

Parallel Multiplier-Accumulator Based on Radix-2 Modified Booth Algorithm by using A VLSI Architecture

A. Sathya¹, S. Fathimabee², S. Divya³

^{1,2,3}M.E Communication System, Department of ECE, Mailam Engineering College

Abstract: In this paper, we proposed a new architecture of multiplier-and-accumulator (MAC) for high-speed arithmetic. By combining multiplication with accumulation and devising a hybrid type of carry save adder (CSA), the performance was improved. Since the accumulator that has the largest delay in MAC was merged into CSA, the overall performance was elevated. The proposed CSA tree uses 1's-complement-based radix-2 modified Booth's algorithm (MBA) and has the modified array for the sign extension in order to increase the bit density of the operands. The CSA propagates the carries to the least significant bits of the partial products and generates the least significant bits in advance to decrease the number of the input bits of the final adder. Also, the proposed MAC accumulates the intermediate results in the type of sum and carries bits instead of the output of the final adder, which made it possible to optimize the pipeline scheme to improve the performance. The proposed architecture was synthesized with 250, 180 and 130 μm , and 90 nm standard CMOS library. Based on the theoretical and experimental estimation, we analyzed the results such as the amount of hardware resources, delay, and pipelining scheme. We used Sakurai's alpha power law for the delay modeling. The proposed MAC showed the superior properties to the standard design in many ways and performance twice as much as the previous research in the similar clock frequency. We expect that the proposed MAC can be adapted to various fields requiring high performance such as the signal processing areas.

Keywords: Booth multiplier, carry save adder (CSA) tree, computer arithmetic, digital signal processing (DSP), multiplier-and-accumulator (MAC)

1. Introduction

With the recent rapid advances in multimedia and communication systems, real-time signal processing like audio signal processing, video/image processing, or large-capacity data processing are increasingly being demanded. The multiplier and multiplier-and-accumulator (MAC) [1] are the essential elements of the digital signal processing such as filtering, convolution, and inner products. Most digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) [2] or discrete wavelet transform (DWT) [3]. 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) [4] is commonly used. However, this cannot completely solve the problem due to the long critical path for multiplication [5], [6].

In general, a multiplier uses Booth's algorithm [7] and array of full adders (FAs), or Wallace tree [8] instead of the array of FAs., i.e., this multiplier mainly consists of the three parts: Booth encoder, a tree to compress the partial products such as Wallace tree, and final adder [9], [10]. The most effective way to increase the speed of a multiplier is to reduce the number of the partial products because multiplication proceeds a series of additions for the partial products. 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 products.

One of the most advanced types of MAC for general-purpose digital signal processing has been proposed by Elguibaly [17]. It is an architecture in which accumulation has been

combined with the carry save adder (CSA) tree that compresses partial products. In the architecture proposed in [17], the critical path was reduced by eliminating the adder for accumulation and decreasing the number of input bits in the final adder. While it has a better performance because of the reduced critical path compared to the previous MAC architectures, there is a need to improve the output rate due to the use of the final adder results for accumulation. Architecture to merge the adder block to the accumulator register in the MAC operator was proposed in [18] to provide the possibility of using two separate $N/2$ -bit adders instead of one N -bit adder to accumulate the N -bit MAC results. Recently, Zicari proposed an architecture that took a merging technique to fully utilize the 4-2 compressor [19]. It also took this compressor as the basic building blocks for the multiplication circuit.

This paper is organized as follows. In Section II, a simple introduction of a general MAC will be given, and the architecture for the proposed MAC will be described in Section III. In Section IV, the implementation result will be analyzed and the characteristic of the proposed MAC will be shown. Finally, the conclusion will be given in Section V.

2. Overview of MAC

In this section, basic MAC operation is introduced. A multiplier can be divided into three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand (X) and the multiplier (Y). The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry. If the process to accumulate the multiplied results

is included, a MAC consists of four steps, as shown in Fig. 1, which shows the operational steps explicitly.

General hardware architecture of this MAC is shown in Fig. 2. It executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y . This is added to the previous multiplication result Z as the accumulation step.

