A New VLSI Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm

Young-Ho Seo, *Member, IEEE*, and Dong-Wook Kim, *Member, IEEE*

***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 carry 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.**

***Index Terms*—Booth multiplier, carry save adder (CSA) tree, computer arithmetic, digital signal processing (DSP), multiplier-and-accumulator (MAC).**

I. Introduction

WITH the recent rapid advances in multimedia and communication systems, real-time signal processings 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 they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetics determines the execution speed 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) [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]. Because Wallace tree is to add the partial products from encoder as parallel as possible, its operation time is proportional to $O(\log_2 N)$, where N is the number of inputs. It uses the fact that counting the number of 1's among the inputs reduces the number of outputs into $\log_2 N$. In real implementation, many (3:2) or (7:3) counters are used to reduce the number of outputs in each pipeline step. 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. To increase the speed of the MBA algorithm, many parallel multiplication architectures have been researched [11]–[13]. Among them, the architectures based on the Baugh–Wooley algorithm (BWA) have been developed and they have been applied to various digital filtering calculations [14]–[16].

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. An 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

Manuscript received June 23, 2008; revised October 14, 2008. First published November 17, 2009; current version published January 20, 2010. This work was supported by the IT R&D program of MKE/IITA. [2009-S-001-01, Signal Processing Elements and their SoC Developments to Realize the Integrated Service System for Interactive Digital Holograms.]

The authors are with Kwangwoon University, Seoul 139-701, Korea.

Digital Object Identifier 10.1109/TVLSI.2008.2009113

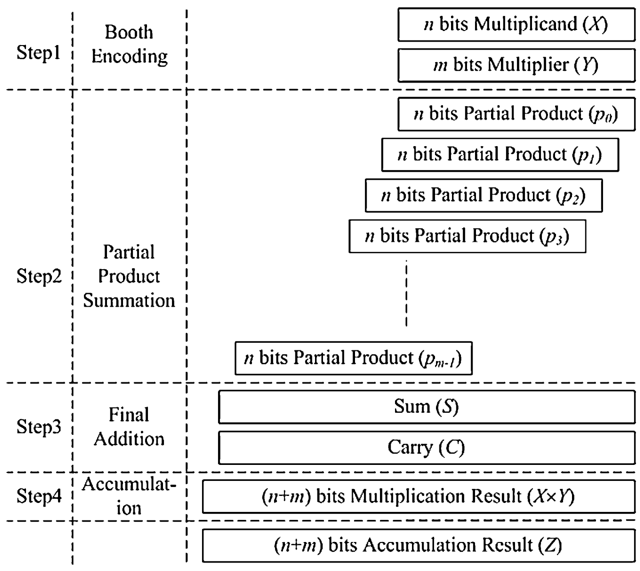

Step1
Booth Encoding
n bits Multiplicand (X)
m bits Multiplier (Y)
Step2
Partial Product Summation
n bits Partial Product (p_0)
n bits Partial Product (p_1)
n bits Partial Product (p_2)
n bits Partial Product (p_3)
n bits Partial Product (p_{m-1})
Step3
Final Addition
Sum (S)
Carry (C)
Step4
Accumulat-ion
(n+m) bits Multiplication Result (X×Y)
(n+m) bits Accumulation Result (Z)

Fig. 1. Basic arithmetic steps of multiplication and accumulation.

also took this compressor as the basic building blocks for the multiplication circuit.

In this paper, a new architecture for a high-speed MAC is proposed. In this MAC, the computations of multiplication and accumulation are combined and a hybrid-type CSA structure is proposed to reduce the critical path and improve the output rate. It uses MBA algorithm based on 1's complement number system. A modified array structure for the sign bits is used to increase the density of the operands. A carry look-ahead adder (CLA) is inserted in the CSA tree to reduce the number of bits in the final adder. In addition, in order to increase the output rate by optimizing the pipeline efficiency, intermediate calculation results are accumulated in the form of sum and carry instead of the final adder outputs.

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.

II. 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.

A 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.

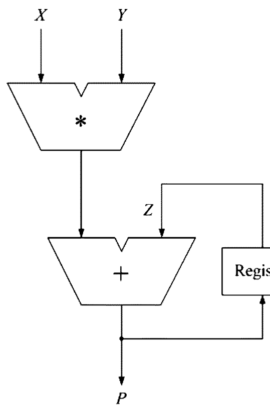


Fig. 2. Hardware architecture of general MAC.

