Evaluation of Booth's Algorithm for Implementation in Parallel Multipliers

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

Abstract

As it has been introduced by Vojin Oklobzija and David Villeger in [1], the Booth encoding technique, used in parallel multipliers seems to be obsolete because of the improvement of compression trees using 4:2 compressors. This artcicle compares the two techniques in the case of different lengths of multipliers, and it appears that the reduction bit with 4:2 compressors allows a higher speed and a highly regular layout since the schematic is simple and repetitive.

Introduction

Because multiplication is a slow operation, it is on the critical path of microprocessors, especially of DSPs, which compute integer and floating point operations in parallel within one cycle. A multiplication can be divided in three parts: The partial products generation, the summation network (or the column compression tree), and the final carry propagate adder. Since the delay of a multiplier depends on the number of partial products to be added, it is attractive to use Booth's Algorithm [2], because it reduces it by almost a factor two, but also generates some extra-bits for the sign extension and the 2's complementation. In this paper, we will compare Booth encoding with the use of 4:2 compressors to achieve the same reduction in the number of partial products.

Number representation

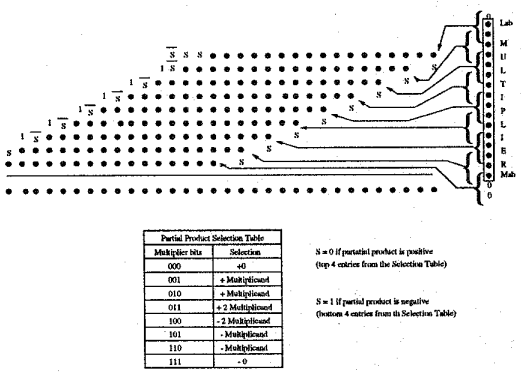
Since multiplication is more complicated with 2's complement, the floating point standard uses signed-magnitude representation. This allows the sign of the product to be computed as the XOR of the signs of the multiplier and the multiplicand. Thus we only need to compare the different algorithms for the case of a positive multiplier and a positive multiplicand.

*Ecole Supérieure d'Ingénieur en Electrotechnique et Electronique, 93162 Noisy le Grand CEDEX FRANCE

Booth encoding

The Booth2 Algorithm

Since Booth encoding creates a smaller number of partial products, and therefore allows their summation to be faster, it is often used in the implementation of parallel multipliers. By grouping the bits of the multiplier into triplets and selecting the partial products from the set 0,+M, +2M, -M, -2M, where M is the multiplicand, the number of partial products can be reduced from N to $\frac{\lfloor N+2 \rfloor}{2}$ (where N is the operand length). The rules of this algorithm, which have been improved by G. W. Bewick (solving the problem of the sign extension [4]) are summarised in Figure 1.



Partial Product Selection Table	
Multiplier bits	Selection
000	+0
001	+ Multiplicand
010	+ Multiplicand
011	+ 2 Multiplicand
100	- 2 Multiplicand
101	- Multiplicand
110	- Multiplicand
111	- 0

S = 0 if partatial product is positive (top 4 entries from the Selection Table)

S = 1 if partial product is negative (bottom 4 entries from th Selection Table)

Figure 1: *Example of 16-bit multiplication using Booth2 Algorithm, from [4].*

This method can be generalized, grouping the bits of the multiplier by 4, 5, or more. Then the difficulty will be to compile +3M , -3M, +5M, -5M, since summing or substracting 4M with M can generate a carry propagation, rending this scheme much slower than the Booth2 Algorithm where 2M is obtained by

1058-6393/96 $5.00 © 1996 IEEE
Proceedings of ASILOMAR-29

shifting the multiplicand one bit to the left.

Critical path

Figure 2 shows the critical path delay of a N-bit multiplier that uses Booth encoding as well as 4:2 compressors and full adders for bit compression, using the fact that the critical path of a 4:2 compressor is 3 XOR [1] (The table does not include the Booth encoding delay).

