International Journal of Electronics Signals and Systems

Volume 2 | Issue 3 Article 6

January 2013

PERFORMANCE EVALUATION OF BOOTH AND WALLACE MULTIPLIER USING FIR FILTER

H. RAGHUNATHA RAO
Department of Electronics and Communication Engineering, Chirala Engineering College, Chirala.,
h.raghunath@gmail.com

PERFORMANCE EVALUATION OF BOOTH AND WALLACE MULTIPLIER USING FIR FILTER

[1]H. RAGHUNATHA RAO, [2]T. ASHOK KUMAR & [3]N.SURESH BABU

[1,2&3]Department of Electronics and Communication Engineering,
Chirala Engineering College, Chirala.

Abstract: An area-and speed efficient multipliers is proposed in the thesis. the proposed booth and Wallace multipliers shows the tradeoff in the performance evaluation for the fir filter applications. For implementation of fir filter in this paper the adders introduced are carry save adder and carry skip adder. For evaluating the fir filter performance the tested combinations are booth carry save , booth carry skip , Wallace carry save , Wallace carry skip.

Keywords: *carry save adder, carry skip adder, booth and Wallace multiplier.*

1. INTRODUCTION

The design of high-speed, area-efficient multipliers is essential for VLSI implementations of digital signal processing systems.Multiplication are frequently required in digital signal processing. Parallel multipliers provide a high-speed method for multiplication, but require large area for VLSI implementations. In most signal processing applications, a product is desired to avoid growth in word-size. Thus, an important design goal is to reduce the area requirements of output multipliers. This paper presents a technique for multiplication which computes the product of two numbers by summing the partial products.

The dominant factors in the design of multipliers for digital filters and other digital signal processing (DSP) applications are the chip area required and the speed of operation. Among the many classes of multipliers, array multipliers and multipliers based on the Booth algorithm have been popular. In array multipliers, multiplication is effected by adding all the partial products generated by an array of AND-gate cells.

In the past few years, significant reduction in the chip area as well as an associated increase in the speed of operation of these multipliers have been achieved through increased device density by taking advantage of advancements in VLSI technology. Unfortunately, this approach has reached a stage of diminishing returns and further improvements in the design of multipliers will occur only if major breakthroughs are achieved in the technology.

In this paper, we explore an alternative approach to the design of area-efficient multipliers for DSP applications, which is independent of technology. Very often in these applications fixed-point arithmetic is used and typically N-bit signals are multiplied by N-bit coefficients.

Two N-bit numbers to be multiplied, *A* and B, and their product *P* can be represented as

$$A = \sum_{i=0}^{N-1} a_i 2^{i-N}$$

$$B = \sum_{i=0}^{N-1} b_i 2^{i-N}$$

$$P = \sum_{i=0}^{2N-1} p_i 2^{i-2N}$$

respectively, where a,, b,, p, E (0, I}.

In the standard N x N parallel multiplier, the N2 bit products are generated simultaneously and are then added by an array of full adders, for an 8 x 8 multiplier.

2. EVALUATION OF CARRY SAVE AND CARRY SKIP ADDERS

A **carry-save adder** is a type of digital adder, used in computer micro architecture to compute the sum of three or more *n*-bit numbers in binary. CSA consists of a sequence of full adders, in which one of the operands is entered in the carry inputs, and the carry outputs, instead of feeding the carry inputs of the following full adders, form a second output word which is then added to the ordinary output in a two-operand adder to form the final sum.

A carry-save adder is a kind of adder with low propagation delay (critical path), but instead of adding two input numbers to a single sum output, it adds three input numbers to an output pair of numbers.

When its two outputs are then summed by a traditional carry-look ahead or ripple-carry adder, we get the sum of all three inputs. When adding three or more numbers together, a sequence of carry-save adders terminated by a single carry-look ahead adder provides much better propagation delays than a sequence of carry-look ahead adders.

In particular, the propagation delay of a carry-save adder is not affected by the width of the vectors being added.

The carry save adder can be viewed as follows i.e.

