Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partial Products*

David Villeger**, Vojin G. Oklobdzija
Electrical and Computer Engineering Department
University of California
Davis, CA 95616

Abstract

The Booth encoding method is evaluated in this paper. Although generally used in parallel multipliers, we show this scheme to be obsolete due to the improvements in bit compression trees. The number of gate levels with and without Booth encoding is compared when 4:2 compressors are used. It was found that a single row of 4:2 compressors reduces the number of partial products to one half, which is the essential function of the Booth encoding technique. We have found that a single row of 4:2 compressors achieves this reduction in less time and with fewer gates used. The case of 2's complement representation is discussed.

1. Introduction

Ever since its first introduction Booth encoding [1] has been a popular technique used to reduce the number of steps in the iterative multiplication process. From there it has by default found its way into the parallel multiplier implementation. The application of Booth recoding results in a reduction of the number of partial products by a factor of two in the initial step. That fact lead to an immediate acceptance of the Booth recoding technique in parallel multiplier implementations without even properly evaluating the technique. Though some authors have raised doubts about it [7] no one has done an in-depth analysis and comparison with other alternatives to reducing the number of partial products. In this paper we evaluate Booth encoding with respect to the use of 4:2 compressors and show that the same reduction can be achieved in less time and complexity. This finding is not only true in the case when 4:2compressors are used but can be extended to the use of higher order families.

* Supported by a grant from the UCD Office of Research and California MICRO #93-118

** Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique, 93162 Noisy le Grand CEDEX FRANCE

We have also shown that the same is true for the use of (3,2) counters in optimized Wallace Tree structures [12]. This finding comes from the fact that an equivalent amount of logic of a properly connected (3,2) counter will also reduce a number of partial products by one half in less time than when Booth recoding is implemented.

2. Booth-MacSorley recoding

The Booth algorithm [1] is widely used in the implementations of hardware or software multipliers because its application makes it possible to reduce the number of partial products. It can be used for both sign-magnitude numbers as well as 2's complement numbers with no need for a correction term or a correction step.

However, the Booth algorithm, if applied in its original form, produces the partial products, the number of which is data dependent and therefore not predictable. Though this feature does not represent a major problem in the software implementations of the algorithm, it is incompatible with the parallel implementations of the multiplier.

Table 1. *Modified Booth recoding*

$x_{i+2}x_{i+1}x_i$	Add to partial product
000	+0Y
001	+1Y
010	+1Y
011	+2Y
100	-2Y
101	-1Y
110	-1Y
111	-0Y

A modification of the Booth algorithm was proposed by Mac-Sorley [2] in which a triplet of bits is scanned instead of two bits. This technique has the advantage of reducing the number of partial products by one half regardless of the inputs. This scheme is summarized in Table 1.

The recoding is performed within two steps: encoding and selection. The purpose of the encoding is to scan the triplet of bits of the multiplier and define the operation to be performed on the multiplicand, as shown in Fig. 1.

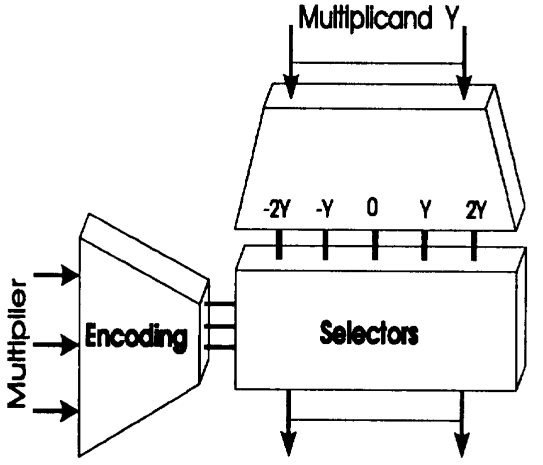

Fig. 1 . *Modified Booth recoding.*

This method is actually an application of a sign-digit representation in radix-4. The Booth-MacSorley algorithm, usually called the Modified Booth algorithm or simply the Booth algorithm, can be generalized to any radix. For example, a 3-bit recoding would require the following set of digits to be multiplied to the multiplicand : 0, ±1, ±2, ±3. The difficulty lies in the fact that ±3Y is computed by summing (or subtracting) 1 to ±2Y, which means that a carry propagation occurs. In spite of the speed improvement in the implementation of addition, the delay caused by the carry propagation renders this scheme to be slower than a conventional one.

