a_{2j}, a_{2j+1})$ and $B(b_{2j}, b_{2j+1})$ are recoded into single MB digit Y_j^{MB} . In general, three bits are comprehended in forming a MB digit. The most significant bit of them has negative weight but the two least significant bits are positively weighted and signed-bit arithmetic is used to transform the above pairs of bits into MB form. Bit-level signed Half Adders (HA) and signed Full Adders (FA) was used for this purpose. Two types of signed HAs such as HA^* and HA^{**} are used. The Boolean equation for half adder HA^* is given by $c = p \vee q, s = p \oplus q$, where p and q are the binary inputs and c, s are the carry and sum outputs respectively. The below figure symbolizes the schematic of HA^{**} . Two types of signed FAs such as FA^* and FA^{**} are used as a building block in the S-MB recoders. Boolean equations and schematics for signed FA^* and FA^{**} are given in below figures respectively. Here p and q are the inputs and C_i, S are the output carry and sum respectively. FA^* implements the relation $2.c_0 - s = p - q + c_i$ where the bits s and C_i are negatively signed. In FA^{**} , the two inputs p and q are negatively signed and FA^{**} implements the relation $-2.c_0 + s = -p - q + c_i$.

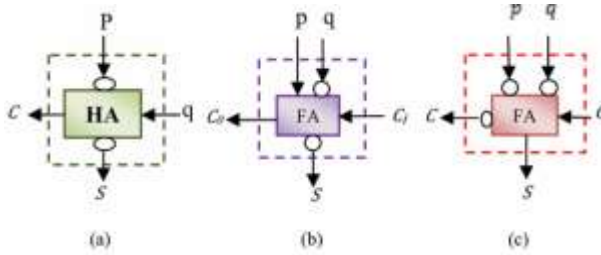


Figure 4: Schematic for signed (a) ha*, (b) fa* and (c) fa**

The S-MB2 recoding technique is illustrated for even number of bits and for odd number of bits. Here the traditional FA is used initially. a_{2j}, b_{2j} and c_{2j} are the inputs of the FA and it produces the carry c_{2j+1} and the sum S_{2j} . The bit $c_{2j,1}$ is the output carry of the conventional HA with the bits a_{2j-1}, b_{2j-1} as inputs. The sum bit S_{2j+1} of HA* is produced by driving the bits c_{2j+1} and the sum output bit generated by the conventional HA with a_{2j+1}, b_{2j+1} as inputs. The HA* produce the negatively signed sum S_{2j+1} and its outputs are given by the Eq.1:

$$\begin{aligned} c_{2j+2,2} &= c_{2j+1,1} \vee (a_{2j+1} \oplus b_{2j+1}) \\ s_{2j+1} &= a_{2j+1} \oplus b_{2j+1} \oplus c_{2j+1} \end{aligned} \quad (1)$$

In this paper, we design a circuit of AM unit which implement the operation $Z=X(A+B)$. The conventional design of the AM operator requires that its inputs A and B are fed to an adder and then the input X and the sum $Y=A+B$ is fed to a multiplier to get the final result Z. The drawback of this method is the delay is high. To reduce the delay we use Carry-Look-Ahead adder but this increases the area of the design and thereby increasing the power consumption. By using the direct recoding of sum to modified booth form we can reduce the delay and power consumption.

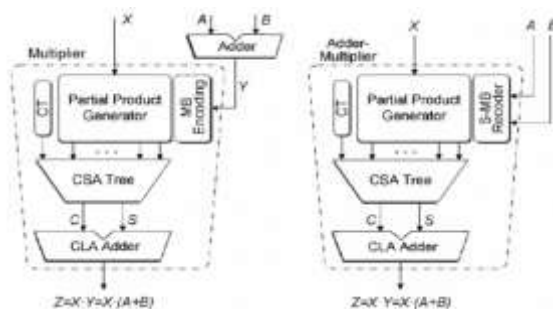


Figure 5: Add-multiply operator based on the (a) conventional design and (b) fused design using direct sum to modified booth recoding

Binary			Y_j^{sm}	MB Encoding			Input carry
Y_{2j+1}	Y_{2j}	Y_{2j-1}		Sign= S_j	X1=One $_j$	X2=Two $_j$	
0	0	0	0	0	0	0	0
0	0	1	+1	0	1	0	0
0	1	0	+1	0	1	0	0
0	1	1	+2	0	0	1	0
1	0	0	-2	1	0	1	1
1	0	1	-1	1	1	0	1
1	1	0	-1	1	1	0	1
1	1	1	0	1	0	0	0

Table I. Modified booth encoding table

IV. RESULTS & DISCUSSIONS

Experimental results of the proposed system were shown.