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

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

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].

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

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}. \tag{3}$$

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

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

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

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

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].

If N-bit data are multiplied, the number of the generated partial products is proportional to N. In order to add them serially, the execution time is also proportional to N. The architecture of a multiplier, which is the fastest, uses radix-2 Booth encoding that generates partial products and a Wallace tree based on CSA as the adder array to add the partial products. If radix-2 Booth encoding is used, the number of partial products, i.e., the inputs to the Wallace tree, is reduced to half, resulting in the decrease in CSA tree step. In addition, the signed multiplication based on 2's complement numbers is also possible. Due to these reasons, most current used multipliers adopt the Booth encoding.

III. 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 N-bit numbers and accumulate into a $2N$-bit number is considered, the critical path is determined by the $2N$-bit accumulation operation. If a pipeline scheme is applied for each step in the standard design of Fig. 1, the delay of the last accumulator must be reduced in order to improve the performance of the MAC. 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 carrys 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.

First, if the multiplication in (4) is decomposed and rearranged, it becomes

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

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

$$X \times Y = d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y. \quad (7)$$

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 and is

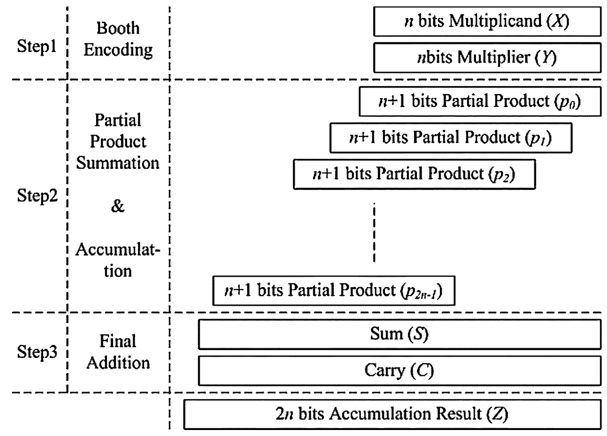

Step1
Booth Encoding
n bits Multiplicand (X)
nbits Multiplier (Y)
Step2
Partial Product Summation
&
Accumulat-tion
n+1 bits Partial Product (p0)
n+1 bits Partial Product (p1)
n+1 bits Partial Product (p2)
n+1 bits Partial Product (p2n-1)
Step3
Final Addition
Sum (S)
Carry (C)
2n bits Accumulation Result (Z)

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

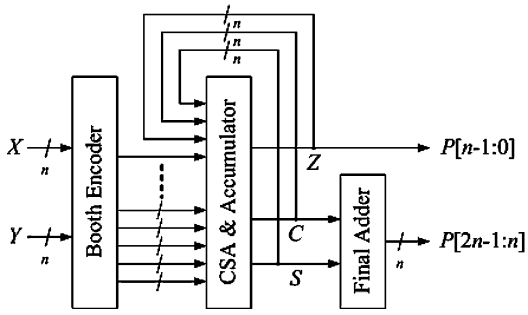


Fig. 4. Hardware architecture of the proposed MAC.