Consequently, only the 2 bit Booth recoding is used and therefore considered in this paper. Therefore, our discussin in this paper is limited to the use of radix 4 and it does not include comparisons with some higher radix Booth algorithms [13].

2. The 4:2 compressor

Several attempts to find repeatable patterns in Wallace trees [3] have been made, leading to the notion of compressors such as 5:2 or 9:2 [4, 5]. The notion of compressors has been a major departure from the traditional notion of Dadda counters, since they require the use of *Carry-In* and *Carry-Out* signals. However, the propagation of the signal is limited to 1 bit by rendering the Carry-In and the corresponding Carry-Out independent. The most popular compressor is actually the 4:2 compressor, introduced by Weinberger [6] and used in [7-10].

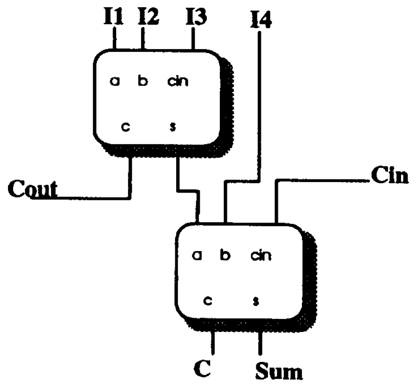

Fig. 2. *Structure of the 4:2 compressor built with Full Adders*

The structure of the 4:2 compressor is shown in Fig. 2. The major advantage of the use of this cell is that it allows a high regular layout. Indeed, the 2-to-1 reduction of the cell leads to a symmetric and regular compression tree.

However, since this cell is built with Full Adders, there is no improvement compared to the usual Wallace tree. It is fair to say that this scheme is even worse than a Wallace tree for a particular width of the multiplier. For example, if we consider that the critical path of a Full Adder is 2 XOR, and the number of levels of 4:2 compressors is 4 for a 24-bit multiplier, then the critical path of such a multiplier is 16 XOR. This number has to be compared with 14 XORs resulting from the application of 7 Full Adder levels in the case of a regular Wallace tree.

Consequently, several designers tried to redesign the 4:2 cell in order to reduce the critical path [9, 10]. An example of a redesigned 4:2 resulting in a 3 XOR critical path is shown in Fig. 3. This reduction occurs only on the path involving the Sum signal of the cell.

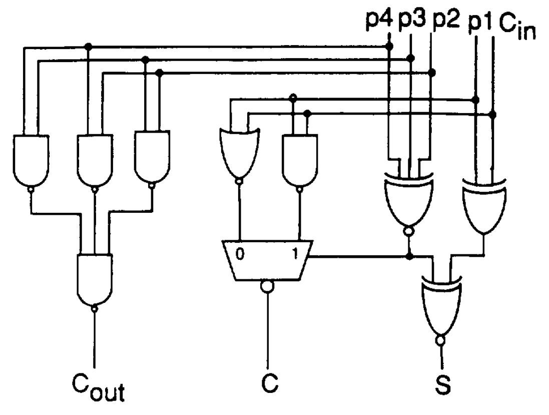

Fig. 3. *Logic of a redesigned 4:2 compressor*

However, as shown in Fig. 4, the path involving the *Carry-In* and the *Carry-Out* is also equivalent to 3 XOR gate delays. In other words, the worst case to cross a level of 4:2 compressor is 3 XOR.

