IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 4, 2020, accepted April 24, 2020, date of publication May 4, 2020, date of current version May 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992286

A High-Performance Multiply-Accumulate Unit by Integrating Additions and Accumulations Into Partial Product Reduction Process

CHE-WEI TUNG AND SHIH-HSU HUANG, (Senior Member, IEEE)
Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan City 32023, Taiwan
Corresponding author: Shih-Hsu Huang (shhuang@cycu.edu.tw)

This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under Grant 107-2218-E-033-007.

ABSTRACT In this paper, we propose a low-power high-speed pipeline multiply-accumulate (MAC) architecture. In a conventional MAC, carry propagations of additions (including additions in multiplications and additions in accumulations) often lead to large power consumption and large path delay. To resolve this problem, we integrate a part of additions into the partial product reduction (PPR) process. In the proposed MAC architecture, the addition and accumulation of higher significance bits are not performed until the PPR process of the next multiplication. To correctly deal with the overflow in the PPR process, a small-size adder is designed to accumulate the total number of carries. Compared with previous works, experimental results show that the proposed MAC architecture can greatly reduce both power consumption and circuit area under the same timing constraint.

INDEX TERMS Digital circuits, logic circuits, multiplying circuits, pipeline processing, power dissipation.

I. INTRODUCTION

The multiply-accumulate (MAC) unit is a fundamental block for digital signal processing (DSP) applications [1]. Especially, in recent years, the development of real-time edge applications has become a design trend [2], [3]. Thus, there is a strong demand for high-speed low-power MAC units.

A conventional MAC unit is composed of two individual blocks: a multiplier and an accumulator (i.e., an accumulate adder). An N-bit MAC unit includes an N-bit multiplier and a $(2N+\alpha-1)$-bit accumulator (adder), where α is the number of guard bits used to avoid overflow (caused by long sequences of multiply-accumulate operations). A lot of previous works [4]–[12] paid attention to the optimization of multiplier and the optimization of adder, respectively.

A multiplier [4]–[7] usually has three steps. The first step is the partial product generation (PPG) process. For example, AND gates can be used to generate a partial product matrix (PPM) for an unsigned multiplication. The second step is the partial product reduction (PPR) process. By using the Dadda tree approach [4], [5], [7] or the Wallace tree approach [4-6], the PPM can be reduced to become two rows. The third step is the final addition. An adder (called the final adder) is used to perform the summation of the final two rows. For an N-bit multiplier, a $(2N-1)$-bit adder is required for the final addition.

Various adder architectures [8]–[10] have been proposed for the trade-offs among delay, area, and power. Furthermore, various MAC unit models can be developed by replacing the multiplier as well as the accumulator (adder) with various architectures. Comparisons on delay, area and power among different MAC unit models are reported in [11], [12].

In a conventional MAC unit, it is necessary to perform two carry propagations: additions in multiplications and additions in accumulations. Note that the carry propagation is time consuming. Therefore, in [13], the multiplier output is fed back to the input of the PPR process. Since the accumulation is handled by the final adder, only one carry propagation is required. Moreover, based on this architecture [13], the area of 16-bit MAC unit can be further reduced 3.2% by fully utilizing the compression [14].

In [13], [14], they do not discuss how to accommodate guard bits in their designs. Ercegovac and Lang [15] add an extra circuit to the final adder for handling guard bits. Owing to this extra circuit, the carry propagation in the final adder

The associate editor coordinating the review of this manuscript and approving it for publication was Gian Domenico Licciardo.

becomes longer. Another drawback of this architecture [15] is that it only supports sign-magnitude numbers.

Different from those previous works [13]–[15] that handle the accumulation in the final adder, Hoang *et al.* [16] use a carry-save adder to implement the final adder. In [16], a carry-save format (one sum vector and one carry vector) is sent to the accumulator without being added to only one vector. Although this architecture [16] can remove the carry propagation in the final adder, it requires a $(2N+\alpha-1)$-bit accumulator. Besides, it is also noteworthy to mention that the concept of this architecture [16] has been used in modern floating-point fused multiply-add (FMA) designs [17], [18].

Based on the architecture proposed by Hoang *et al.* [16], some attentions [19], [20] have been paid to the compressor design for the PPR process. For example, by using the compressor proposed in [20], the number of LUTs for 8-bit MAC unit can be reduced 21.3% in a Virtex 7 FPGA platform.