the value that is fed back as the addition result for the sum and carry

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

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

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

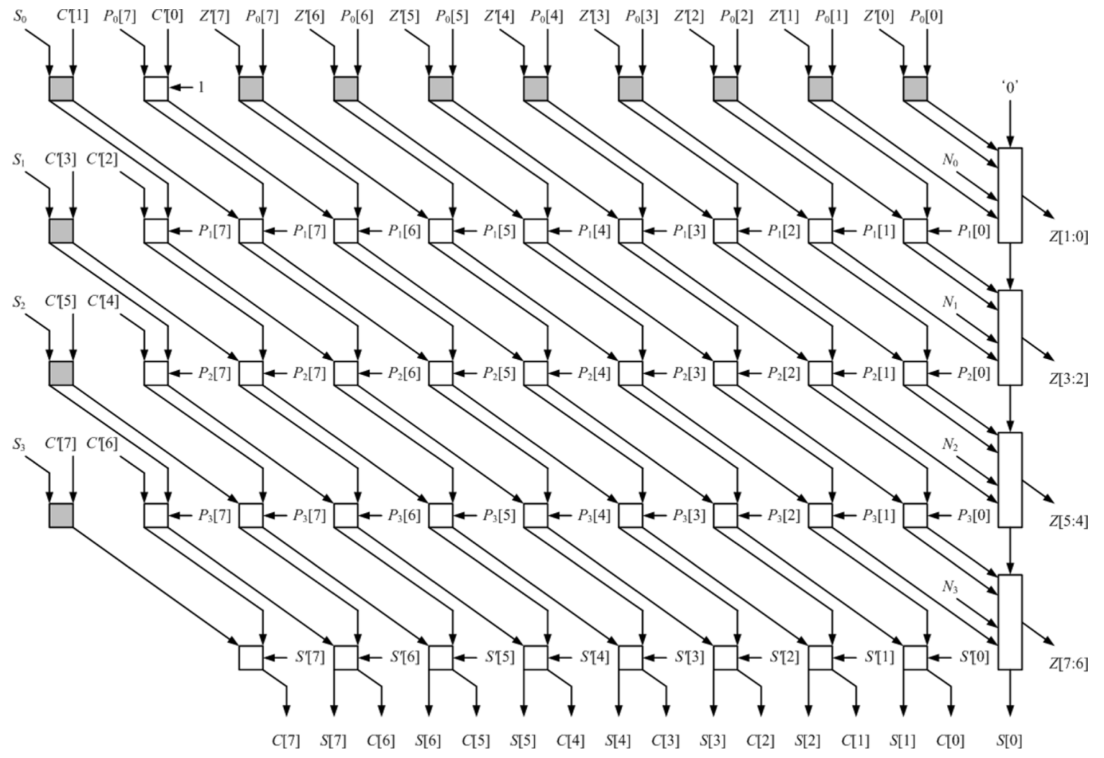
Thus, (8) is finally separated into three terms as

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

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

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

If each term of (11) is matched to the bit position and rearranged, it can be expressed as (12), which is the final equation for the proposed MAC. The first parenthesis on the right is the operation to accumulate the first partial product with the added result of the sum and the carry. The second parenthesis is the



S0 C'[1] P0[7] C'[0] Z'[7] P0[7] Z'[6] P0[6] Z'[5] P0[5] Z'[4] P0[4] Z'[3] P0[3] Z'[2] P0[2] Z'[1] P0[1] Z'[0] P0[0]
1
'0'
N0
S1 C'[3] C'[2]
P1[7] P1[7] P1[6] P1[5] P1[4] P1[3] P1[2] P1[1] P1[0]
Z[1:0]
N1
S2 C'[5] C'[4]
P2[7] P2[7] P2[6] P2[5] P2[4] P2[3] P2[2] P2[1] P2[0]
Z[3:2]
N2
S3 C'[7] C'[6]
P3[7] P3[7] P3[6] P3[5] P3[4] P3[3] P3[2] P3[1] P3[0]
Z[5:4]
N3
S'[7] S'[6] S'[5] S'[4] S'[3] S'[2] S'[1] S'[0]
Z[7:6]
C[7] S[7] C[6] S[6] C[5] S[5] C[4] S[4] C[3] S[3] C[2] S[2] C[1] S[1] C[0] S[0]

Fig. 5. Architecture of the proposed CSA tree.

one to accumulate the middle partial products with the sum of the CSA that was fed back. Finally, the third parenthesis expresses the operation to accumulate the last partial product with the carry of the CSA

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

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.

The hardware architecture of the MAC to satisfy the process in Fig. 3 is shown in Fig. 4. The n-bit MAC inputs, X and Y, are converted into an $(n+1)$-bit partial product by passing through the Booth encoder. In the CSA and accumulator, accumulation is carried out along with the addition of the partial products. As a result, n-bit S, C, and Z (the result from adding the lower bits of the sum and carry) are generated. These three values are fed back and used for the next accumulation. If the final result for the MAC is needed, $P[2n-1:n]$ is generated by adding S and C in the final adder and combined with $P[n-1:0]$ that was already generated.