- A digital adder
- Used to sum 3 or more binary numbers
- Outputs two numbers of equal dimensions as the input

Advantages of Carry Save Adder

- Produces all of its outputs in parallel resulting in the same delay as a full adder.
- Very little propagation delay when implemented.
- Allows for high clock speeds.

Carry Skip Adder is an alternative way of reducing the delay in the carry-chain of a RCA by checking if a carry will propagate through to the next block. This is called carry-skip adders.

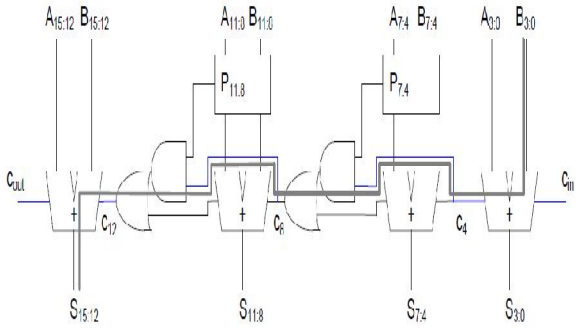

Figure 1 Carry-Skip Adder.

The carry-out of each block is determined by selecting the carry-in and Gi:j using Pi:j. When Pi:j = 1, the carry-in cj is allowed to get through the block immediately. Otherwise, the carry-out is determined by Gi:j. The CSKA has less delay in the carry-chain with only a little additional extra logic. Further improvement can be achieved generally by making the central block sizes larger and the two-end block sizes smaller.

Advantages of Carry Skip Adder

- Relatively constant increase in performance.
- The performance and the consistency of the carry skip adder will be excellent.
- There is no guaranteed wasted power.

3. WALLACE TREE MULTIPLIER

The Wallace tree multiplier will use a set of adders to produce the final outputs. Carry propagate adds are relatively slow, because of the long wires needed to propagate carries from low order bits to high order bits.

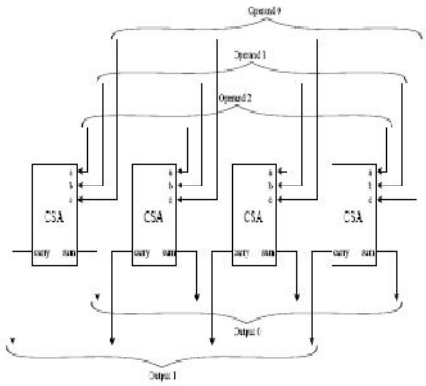

Figure 2 Reducing 3 operands to 2 using CSAs

Probably the single most important advance in improving the speed of multipliers, pioneered by Wallace is the use of carry save adders (CSAs also known as combination of full adders and half adders or 3-2 counters), to add three or more numbers in a redundant and carry propagate free manner. The method is illustrated in Figure 2.

By applying the basic three input adder in a recursive manner, any number of partial products can be added and reduced to 2 numbers without a carry propagate adder. A single carry propagate addition is only needed in the final step to reduce the 2 numbers to a single, final product. The general method can be applied to trees and linear arrays alike to improve the performance.

Wallace Trees are combinatorial logic circuits used to multiply binary integers. Constructed using full adders and half adders, they are a fast, efficient method to implement multiplication. Integer multiplication can be performed using any of several methods. The traditional shift-add approach and ROM lookup tables are two methods used to implement multiplication, but each has its drawbacks. The time needed to calculate products using the shift-add method increases linearly as the number of bits in the operands increases, and the size of the lookup ROM increases exponentially with increases in the size of the operands.

Wallace Trees, which use full and half adders to calculate partial results, resolve much of these problems without unduly increasing hardware requirements. Although it requires more hardware than shift-add multipliers, it produces a product in far less time.

The advantages of the Wallace Tree Multiplier are listed as follows:

•Each layer of the tree reduces the number of vectors by a factor of 3:2

•Minimum propagation delay.

•The benefit of the Wallace tree is that there are only O (log n) reduction layers, but adding partial products with regular adders would require O (log n) 2 times.