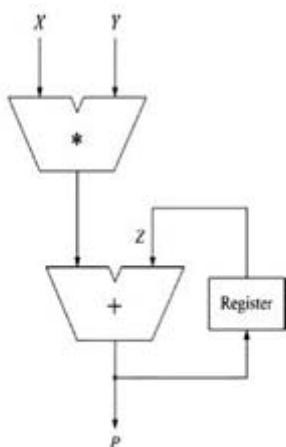


Fig. 2. Hardware architecture of general MAC.

The N -bit 2's complement binary number X can be expressed as

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, \quad x_i \in \{0, 1\}. \quad (1)$$

If (1) is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm, it would be [7].

$$X = \sum_{i=0}^{N/2-1} d_i 4_i \quad (2)$$

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}. \quad (3)$$

If (2) is used, multiplication can be expressed as

$$X \times Y = \sum_{i=0}^{N/2-1} d_i 2^{2i} Y. \quad (4)$$

If these equations are used, the afore-mentioned multiplication-accumulation results can be expressed as

$$P = X \times Y + Z = \sum_{i=0}^{N/2-1} d_i 2^i Y + \sum_{j=0}^{2N-1} z_j 2^j. \quad (5)$$

Each of the two terms on the right-hand side of (5) is calculated independently and the final result is produced by adding the two results. The MAC architecture implemented by (5) is called the standard design [6].

3. Proposed MAC architecture

In this section, the expression for the new arithmetic will be derived from equations of the standard design. From this result, VLSI architecture for the new MAC will be proposed. In addition, a hybrid-typed CSA architecture that can satisfy the operation of the proposed MAC will be proposed.

A. Derivation of MAC Arithmetic

1) *Basic Concept:* If an operation to multiply two TV-bit numbers and accumulate into a 27V-bit number is considered, the critical path is determined by the 27V-bit accumulation operation. The overall performance of the proposed MAC is improved by eliminating the accumulator itself by combining it with the CSA function. If the accumulator has been eliminated, the critical path is then determined by the final adder in the multiplier. The basic method to improve the performance of the final adder is to decrease the number of input bits. In order to reduce this number of input bits, the multiple partial products are compressed into a sum and a carry by CSA. The number of bits of sums and carries to be transferred to the final adder is reduced by adding the lower bits of sums and carries in advance within the range in which the overall performance will not be degraded. A 2-bit CLA is used to add the lower bits in the CSA. In addition, to increase the output rate when pipelining is applied, the sums and carries from the CSA are accumulated instead of the outputs from the final adder in the manner that the sum and carry from the CSA in the previous cycle are inputted to CSA. Due to this feedback of both sum and carry, the number of inputs to CSA increases, compared to the standard design and [17]. In order to efficiently solve the increase in the amount of data, a CSA architecture is modified to treat the sign bit.

2) *Equation Derivation:* The aforementioned concept is applied to (5) to express the proposed MAC arithmetic. Then, the multiplication would be transferred to a hardware architecture that complies with the proposed concept in which the feedback Value for accumulation will be modified and expanded for the new MAC. If the multiplication in (4) is decomposed, it becomes

$$X \times Y = d_0 2^0 Y + d_1 2^2 Y + d_2 2^4 Y + \dots + d_{N/2-1} 2^{N-2} Y. \quad (6)$$

If (6) is divided into the first partial product, sum of the middle partial products, and the final partial product, it can be re-expressed as (7).

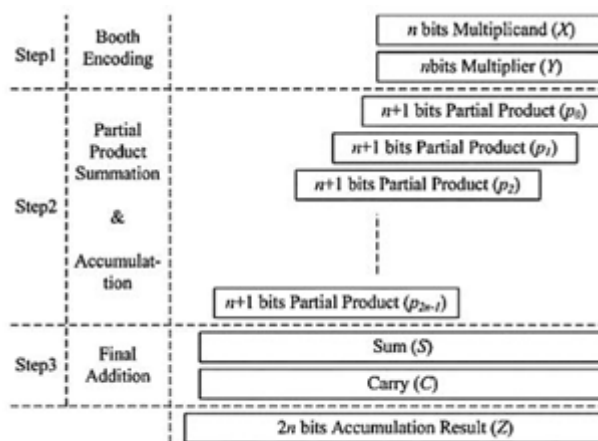


Fig. 3. Proposed arithmetic operation of multiplication and accumulation.