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 8×8-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 ith bit of the feedback sum and carry. $Z[i]$ is the ith bit of the sum of the lower bits for each partial product that were added in advance and $Z'[i]$ is the previous result. In addition, $P_j[i]$ corresponds to the ith bit of the jth partial product. Since the multiplier is for 8 bits, totally four partial products ($P_0[7:0]\sim P_3[7:0]$) are generated from the Booth encoder. In (11), $d_0 Y$ and $d_{N/2-1}2^{N-2}Y$ correspond to $P_0[7:0]$ and $P_3[7:0]$, respectively. This CSA requires at least four rows of FAs for the four partial products. Thus, totally five FA rows are necessary since one more level of rows are needed for accumulation. For an $n\times n$-bit MAC operation, the level of CSA is $(n/2+1)$. The white square in Fig. 5 represents an FA and the gray square is a half adder (HA). The rectangular symbol with five inputs is a 2-bit CLA with a carry input.

The critical path in this CSA is determined by the 2-bit CLA. It is also possible to use FAs to implement the CSA without CLA. However, if the lower bits of the previously generated partial product are not processed in advance by the CLAs, the number of bits for the final adder will increase. When the entire multiplier or MAC is considered, it degrades the performance.

In Table I, the characteristics of the proposed CSA architecture have been summarized and briefly compared with other architectures. For the number system, the proposed CSA uses 1's

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

TABLE III
GATE SIZE OF LOGIC CIRCUIT ELEMENT

Element	Gate Size
Inverter	0.8
2/3/4-NAND	1/1.5/2.5
2/3/4-NOR	1/2/2.2
2/3/4-XOR	2/4/6
2/3/4-AND	1.2/1.5/2
2/3/4-OR	1.2/1.5/2
Half Adder	3.2
Full Adder	6.7
D Flip-Flop	6.2
4×1 MUX	6
8×1 MUX	14.2
2 bits CLA	7
4 bits CLA	20.5

TABLE IV
ESTIMATION OF GATE SIZE BY SYNTHESIS

	CSA		Booth Encoder	Final Adder		Total (C/L)	
nm	[17]	Proposed		[17]	Proposed	[17]	Proposed
90	1,067	1,009	713	104	97	1,884	1,819
130	1,216	1,158	864	118	110	2,198	2,131
180	1,581	1,484	808	120	114	2,510	2,407
250	2,027	2,001	1,129	141	131	3,297	3,261

complement, but ours uses a modified CSA array without sign extension. The biggest difference between ours and the others is the type of values that is fed back for accumulation. Ours has the smallest number of inputs to the final adder.

IV. IMPLEMENTATION AND EXPERIMENT

In this section, the proposed MAC is implemented and analyzed. Then it would be compared with some previous researches. First, the amount of used resources in implementing in hardware is analyzed theoretically and experimentally, then the delay of the hardware is analyzed by simplifying Sakurai's alpha power law [20]. Finally, the pipeline stage is defined and the performance is analyzed based on this pipelining scheme. Implementation result from each section will be compared with the standard design [6] and Elguibaly's design [17], each of which has the most representative parallel MBA architecture.

A. Hardware Resource

1) Analysis of Hardware Resource: The three architecture mentioned before are analyzed to compare the hardware resources and the results are given in Table II. In calculating the amount of the hardware resources, the resources for Booth encoder is excluded by assuming that the identical ones were used for all the designs. 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. The gate count for each design was obtained by synthesizing the logic elements in an optimal form and the result was generated by multiplying it with the estimated number of hardware resources. The gate counts for the circuit elements obtained through synthesis are shown in Table III, which are based on a two-input NAND gate.

Let us examine the gate count for several elements in Table III first. Since the gate count is 3.2 for HA and 6.7 for FA, FA is about twice as large as HA. Because the gate count for a 2-bit CLA is 7, it is slightly larger than FA. In other words, even if a 2-bit CLA is used to add the lower bits of the partial products in the proposed CSA architecture, it can be seen that the hardware resources will not increase significantly.

As Table II shows, the standard design uses the most hardware resources and the proposed architecture uses the least. The proposed architecture has optimized the resources for the CSA by using both FA and HA. By reducing the number of input bits to the final adder, the gate count of the final adder was reduced from 109.5 in [17] to 97.