Nb of Bits	Delay in XOR	Nb of Bits	Delay in XOR	Nb of Bits	Delay in XOR
4	1(FA)=2XOR	10	2(4:2)=6XOR	31	3(4:2)=9XOR
5	1(FA)=2XOR	14	2(4:2)=6XOR	32	3(4:2)+1(FA)=11XOR
6	1(4:2)=3XOR	15	2(4:2)=6XOR	33	3(4:2)+1(FA)=11XOR
7	1(4:2)=3XOR	16	2(4:2)+1(FA)=8XOR	62	4(4:2)=12XOR
8	1(4:2)+1(FA)=5XOR	17	2(4:2)+1(FA)=8XOR	63	4(4:2)=12XOR
9	1(4:2)+1(FA)=5XOR	30	3(4:2)=9XOR	64	4(4:2)+1(FA)=14XOR

Figure 2: *Critical paths of a multiplier using Booth2 Algoritm.*

Using 4:2 compressors

The 4:2 compressors

The use of 4:2 compressors reduces the number of partial products to be added by one half. This idea was first introduced by A. Weinberger [5], and improoved by V. G. Oklobdzija and D. Villeger [1]. Figure 3 shows how a sequence of 4:2 compressors is used to achieve this result, and Figure 4 shows their application in the partial products reduction for 8-bit multiplication.

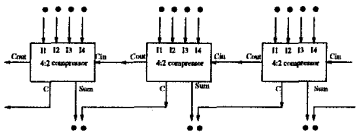


Figure 3: *sequence of 4:2 compressors.*

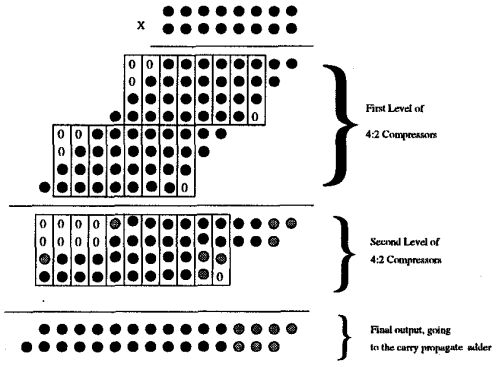


Figure 4: *Reduction of 8 partial products using 4:2 compressors*

Critical path

Figure 5 shows the critical path of a N-bit multiplier using 4:2 compressors and full adders to reduce the number of partial products. This table does not include the delays of the AND gates which generate the partial products.

Nb of Bits	Delay in XOR	Nb of Bits	Delay in XOR	Nb of Bits	Delay in XOR
4	1(4:2)=3XOR	11	3(4:2)=9XOR	33	4(4:2)+1(FA)=14XOR
5	1(4:2)+1(FA)=5XOR	16	3(4:2)=9XOR	34	4(4:2)+1(FA)=14XOR
6	2(4:2)=6XOR	17	3(4:2)+1(FA)=11XOR	35	5(4:2)=15XOR
8	2(4:2)=6XOR	18	3(4:2)+1(FA)=11XOR	64	3(4:2)=15XOR
9	2(4:2)+1(FA)=8XOR	19	4(4:2)=12XOR	65	5(4:2)+1(FA)=17XOR
10	2(4:2)+1(FA)=8XOR	32	4(4:2)=12XOR	66	2(4:2)+1(FA)=17XOR

Figure 5: *Critical paths of a multiplier using 4:2 compressors.*

Booth encoding versus 4:2 compressors

As we said before, a single row of 4:2 compressors can achieve a better reduction of the number of partial products than the Booth2 Algorithm ($\frac{N}{2}$ versus $\frac{\lfloor N+2 \rfloor}{2}$. In this part, we will look at the schemes shown in Figure 6 and compare the speed improvment of the two methods. Figure 7 graphs the critical path delays of a multiplier against the length of the multiplicands when using Booth's Algorithm and when using only 4:2 compressors. These delays do not include the time for Booth Encoding (for the Booth Multiplier) or for the AND gate (that generates each partial product in the multiplier using only 4:2 compressors).