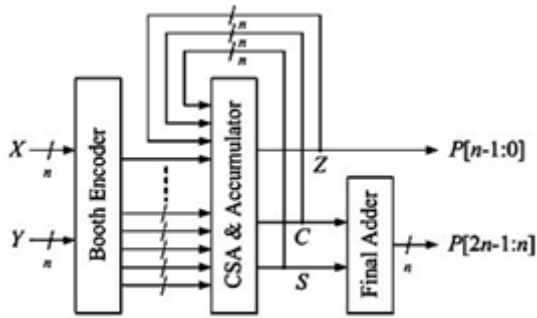


Fig. 4. Hardware architecture of the proposed MAC

The reason for separating the partial product addition as (7) is that three types of data are fed back for accumulation, which are the sum, the carry, and the preadded results of the sum and carry from lower bits. Now, the proposed concept is applied to Z in (5). If Z is first divided into upper and lower bits and rearranged, (8) will be derived. The first term of the right-hand side in (8) corresponds to the upper bits. It is the value that is fed back as the sum and the carry. The second term corresponds to the lower bits.

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=N}^{2N-1} z_i 2^i. \quad (8)$$

The second term can be separated further into the carry term and sum term as

$$\sum_{i=N}^{2N-1} z_i 2^i = \sum_{i=0}^{N-1} z_{N+i} 2^{i+2N} = \sum_{i=0}^{N-2} (c_i + s_i) 2^{i+2N}. \quad (9)$$

Thus, (8) is finally separated into three terms as

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=0}^{N-2} c_i 2^{i+2N} + \sum_{i=0}^{N-2} s_i 2^{i+2N}. \quad (10)$$

If (7) and (10) are used, the MAC arithmetic in (5) can be expressed as

$$P = \left(d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y \right) + \left(\sum_{i=0}^{N-1} z_i 2^{i+2N} + \sum_{i=0}^{N-2} c_i 2^{i+2N} + \sum_{i=0}^{N-2} s_i 2^{i+2N} \right). \quad (11)$$

$$P = \left(d_0 2Y + \sum_{i=0}^{N-1} z_i 2^i \right) + \left(\sum_{i=1}^{N/2-1} d_i 2^{2i} Y + \sum_{i=0}^{N-2} c_i 2^{i+2N} \right) + \left(d_{N/2-1} 2^{N-2} Y + \sum_{i=0}^{N-2} s_i 2^{i+2N} \right). \quad (12)$$

B. Proposed MAC Architecture

If the MAC process proposed in the previous section is rearranged, it would be as Fig. 3, in which the MAC is organized into three steps. When compared with Fig. 1, it is easy to identify the difference that the accumulation has been merged into the process of adding the partial products. Another big difference from Fig. 1 is that the final addition process in step 3 is not always run even though it does not appear explicitly in Fig. 3. Since accumulation is carried out using the result from step 2 instead of that from step 3, step 3 does not have to be run until the point at which the result for the final accumulation is needed.

C. Proposed CSA Architecture

The architecture of the hybrid-type CSA that complies with the operation of the proposed MAC is shown in Fig. 5, which performs 8 x 8-bit operation. It was formed based on (12). In Fig. 5, S_i is to simplify the sign expansion and N_i is to compensate 1's complement number into 2's complement number. $S[i]$ and $C[i]$ correspond to the i th bit of the feedback sum and carry. $Z[i]$ is the i th bit of the sum of the lower bits for each partial product that were added in advance and $Z'[i]$ is the previous result.