2) Gate Count by Synthesis: The proposed MAC and [17] were implemented in register-transfer level (RTL) using hardware description language (HDL). The designed circuits were synthesized using the Design Complier from Synopsys, Inc., and the gate counts for the resulting netlists were measured and summarized in Table IV. The circuits in Table IV are for 16-bit MACs. In order to examine the various circuit characteristics for different CMOS processes, the most popular four process libraries (0.25, 0.18, 0.13 μm, 90 nm) for manufacturing digital semiconductors were used. It can be seen that the finer the process is, the smaller the number of gates is.

As shown in Table II, the gate count for our architecture is slightly smaller than that in [17]. It must be kept in mind that if a circuit is implemented as part of a larger circuit, the number of gates may change depending on the timing for the entire circuit and the electric environments even though identical constraints were applied in the synthesis. The results in Table IV were for the combinational circuits without sequential element. The total gate count is equal to the sum of the Booth encoder, the CSA, and the final adder.

TABLE V
NORMALIZED CAPACITANCE AND GATE DELAY
$(\eta = 2, c = C/C_g, t = 0.1 \times T/\gamma)$

Gate	Comment	C_i	T_g
Inverter	-	3	t+c
8×1 MUX	4-level logic	4	35.2+t+c
D-F/F	Slave delay	4	16.1+t+c
1 bit FA	input-to-sum	12	39.6+t+c
1 bit FA	input-to-carry	12	38.7+t+c
2 bits CLA	input-to-sum	12	64.9+t+c
2 bits CLA	input-to-carry	16	53.9+t+c
4 bits CLA	input-to-sum	12	96.8+t+c
4 bits CLA	input-to-carry	24	88+t+c

TABLE VI
DELAY TIME ANALYSIS AND COMPARISON

Step	[6]		[17]		The Proposed	
	General	16 bits	General	16 bits	General	16 bits
Step1	Booth Encoding $52.8n + 59.9$	904.7	Booth Encoding $10.6n + 81.1$	250.7	Booth Encoding $10.6n + 81.1$	250.7
Step2	CSA $25.95n - 51.9$	363.3	Hybrid CSA $33.55n - 67.1$	469.7	Hybrid CSA $33.55n$	536.8
Step3	Final Addition $57.2n$	915.2	Final Addition $28.6n + 57.2$	514.8	Final Addition $28.6n$	457.6
Step4	Accumulation $57.2n$	915.2	- -	-	- -	-
Critical Path	Accumulation $57.2n$	915.2	Final Addition $28.6n + 57.2$	514.8	Hybrid CSA $33.55n$	536.8

B. Delay Model

1) Modeling: In this paper, Sakurai's alpha power law [20] is used to estimate the delay. Because CMOS process is used and the interconnect delay that is not due to gates related to logic operation is ignored, $\alpha = 1$ was used. The delay by simplifying the alpha power law was modeled in [17]. order for easy comparisons with other architectures, the modeled values identical to [17] are used in this paper. The normalized input capacitance (C_i) and gate delay (T_d) for the hardware building blocks with these modeled values are shown in Table V.

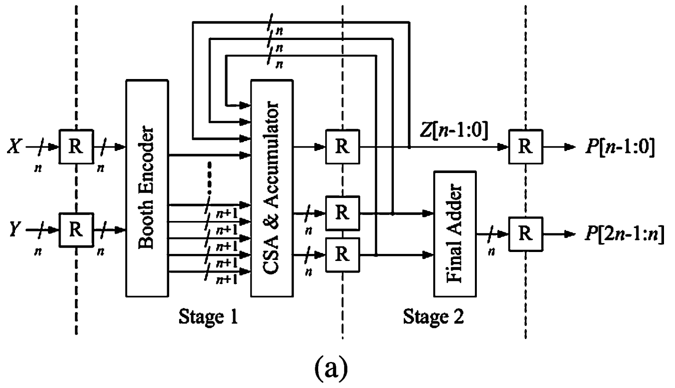
In Table II, η is the ratio of the saturation velocity. C and C_g are the load gate capacitance and gate capacitance of the minimum-area transistor, respectively. T is the duration time and γ is the falling time of the minimum-area inverter due to C_g. Since delay modeling and its simplification process is not the focus of this paper, it will not be described in detail here. For additional description, refer to [17] and [20].