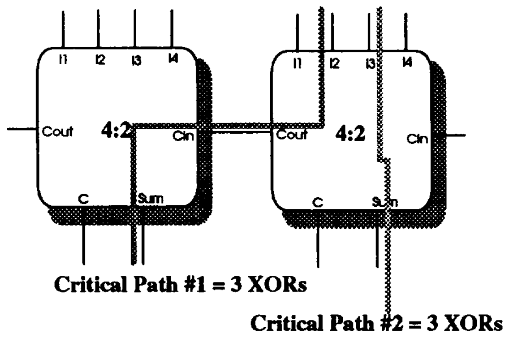

Fig. 4. *Critical paths in a row a 4:2 compressors*

3. Booth recoding versus the use of 4:2 compressors

Booth recoding necessitates the internal use of 2's complement representation in order to efficiently perform subtraction of the partial products as well as additions. However, floating point standard specifies sign magnitude representation which is also followed by most of the non-standard floating point number representations in use. Thus, we assume the use of sign magnitude representation and compare those multiplier implementations using Booth encoding with the ones not using it but resorting to efficient partial product addition techniques such as the use of 4:2 compressors. The conclusions are summarized in Table 2.

The advantage of Booth recoding is that it generates only half of the partial products compared to the multiplier implementation which does not use Booth recoding. However, the benefit achieved comes at the expense of increased hardware complexity. Indeed, this implementation requires hardware for the encoding and for the selection of the partial products (0, ±Y, ±2Y). An optimized encoding is shown in Fig. 5. The multiplexers and buffers are considered to be equivalent to an XOR gate. This implementation is then equivalent to one level of XOR gates and one level of AND gates.

The selection can be implemented with a simple 5 input multiplexer, which is roughly equivalent to 3 XOR gates. However, since one input is grounded, this circuit can be designed with only a 4 input multiplexer, that is 2 XOR gates, and an AND gate. In this case, the Booth recoding circuit is equivalent to 3 XOR plus 2 AND gates.

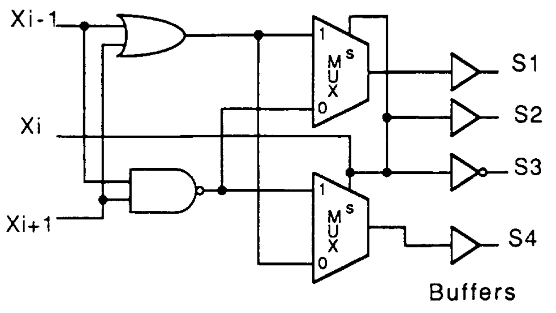

Fig.5. *An Optimized Encoding Circuit*

On the other hand, reducing the number of partial products by one half can be achieved with one level of AND gates and one row of 4:2 compressors. The 4:2 cell is designed with 3 XOR levels of delay as shown in Fig. 3. and implemented in [9,10]. The use of higher order compressors [5] would furter lower this delay.

However, the main disadvantage of the Booth technique is the complexity introduced by the internal use of 2's complement representation which is necessary to compute negative partial products. Indeed, since the Booth recoding method calculates -Y and -2Y, it needs to extend the sign of negative partial products. It further needs to complement Y when -Y or -2Y is requested, that is to calculate $-Y = \overline{Y} + 1$ where $\overline{Y}$ means inversion of every bit of Y. Consequently, two extra bits are necessary in the scheme: one for the sign extension and one for conversion into 2's complement. Both of the bits will be placed in the same row, therefore not increasing the number of rows. However, the correction bit (which is needed for correct sign calculation) will be placed right in the middle of the multiplier tree, therefore not only increasing the number of rows by one but creating this increase in the worst possible place, i.e. in the critical path of the multiplier.