4. BOOTH MULTIPLIER

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed[3]. Booth's algorithm is of interest in the study of computer architecture. Booth's algorithm examines adjacent pairs of bits of the *N*-bit multiplier *Y* in signed two's complement representation.

It includes an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to N-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P remains unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product.

The representation of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. Here, the order of the steps is not determined.

Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P. There are many variations and optimizations on these details.

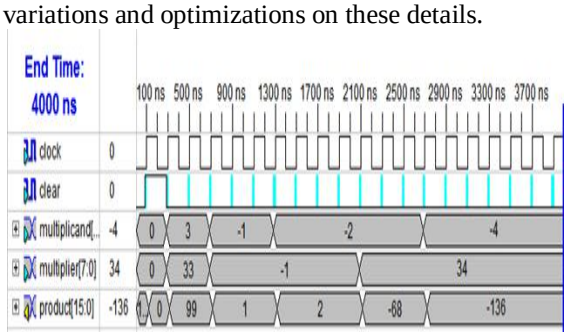

Figure 3 Wallace Waveform

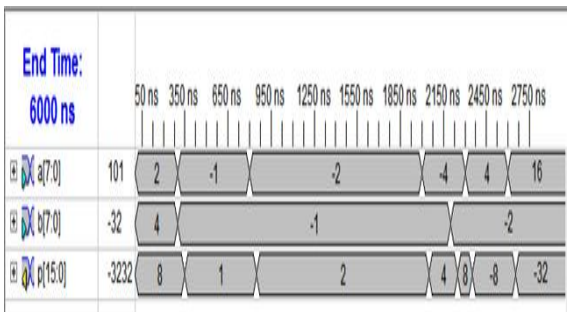

Figure 4 Booth Waveform

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r.

Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).

A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros.
S: Fill the most significant bits with the value of (−m) in two's complement notation. Fill the remaining (y + 1) bits with zeros.

P: Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bit with a zero.

Determine the two least significant (rightmost) bits of P.

If they are 01, find the value of P + A. Ignore any overflow. If they are 10, find the value of P + S. Ignore any overflow.

If they are 00, do nothing. Use P directly in the next step. If they are 11, do nothing. Use P directly in the next step. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.

Repeat steps 2 and 3 until they have been done y times. Drop the least significant (rightmost) bit from P. This is the product of m and r. The Booth and wallance waveforms are as shown in Figure 3 and Figure 4.

5. CONCLUSION

By the results we can conclude that Wallace multiplier is area and speed efficient than booth multiplier and booth multiplier is power efficient. Compared with the existing works, the proposed methods has better area power and speed efficient. Trade off is there between the speed, power and area depend on the applications.