Top module



Figure 6: top module resultant

The above simulation has a and b as the input for summation and x as the multiplication factor. so FAM result is shown as sum in the simulation result. a=2 and b=2 and the final result is 12

S mbrecode

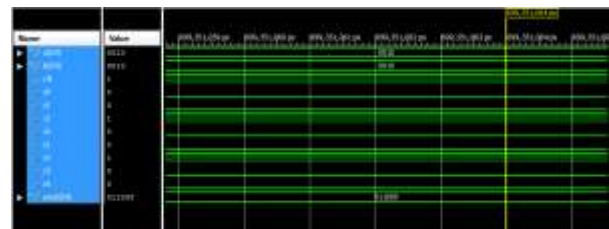


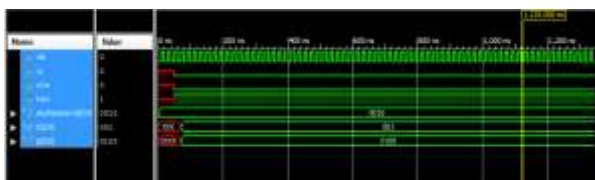
Figure 7: adder resultant

The above simulation window shows the full adders result in the s mbrecode.

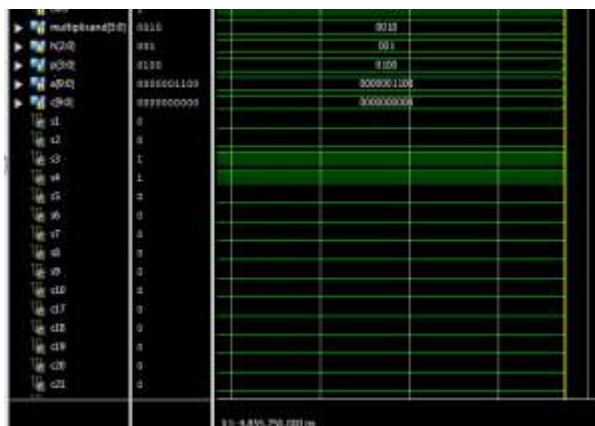
Mb result



Ppg result



Final sum csa



Authors Profile:

CHOWTAPALLI DAYAKAR is pursuing his M. Tech in Department of Electronics and Communication Engineering at Universal College of Engineering & Technology, Guntur. His specialization is VLSID

P.RANJITH KUMAR is an Assistant professor in the Department of Electronics and Communication Engineering at Universal College of Engineering & Technology, Guntur. He has published several papers in his 7 years of teaching experience on his interested area of VLSI signal processing.

References

1. Y.-H. Seo and D.-W. Kim, "A new VLSI architecture of parallel multiplier-accumulator based on Radix-2 modified Booth algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 2, pp. 201–208, Feb. 2010.
2. A. Peymandoust and G. de Micheli, "Using symbolic algebra in algorithmic level DSP synthesis," in *Proc. Design Automation Conf.*, Las Vegas, NV, 2001, pp. 277–282.
3. W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874, Mar. 2003.
4. C. N. Lyu and D. W. Matula, "Redundant binary Booth recoding," in *Proc. 12th Symp. Comput. Arithmetic*, 1995, pp. 50–57.
5. J. D. Bruguera and T. Lang, "Implementation of the FFT butterfly with redundant arithmetic," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 717–723, Oct. 1996.
6. W.-C. Yeh, "Arithmetic Module Design and its Application to FFT," Ph.D. dissertation, Dept. Electron. Eng., National Chiao-Tung University, Chiao-Tung, 2001.
7. R. Zimmermann and D. Q. Tran, "Optimized synthesis of sum-of-products," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, Washington, DC, 2003, pp. 867–872.
8. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford: Oxford Univ. Press, 2000.
9. O. L. Macsorley, "High-speed arithmetic in binary computers," *Proc. IRE*, vol. 49, no. 1, pp. 67–91, Jan. 1961.
10. N. H. E. Weste and D. M. Harris, "Datapath subsystems," in *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Readington: Addison-Wesley, 2010, ch. 11.
11. S. Xydis, I. Triantafyllou, G. Economakos, and K. Pekmestzi, "Flexible datapath synthesis through arithmetically optimized operation chaining," in *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*, 2009, pp. 407–414.