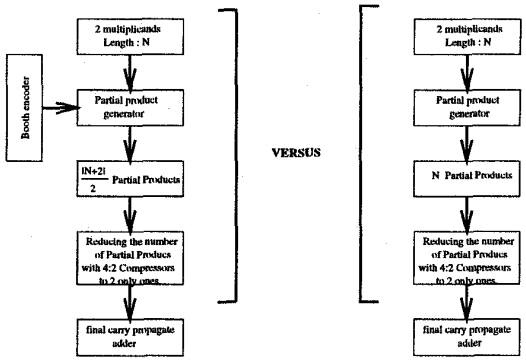


Figure 6: *The two schemes to be compared.*

The delay using Booth Encoding is universally smaller. However, the difference in the critical path delays for the two methods is minimum at one XOR gate delay when the length of the operands is a power of 2, and two XOR gate delays when the length of the operands is $2^N + 1$. In both of those cases the Booth Algorithm implementation will certainly be

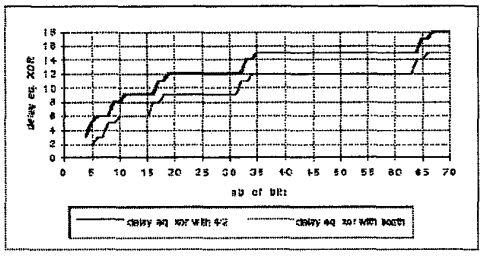


Figure 7: *Critical paths of a multiplierdepending on the operands' length.*

slower since the delay of the Booth encoder is higher than 1 AND gate and 2 XOR gates. In fact, to prove that using the Booth Algorithm is worse than using only 4:2 compressors, we must look at a case where the difference in the critical path delays is maximized at 3 XOR delays. Therefore, we will examine the case of a 24-bit multiplier and compare the delays of the two methods for achieving the first reduction of the number of partial products by a half. The tools we used to complete this are Viewdraw and Epoch (to compile the design and analyze the layout and the critical paths).

Example for a 24-bit multiplier

The design of the parts of 24-bit multipliers to generate the partial products and reduce their number by half (using Booth's Algorithm and using only 4:2 compressors) was compiled using the technology mos1u3m1p from Epoch library. The average temperature was assumed to be 25C, with a 5 Volt power supply and an external capacity load of 50 fF added at each output of the design.
The results obtained from the timing analysis of the layouts are summarized in Figure 8.

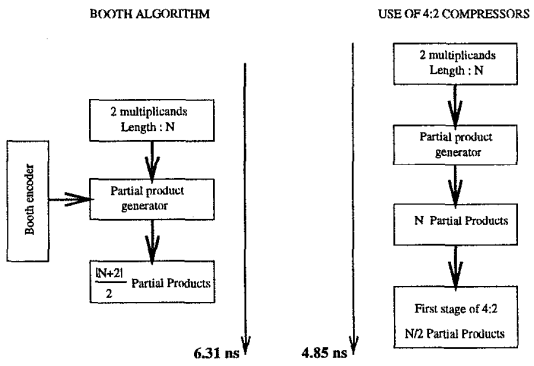


Figure 8: *delays of the schemes.*

The scheme involving the Booth encoding was found to be slower than the one involving the 4:2 compressors. The difference between the two delays in the simulation is 1.46ns. This is the result of a number of factors. First, the use of 4:2 compressors allows a higher regularity of the layout which mean less wire and improved speed compared to the Booth Encoder which is more irregular and as a result more difficult to route. Additionally, the Booth Encoding must choose between the partial product M and 2M; therefore the outputs of the partial product selector (selecting the partial product in the set M, 2M) must be able to drive $2N + 1$ elementary gates, which require some buffers. This increases the power consumption and also the delays. On the other hand, the Booth algorithm requires less hardware than using 4:2 compressors, which is an advantage, regarding to the price of chip.

Conclusion

We have schown in this paper that the use of 4:2 compressors can achieve in the worst case, the same reduction of the number of partial products as Booth's Algorithm in less time. The majority of multipliers designed today use Booth's Algorithm, which may not be the best choice. In the future designers should consider using 4:2 compressors instead of Booth encoding when designing multipliers for high-speed microprocessors. Moreover, it has been proved that the use of higher order compressors (9:2 for example...) would result in greater increases in speed [1].

References

[1] V. G. Oklobdzija, D. Villeger, " *Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partail Products*".

[2] A. D. Booth, " *A signed Binary Multiplication Technique*" , Quaterly J. Mechan. Appl. Math., Vol. IV, 1951.

[3] 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.

[4] Gary W. Bewick, " *Fast Multiplication : Algorithms and Implementation*", Dissertation for the Degree of Doctor in Philosophy, February 1994.

[5] A. Weinberger, *4:2 carry-save adder module*", IBM tech. Disclosure Bulletin, Vol. 23, Jan. 1981.