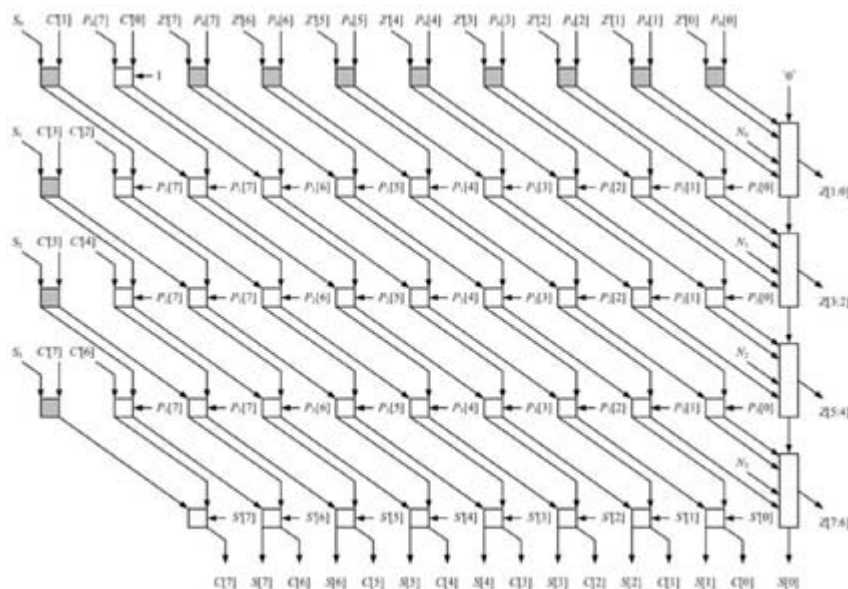


Fig. 5. Architecture of the proposed CSA tree.

4. Implementation and Experiment

In this section, the proposed MAC is implemented and analyzed. Then it would be compared with some previous researches.

A. Hardware Resource

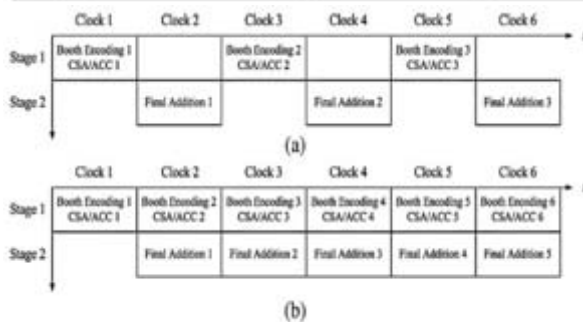
Analysis of Hardware Resource: The three layer architecture mentioned before is analyzed to compare the hardware resources and the results are given in Table II. The hardware resources in Table II are the results from counting all the logic elements for a general 16-bit architecture. The 90 nm CMOS HVT standard cell library from TSMC was used as the hardware library for the 16 bits. As Table II shows, the standard design uses the most hardware resources and the proposed architecture uses the least.

TABLE I
CHARACTERISTICS OF CSA

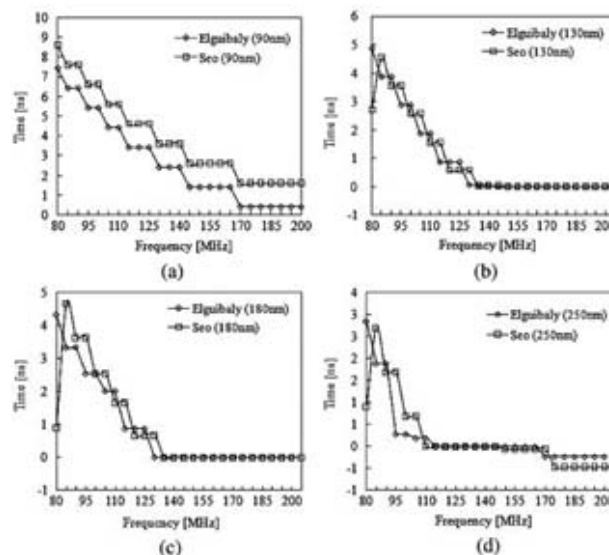
	[6]	[17]	The Proposed
Number System	2's Complement	1's Complement	1's Complement
Sign Extension	Used	Used	Not Used
Accumulation	Result Data of Final Addition	Result Data of Final Addition	Sum and Carry of CSA
CSA Tree	FA, HA	FA, 2 bits CLA	FA, HA, 2 bits CLA
Final Adder	2n bits	(n+2) bits	n bits

TABLE II
CALCULATION OF HADWARE RESOURCE

Component	[6]		[17]		The Proposed	
	General	16 bits	General	16 bits	General	16 bits
FA	$(\frac{n^2}{2} + n)$	964.8	$(\frac{n^2}{2} + 2n + 3)$	1092.1	$(\frac{n^2}{2} + \frac{n}{2})$	911.2
HA	0	0	0	0	$\frac{3n}{2}$	76.8
2 bit CLA	0	0	$(\frac{n}{2} - 1)$	49	$\frac{n}{2}$	56
Accumulator (2n+1) bits CLA		214		-		-
Final adder	2n bits	197	(n+2) bits	109.5	n bits	97
Total		1375.8		1250.6		1141



Pipelined operational sequence. (a) Elgahaly's operation. (b) Proposed operation.