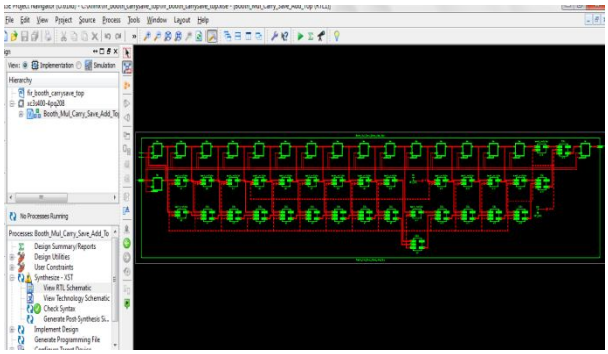

Figure 5 RTL Schematic for Booths Multiplier

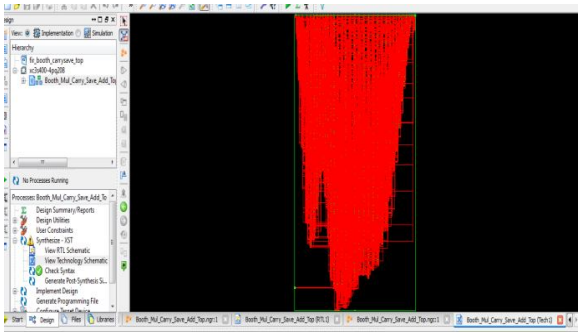

Figure 6 Technology Schematic

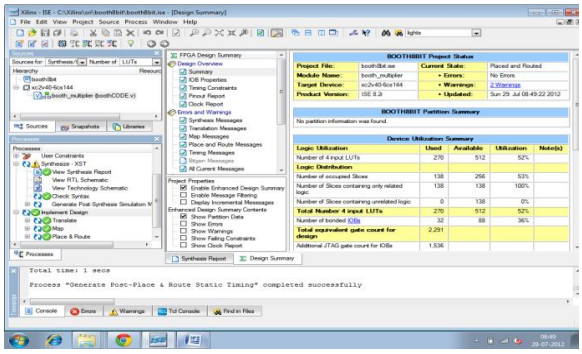

Figure 7 Design Summary of Booths Multiplier

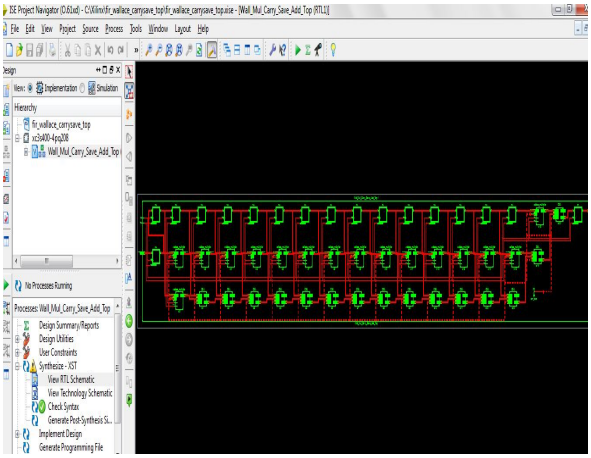

Figure 8 RTL Schematic of Wallace Tree Multiplier

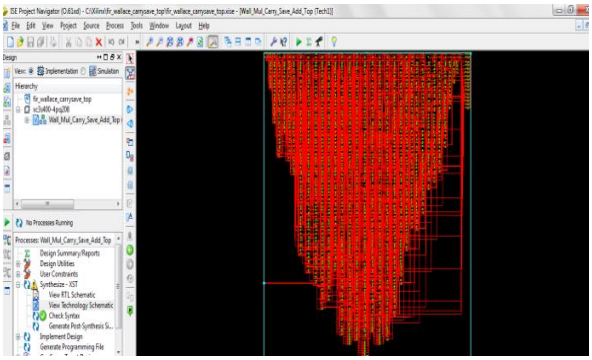

Figure 9 Technology Schematic of Wallace Tree Multiplier

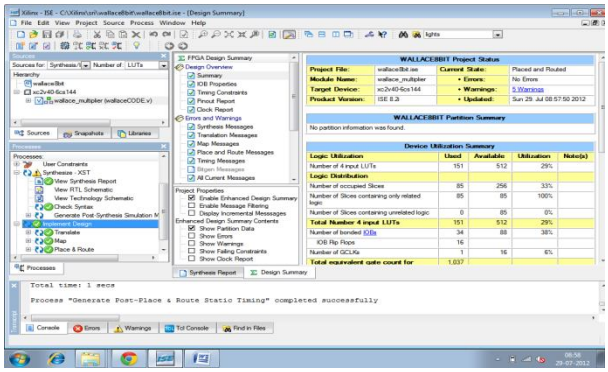

Figure 10 Design Summary of Wallace tree multiplier

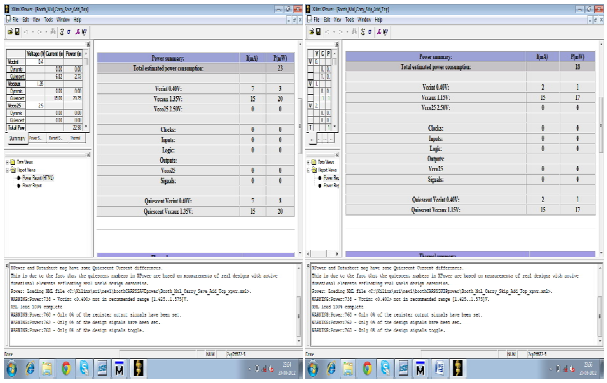

Figure 11 FIR Implementation using Booth Multipliers

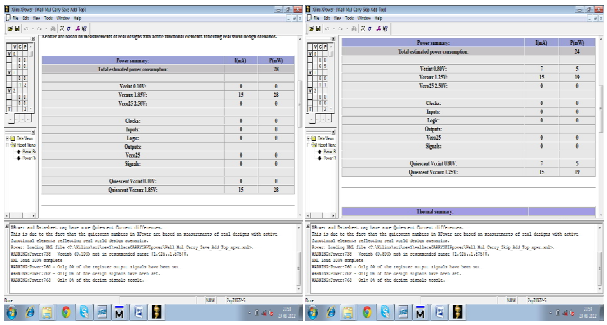

Figure 12 FIR Implementation using Wallace Multipliers

Table 1 Comparison report

	TIMING REPORT	POWER	MEMORY
FIR_BOOTH_MUL_CARRY_SAVE	77.446 ns	23mw	280.51 MB
FIR_BOOTH_MUL_CARRY_SKIP	57.218ns	18mw	325.14MB
FIR_WALLACE_MUL_CARRY_SAVE	50.503ns	28mw	275.02MB
FIR_WALLACE_MUL_CARRY_SKIP	34.884ns	24mw	286.08MB

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

REFERENCES:

[1] J. M. Jou, S. R. Kuang, and R. D. Chen, "Design of lower-error fixedwidth multipliers for DSP applications," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process, vol. 46, no. 6, pp. 836–842, Jun. 1999.

[2] L. D. Van, S. S. Wang, and W. S. Feng, "Design of the lower-error fixedwidth multiplier and its application," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process, vol. 47, no. 10, pp. 1112–1118, Oct. 2000.

[3] L. D. Van and C. C. Yang, "Generalized low-error area-efficient fixedwidth multipliers," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