Table 2. *Comparison of sign magnitude number multiplication using Booth encoding and sign-magnitude number multiplication not using Booth encoding.*

With Booth encoding	Without Booth encoding
Internal representation : 2's complement (some partial products need to be subtracted).	*Internal representation :* sign magnitude (all the partial products are positive).
Hardware for encoding and selection	One row of 4:2 compressors
Sign extension.	Only a XOR is used to compute the sign in parallel.
2 extra bits (sign extension and complementation).	No extra bit.
The normalization requires some Leading Zero Detectors and Leading One Detectors.	The normalization and even the rounding are easy [5].
The schematic and the layout are not regular.	The simplicity of the schematic allows a highly regular layout.
1 XOR (encoding) plus 3 XOR (multiplexer) equals 4 XOR.	1 AND (partial product generation) plus 3 XOR (4:2 compressor).

4. Conclusion

When Booth recoding is used, the schematic and the layout of the resulting implementation are less regular leading to a more difficult design or VHDL description. In terms of speed, the Booth technique is at best equal or worse than the use of the 4:2 compressors. In case of 2's complement representation and without Booth encoding, the last row of partial product (depending on the sign of the multiplier) is generated by using an AND gate with an inverted input. In other words, the number of gate levels is the same as in the sign magnitude case. However, in 2's complement case the sign extension is still needed. This feature makes the two schemes comparable. The scheme which does not use Booth encoding is slightly better because of the simplicity and the fewer number of gate levels.

Acknowledgment
Careful reading and comments of Antonio De La Serna are very much appreciated.

References

[1] A. D. Booth, *"A Signed Binary Multiplication Technique"*, Quarterly J. Mechan. Appl. Math., Vol. IV, 1951.
[2] O. L. Mac Sorley, *"High Speed Arithmetic in Binary Computers"*, IRE Proc., 1961.
[3] C. S. Wallace, *"A Suggestion For A Fast Multiplier"*, IEEE Transaction on Computers, Vol. EC13, pp. 14-17, February 1964.
[4] R. S. Lim, *"High-Speed Multiplication and multiple summand addition"*, 4th Intnternational Symposium on Computer Arithmetic, Santa Monica, California, June 1978.
[5] P. Song, G. De Michelli, *"Circuit and Architecture Trade-offs for High Speed Multiplication"*, IEEE Journal of Solid State Circuits, Vol.26, No.9, September 1991.
[6] A. Weinberger, *"4:2 Carry-Save Adder Module"*, IBM Technical Disclosure Bulletin.,Vol.23, January 1981.
[7] M. R. Santoro, *"A Pipelined 64X64b Iterative Array Multiplier"*, Digest of Technical Papers Int'l Solid-State Circuits Conference, February 1988.
[8] M. Nagamatsu et al, *"A 15nS 32X32-bit CMOS Multiplier with an Improved Parallel Structure"*, Digest of Technical papers, IEEE Custom Integrated Circuits Conference 1989.
[9] J. Mori et al, *"A 10nS 54X54-b Parallel Structured Full Array Multiplier with 0.5-u CMOS Technology"*, IEEE Journal of Solid State Circuits, Vol. 26, No. 4, April 1991.
[10] T. Soulas, D.Villeger, V. G. Oklobdzija, *"An ASIC Multiplier for Complex Numbers"*, Proceedings of EURO-ASIC-93, The European Event in ASIC Design, Paris, France, February 22-25, 1993.
[11] M. R. Santoro, G. Bewick, M. Horowitz, *"Rounding Algorithms For IEEE Multipliers"*, IEEE 9th Symposium on Computer Arithmetic proceedings, September 1989.
[12] V. G. Oklobdzija, D.Villeger, S. S. Liu, *"A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach"*, Submitted to IEEE Transactions on Computers, October 1993.
[13] P.E.Madrid, B. Miller, E.E. Schwartzlander, *"Modified Booth Algorithm for High Radix Fixed-Point Multiplication"*, IEEE Transactions on VLSI Systems, Vol.1, No.2, June 1993.