5. Conclusion

In this paper, a new MAC architecture to execute the multiplication-accumulation operation, which is the key operation, for digital signal processing and multimedia information processing efficiently, was proposed. By removing the independent accumulation process that has the largest delay and merging it to the compression process of the partial products, the overall MAC performance has been improved almost twice as much as in the previous work.

References

- [1] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw-Hill, 1984.
- [2] Information Technology-Coding of Moving Picture and Associated Audio, MPEG-2 Draft International Standard, ISO/IEC 13818-1, 2, 3, 1994.
- [3] JPEG 2000 Part I Final Draft, ISO/IEC JTC1/SC29 WG1.
- [4] O. L. MacSorley, "High speed arithmetic in binary computers," Proc. IRE, vol. 49, pp. 67-91, Jan. 1961.
- [5] S. Waser and M. J. Flynn, Introduction to Arithmetic for Digital Systems Designers. New York: Holt, Rinehart and Winston, 1982.
- [6] A. R. Omondi, Computer Arithmetic Systems. Englewood Cliffs, NJ:
- [7] D. Booth, "A signed binary multiplication technique," Quart. J. Math., vol. IV, pp. 236-240, 1952.
- [8] S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron Comput., vol. EC-13, no. 1, pp. 14-17, Feb. 1964.
- [9] R. Cooper, "Parallel architecture modified Booth multiplier," Proc. Inst. Electr. Eng. G, vol. 135, pp. 125-128, 1988.
- [10] N. R. Shanbag and P. Juneja, "Parallel implementation of a 4 X 4-bit multiplier using modified Booth's algorithm," IEEE J. Solid-State Circuits, vol. 23, no. 4, pp. 1010-1013, Aug. 1988.
- [11] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 X 54 regular structured tree multiplier," IEEE J. Solid-State Circuits, vol. 27, no. 9, pp. 1229-1236, Sep. 1992.

- [12] J. Fadavi-Ardekani, "M N Booth encoded multiplier generator using optimized Wallace trees," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 120-125, Jun. 1993.
- [13] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54 54 multiplier using pass-transistor multiplexer," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 251-257, Mar. 1995.
- [14] A. Tawfik, F. Elguibaly, and P. Agathoklis, "New realization and implementation of fixed-point IIR digital filters," *J. Circuits, Syst., Comput.*, vol. 7, no. 3, pp. 191-209, 1997.
- [15] Tawfik, F. Elguibaly, M. N. Fahmi, E. Abdel-Raheem, and P. Agathoklis, "High-speed area-efficient inner-product processor," *Can. J. Electr. Comput. Eng.*, vol. 19, pp. 187-191, 1994.
- [16] F. Elguibaly and A. Rayhan, "Overflow handling in inner-product processors," in *Proc. IEEE Pacific Rim Conf. Commun., Comput., Signal Process.*, Aug. 1997, pp. 117-120.
- [17] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified Booth algorithm," *IEEE Trans. Circuits Syst.*, vol. 27, no. 9, pp. 902-908, Sep. 2000.
- [18] A. Fayed and M. Bayoumi, "A merged multiplier-accumulator for high speed signal processing applications," *Proc. ICASSP*, vol. 3, pp. 3212-3215, 2002.
- [19] P. Zicari, S. Perri, P. Corsonello, and G. Cocorullo, "An optimized adder accumulator for high speed MACs," *Proc. ASICON 2005*, vol. 2, pp. 757-760, 2005.
- [20] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 584-594, Feb. 1990
- [21] Soojin Kim and Kyeongsoon Cho, *Design of High-speed Modified Booth Multipliers Operating at GHz Ranges*, World Academy of Science, Engineering and Technology; pp. 1-4, 6.1. 2010.
- [22] Qingzheng LI, Guixuan LIANG, Amine BERMAK, *A High-speed 32-bit Signed/Unsigned Pipelined Multiplier*, *IEEE 5th International Symposium on Electronic Design, Test & Applications*; pp. 207-211, 2010.
- [23] Rajput, R.P., Swamy, M.N.S, "High Speed Modified Booth Encoder Multiplier for signed and unsigned numbers", *IEEE* page 649-654, 28-30 March 2012.