In this paper, we propose a novel MAC architecture for high performance. In order to reduce critical path delays and power dissipations (caused by carry propagations), our basic idea is to integrate a part of additions (including a part of the final addition in the multiplication and a part of the addition in the accumulation) into the PPR process. In the proposed MAC unit, the final addition of higher significance bits is not performed in the current multiplication. Instead, the final addition and accumulation of higher significance bits are performed in the PPR process of the next multiplication. As a result, the lengths of carry propagations can be greatly reduced. Moreover, to correctly deal with the overflow during the PPR process, an α-bit accumulator (adder) is designed to count the total number of carries, where α is the number of guard bits. Experimental results consistently show that the proposed approach works well in practice.

The proposed MAC unit is a two-stage (i.e., two-cycle) pipeline design. The first stage performs the PPG process, the PPR process, a part of the final addition and the α-bit addition (for handling overflow). The second stage performs an addition to produce the accumulation result. Note that, for power saving, the second stage can only be executed in the last cycle (of the entire sequence of multiply-accumulate operations) by applying the gating technique. For an N-bit MAC unit, the main differences between the proposed architecture and the conventional architecture are below.

- Final addition in the multiplication. The conventional architecture requires a *(2N-1)*-bit adder. On the other hand, the proposed architecture only requires a *(2N-k-1)*-bit adder, where k denotes the number of higher significance bits whose additions (accumulation) are not performed in the final addition.
- Accumulation in the MAC unit. The conventional architecture requires a $(2N+\alpha-1)$-bit adder. On the other hand, the proposed architecture only requires a $(k+\alpha)$-bit adder. Moreover, by applying the gating technique, the $(k+\alpha)$-bit adder can only be executed in the last cycle.

It is noteworthy to mention that the time-consuming carry propagation is also a challenging issue in the modular multiplication [21], [22], [23]. Thus, some high-radix, scalable, and signed digit multipliers [21], [22], [23] have been proposed for modular multiplication. Different from those previous works [21], [22], [23], the proposed MAC unit is designed for standard arithmetic (instead of modular arithmetic).

The remainder of this paper is organized as follows. In Section II, we present the motivation. The architecture of the proposed MAC unit is described in Section III. Then, in Section IV, we use two examples, including an example for the multiplication of two unsigned numbers and an example for the multiplication of two 2's complement numbers, to demonstrate the detailed process of the proposed MAC unit. In Section V, we report the detailed experimental results. In Section VI, we make an extension to the application of a systolic array. Finally, some concluding remarks are given in Section VII.

II. MOTIVATION

In a conventional MAC unit, carry propagations of additions (including final additions in multiplications and additions in accumulations) often result in large power consumption and large path delay. To resolve this problem, we are motived to reduce the lengths of carry propagations in the final addition and the accumulation. Our basic idea is to integrate a part of additions (including a part of the final addition and a part of the accumulation) into the PPR process. As a result, the lengths of carry propagations can be reduced.

In the proposed MAC architecture, to reduce critical path delays (caused by the carry propagations), the addition and accumulation in higher significance bits are not performed until the PPR process of next multiplication. In other words, our PPM (for the PPR process) consists of two PPMs: one PPM is derived by the PPG and the other PPM is derived by the accumulation.

Here we use our 4-bit MAC (displayed in Fig. 1) for illustration. As shown in Fig. 1(a), our PPM is formed by two PPMs: one PPM is from the PPG and the other PPM is from the accumulation. Then, as shown in Fig. 1(b), we use the Dadda tree approach to reduce our PPM to two rows.

In the final addition process, we only use an *(2N-k-1)*-bit adder for the addition of the two rows (obtained by the Dadda tree approach), where N is the number of bits of each input and k represents the number of higher significance bits whose additions (accumulation) are not performed. Since this example is a 4-bit MAC, we have $N = 4$. Moreover, in this example, we assume that $k = 3$. Thus, as shown in Fig. 1(b), we only need to use a 4-bit adder for the final addition. Note that the addition (accumulation) in higher significance bits is not performed.

As displayed in Fig. 1(b), the two product terms in the column of the highest significance bit have exceeded the PPM width. Here we adopt the concept of [24] to handle overflow. The values of these two product terms are sent to an α-bit adder (accumulator) for counting the total number

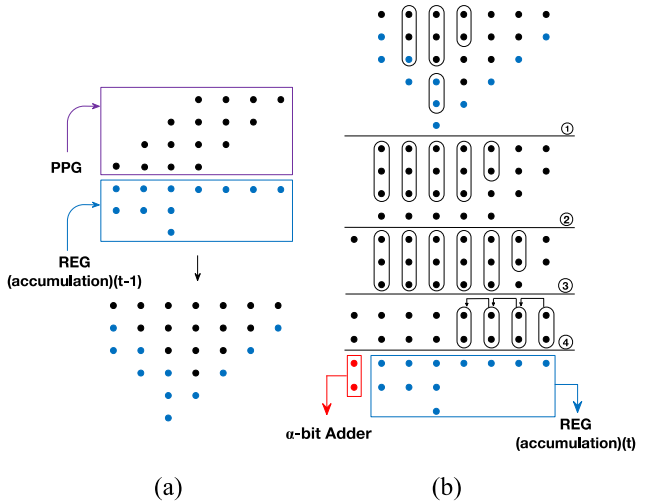

FIGURE 1. Our 4-bit MAC. (a) Our PPM (b) Our PPR process.