2) Delay Analysis: The results of delay modeling for the Booth encoder (T_b), the CSA (T_c), and the final adder (T_f) using Table VI and [17] and [20] are given in (13)–(16). In (13), T_s, T_p, and T_m represent the select logic delay, buffer delay, and MUX delay, respectively

$$T_b = T_s + (n+2)T_p + T_m \tag{13}$$

$$T_b = 12.3 + (n+2) \times 10.6 + 47.6 = 10.6n + 81.1 \tag{14}$$

$$T_c = \left(\frac{n}{2}\right) T_2(\text{carry}) = \left(\frac{n}{2}\right) 67.1 + 33.5n \tag{15}$$



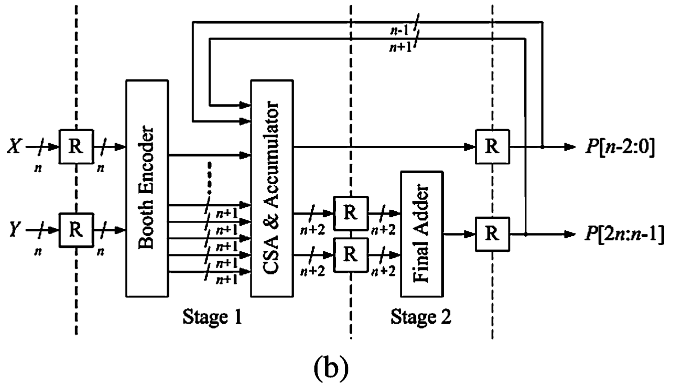


Fig. 6. Pipelined hardware structure. (a) Proposed structure. (b) Elguibaly's structure.

$$T_f = \left(\frac{n}{4}\right) T_4(\text{carry}) = 28.6n. \tag{16}$$

The delays in Table VI were obtained using the hardware resources in Table II and the gate delays in Table V. From Table VI, it is easily recognizable that the delay of [6] is considerably larger than others. The proposed architecture uses the same Booth encoder as in [17] and the delay is also identical to $10.6n + 81.1$. Because the critical path for the CSA tree is determined by the 2-bit CLA, the delay is proportional to it. The proposed architecture has one more 2-bit CLA compared to [17], as shown in Table II where the delay is greater by 67.1. The number of input bits for the final adder is less by one in our architecture and the delay is also faster by 57.2.

If pipelining is applied for each step, the critical path for the proposed architecture is $33.55n$ and it corresponds to the value of 536.8 for 16-bit MAC. If clock speed is simply considered, the characteristic for the proposed architecture may seem inferior to [17]. However, if the performance of the actual output rate is considered, it can be verified that the proposed architecture is superior. The reason will be explained in detail in the next section with the pipelining scheme.

In addition, we compared the proposed architecture with that of [18]. Because of the difficulties in comparing other factors, only delay is compared. The sizes of both MACs were 8×8 bits and implemented by a 0.35 μm fabrication process. The delay of ours was 3.94 ns, while in [18], it was it 4.26 ns, which means that ours improved about 7.5% of the speed performance. This improvement is mainly due to the final adder. The architecture from [18] should include a final adder with the size of $2n$ to perform an $n \times n$ multiplication. It means that the operational bottleneck is induced in the final adder no matter how much delays

