Parallel multiplier-accumulator based on radix-2 modified Booth algorithm by using a VLSI architecture

Baile Shruthi  
M Tech Student  
JNIT, Hyderabad.

K Venkateswarlu, M Tech,  
Associate Professor,  
JNIT, Hyderabad.

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 delayed 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 modelling. 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.

I.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 transforms (DWT) [3]. Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic 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 , where 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. 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
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-purposed 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 /2-bit adders instead of one -bit adder to accumulate the -bit MAC results. Recently, Zicari proposed an architecture that took an merging technique to fully utilize the 4–2 compressor [19]. It 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 lookahead 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.

![Fig.1. Basic arithmetic steps of multiplication and accumulation.](image-url)
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].

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

I) 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 must be compressed into a sum and a carry by CSA. The number of bits of sums and carries 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

\[ X \times Y = d_02^Y + d_12^2Y + d_22^3Y + \ldots + d_{N/2-1}2^{N-2}Y \quad (6) \]

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

\[ X \times Y = d_02^Y + \sum_{i=1}^{N/2-2} d_i2^i2^Y + d_{N/2-1}2^{N-2}Y \quad (7) \]

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 the 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.

\[ Z = \sum_{i=0}^{N-1} z_i2^i + \sum_{i=N}^{2N-1} z_i2^i \quad (8) \]

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

\[ \sum_{i=N}^{2N-1} z_i2^i = \sum_{i=0}^{N-1} z_{N+i}2^{i+2^N} = \sum_{i=0}^{N-2} (c_i + s_i)2^i2^N \quad (9) \]

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

\[ Z = \sum_{i=0}^{N-1} z_i2^i + \sum_{i=0}^{N-2} c_i2^i2^N + \sum_{i=0}^{N-2} s_i2^i2^N \quad (10) \]

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

\[
P = \left( d_02^Y + \sum_{i=1}^{N/2-2} d_i2^i2^Y + d_{N/2-1}2^{N-2}Y \right) \\
+ \left( \sum_{i=0}^{N-1} z_i2^i2^Y + \sum_{i=0}^{N-2} c_i2^i2^N + \sum_{i=0}^{N-2} s_i2^i2^N \right) 
\]

(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 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.

\[
P = \left( d_02^Y + \sum_{i=0}^{N-1} z_i2^i \right) + \left( \sum_{i=1}^{N/2-1} d_i2^i2^Y + \sum_{i=0}^{N-2} c_i2^i2^N \right) \\
+ \left( d_{N/2-1}2^{N-2}Y + \sum_{i=0}^{N-2} s_i2^i2^N \right) 
\]

(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 S, C, and Z accumulators, accumulation is carried out along with the addition of the partial products. As a result, [2n-1:n]bit , and (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, is generated by adding S and C in the final adder and combined with $P[n-1:0]$ that was already generated.

![Diagram showing the proposed arithmetic operation of multiplication and accumulation](image1)

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

These three values are fed back and used for the next accumulation. If the final result for the MAC is needed, is generated by adding S and C in the final adder and combined with $P[n-1:0]$ that was already generated.

![Diagram showing the hardware architecture of the proposed MAC](image2)

**Fig. 4.** Hardware architecture of the proposed MAC.

**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 the 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.

![Diagram showing the architecture of the proposed CSA tree](image3)

**Fig. 5.** Architecture of the proposed CSA tree.

### IV. SYNTHESIS AND SIMULATION RESULTS

We have coded the proposed MAC in Verilog HDL using the proposed design and the existing MAC designs of [6] and [7] for bit-widths 16. All the designs are synthesized in the Xilinx Synthesis XST Tool and Simulated using Xilinx ISE simulator. The synthesis and simulation results are shown in Fig.6 and Fig.7.
Figure 6: RTL schematic of 16x16 MAC unit.

Figure 7: Simulated results of 16x16 MAC unit.

The design summary of the proposed 16x16 MAC unit.

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.
REFERENCES