[4] Y. C. Liao, H. C. Chang, and C. W. Liu, "Carry estimation for two's complement fixed-width multipliers," in Proc. IEEE Workshop on Signal Process. Syst. Design Implementation, 2006, pp. 345–350. [5] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 6, pp. 1312–1325, Jun. 2010.

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

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

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

[9] T. B. Juang and S. F. Hsiao, "Low-error carry-free fixed-width multipliers with low-cost compensation circuits," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 52, no. 6, pp. 299–303, Jun. 2005.

[10] K. K. Parhi, J. G. Chung, K. C. Lee, and K. J. Cho, "Low-error fixed-width modified Booth multiplier," U.S. Patent 7 334 200, Feb. 19, 2008.

[11] H. A. Huang, Y. C. Liao, and H. C. Chang, "A self-compensation fixedwidth Booth multiplier and its 128-point FFT applications," in Proc. IEEE ISCAS, 2006, pp. 3538–3541.

[12] J. P. Wang, S. R. Kuang, and S. C. Liang, "High-accuracy fixedwidth modified Booth multipliers for lossy applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 1, pp. 52–60, Jan. 2011.

Authors Profile:

H.Raghunatha Rao is pursuing M.Tech in VLSI &ES at Chirala Engineering College,Chirala.

T.Ashok kumar is Working as Assoc.prof in ECE Dept. CEC,Chirala.He was Awarded with M.Tech in CSE from Anna University He has over 5 year teaching Experience.

Prof.N.Suresh Babu is Vice- Principal & HOD of ECE Dept in CEC,Chirala.He got his M.Tech in Microwave Engineering from Birla Innstitute of technology, Ranchi. He has 13 years of teaching Experience and 2 Years of Industrial Experience in various organisations .

◈◈◈