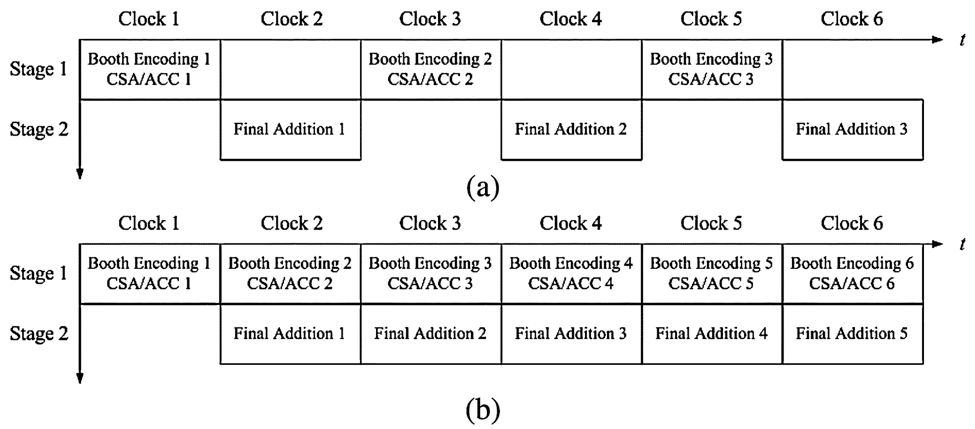

Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
t
Stage 1
Stage 2
Booth Encoding 1 CSA/ACC 1
Booth Encoding 2 CSA/ACC 2
Booth Encoding 3 CSA/ACC 3
Final Addition 1
Final Addition 2
Final Addition 3
(a)
Booth Encoding 4 CSA/ACC 4
Booth Encoding 5 CSA/ACC 5
Booth Encoding 6 CSA/ACC 6
Final Addition 4
Final Addition 5
(b)

Fig. 7. Pipelined operational sequence. (a) Elguibaly's operation. (b) Proposed operation.

TABLE VII
PIPELINE STAGE

Stage	[6]	[17]	The Proposed
Stage1	$139.95n + 87.1$	$44.15n + 14$	$44.15n + 81.1$
Stage2	$57.2n + 28.5$	$28.6n + 57.2$	$28.6n$

TABLE VIII
PIPELINE AND PERFORMANCE ANALYSIS

Item	[6]	[17]	The Proposed
Output Rate	1 clock	2 clocks	1 clock
Pipeline Delay	$139.95n + 87.1$	$88.3n + 28$	$44.15n + 81.1$

are reduced in the multiplication or accumulation step, which is the general problem in designing a multiplier. However, our design uses $(n-2)$-bit final adder, which causes the speed improvement. This improvement is getting bigger as the number of input bits increases.

C. Pipelining

1) Stage Analysis: The pipeline stages were determined based on the delay modeling obtained earlier. step 1 and step 2 in Fig. 3 that correspond to the Booth encoding and CSA operation, respectively, are set to stage 1 and step 3, which correspond to the final adder and are set to stage 2. Such pipeline stage can be organized as shown in Table VII and the clock frequency is determined by this result.

In Table VII, it can be seen that CSA can operate at a higher clock rate in [17] compared to the proposed architecture. However, it does not mean that the overall MAC performance is better. The reason why the proposed architecture has slightly higher delay and hardware resources is that the focus of ours has been on the overall performance. This will be examined in detail in the next section.

2) Pipeline Structure and Operation: A hardware incorporates a pipelining scheme to increase the operation speed and ours did too, which is shown in Fig. 6(a), with the one from Elguibaly's scheme [17] in Fig. 6(b) for the purpose of comparison. The difference between the two is because ours carries out the accumulation by feeding back the final CSA outputs rather than the final adder results as in Fig. 6(b).

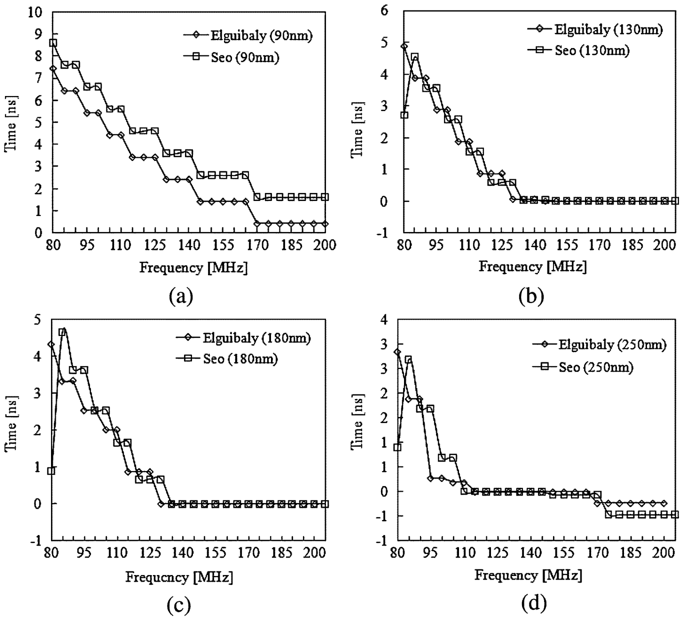


Fig. 8. Timing analysis of the synthesized circuits. (a) 90 nm. (b) 0.13 μm. (c) 0.18 μm. (d) 0.25 μm.

These two schemes are also compared in the time sequence in Fig. 7(a) and (b) for Fig. 6(a) and (b), respectively. While an accumulated result cannot be output by the method in [17] every clock period because of a structural drawback for the accumulation, ours can output a result in every clock cycle. Thus, even though our delay is a little longer than [17], as shown in Table VII or Table VIII, ours shows much better overall performance or the output rate.

3) Timing Analysis: After synthesizing using 0.25, 0.18, 0.13, and 0.09 μm processes, static timing analyses (STAs) were performed and the results are shown in Fig. 8 graphically. This result is an important result for the physical synthesis and placement and routing (P & R) process in actual chip production. In this figure, the frequencies in x axis mean the target frequencies of the constraints imposed in synthesis and the times in y axis are the timing margins (slacks), i.e., we observed the timing margins increasing target frequency from 80 to 200 MHz.

The finer the process is, the more timing margin (slack) for STA the proposed MAC needs compared to [17]. Especially for the 90 nm process, the design compiler can execute more detailed and precise synthesis and optimization if the data path and structure for the designed circuit are more structural and regular. This is because it becomes easier for the design compiler to repeat the process of mapping various cells and carrying out STA in order to generate good circuit satisfying the conditions in the constraint. It requires more diverse driving forces to drive the standard cells. Fig. 8(d) has the biggest slack time that is over 20%. In general, if the correlation between EDA tools used during the back-end process is assumed to be 5% and a timing margin is greater than +2 ns, further optimization is not needed for the later physical synthesis and P&R process. It can also easily overcome the routing congestion that is a very frequently occurring problem.

If the STA result is considered with synthesis, it can be concluded that the proposed architecture is very structural and regular.

V. 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.

The proposed hardware was implemented and synthesized through four types of CMOS processes. When examination is based on theoretical and experimental results, the proposed MAC required the hardware resources as much as the previous research. The delay was modeled using Sakurai's alpha power law. While the delay has been increased slightly compared to the previous research, actual performance has been increased to about twice if the pipeline is incorporated.

Consequently, we can expect that the proposed architecture can be used effectively in the area requiring high throughput such as a real-time digital signal processing.

References

[1] J. J. F. Cavanagh, *Digital Computer Arithmetic*. New York: McGraw-Hill, 1984.
[2] *Information Technology-Coding of Moving Picture and Associated Autio, MPEG-2 Draft International Standard*, ISO/IEC 13818-1, 2, 3, 1994.
[3] *JPEG 2000 Part I Fina1119l 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: Prentice-Hall, 1994.
[7] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Math.*, vol. IV, pp. 236–240, 1952.
[8] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[9] A. 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 4×4-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 54×54 regular structured tree multiplier," *IEEE J. Solid-State Circuits*, vol. 27, no. 9, pp. 1229–1236, Sep. 1992.
[12] J. Fadavi-Ardekani, "$M \times N$ 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] A. 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.

Young-Ho Seo (M'05) received the M.S. and Ph.D. degrees from the Department of Electronic Materials Engineering, Kwangwoon University, Seoul, Korea, in 2000 and 2004, respectively.

From 2003 to 2004, he was a Researcher at Korea Electrotechnology Research Institute (KERI). He was also a Research Professor in the Department of Electronic and Information Engineering, Yuhan College, Buchon, Korea. He was an Assistant Professor in the Department of Information and Communication Engineering, Hansung University, Seoul. He is currently an Assistant Professor in the Division of General Education, Kwangwoon University. His current research interests include 2-D/3-D digital image processing, system-on-a-chip design, and contents security.

Dong-Wook Kim (S'82–M'85) received the B.S. and M.S. degrees from the Department of Electronic Engineering, Hangyang University, Seoul, Korea, in 1983 and 1985, respectively, and the Ph.D. degree from the Department of Electrical Engineering, Georgia Institute of Technology, Atlanta, in 1991.

He is currently a Professor and the Dean of Academic Affairs at Kwangwoon University, Seoul. His current research interests include digital system design, digital testability and design-for-test, digital embedded systems for wired and wireless communication, and design of digital signal processors.