of '1' (i.e., the total number of carries) appearing in this column during the entire sequence of multiply-accumulate operations. On the other hand, as shown in Fig. 1(b), the PPM formed by the product terms in other columns (including the addition result of the (2*N*-*k*-1)-bit adder) are stored in register *accumulation*. Note that the result of register *accumulation* (i.e., the PPM formed by the current accumulation) will be combined with the next PPG result to form the next PPM.

III. PROPOSED ARCHITECTURE

In this section, we present the proposed two-stage (i.e., two-cycle) MAC architecture. The first stage performs the PPG process, the PPR process (based on the PPM that combines the PPG result and the accumulation result), the *(2N-k-1)*-bit addition (i.e., a part of the final addition) and the α-bit addition (for dealing with the overflow in the PPR process). Then, the second stage performs the $(k + \alpha)$-bit addition to produce the accumulation result.

The main features of the proposed architecture are below.

- To reduce the lengths of carry propagations, we integrate a part of additions into the PPR process.
- To handle overflow in the PPR process, an α-bit adder is used to count the total number of carries.
- By applying the gating technique, the second stage can only be executed in the last cycle (of the entire sequence of multiply-accumulate operations) for power saving.

The proposed two-stage pipeline MAC unit is displayed in Fig. 2. Our PPM (for the PPR process) is composed of two PPMs: one PPM is derived by the PPG and the other PPM is derived by the accumulation.

For an unsigned MAC unit, in the PPG process, "AND" gates can be directly used to generate the PPM. For a signed MAC unit, because the influences of the sign bit should be taken into account, several PPG algorithms [25]–[27] have been proposed to generate the signed PPM. In the proposed architecture, the Baugh-Wooley algorithm [25], [26] is adopted in the PPG process to generate the signed PPM.

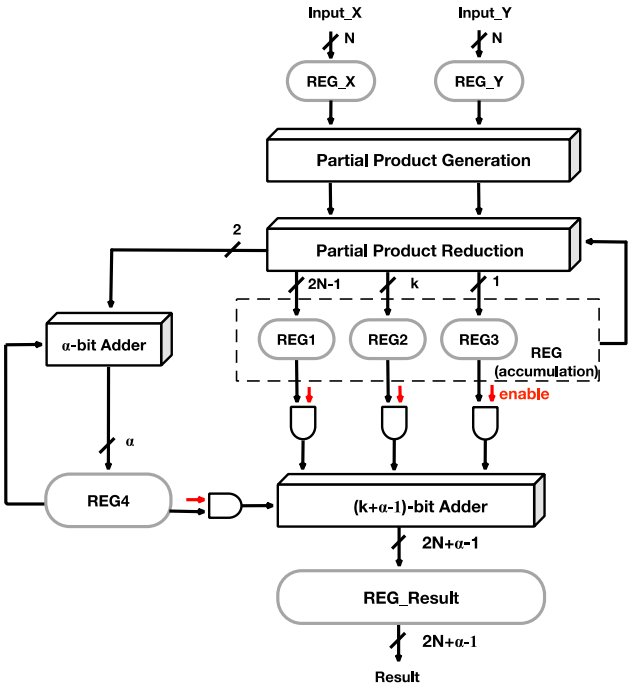

FIGURE 2. The proposed MAC architecture.

In the first stage of the proposed MAC unit, we only perform the *(2N-k-1)*-bit final addition. In other words, the final addition of higher significance bits is not performed. Register *accumulation* is used to store the PPM derived by the accumulation (i.e., the result after the PPR process and the *(2N-k-1)*-bit final addition). Thus, register *accumulation* includes three parts: REG1 (i.e., the first row) has *2N-1* bits, REG2 (i.e., the second row) has *k* bits, which can define by the user, and REG3 (i.e., the third row) has *1* bit. Using our 4-bit MAC shown in Fig. 1(b) as an example, REG1 has 7 bits, REG2 has 3 bits, and REG3 has 1 bit. Note that, the initial value of each bit in register *accumulation* is '0'. In each cycle, the result of register *accumulation* will be combined with the PPG result to form our PPM for the PPR process.

In the PPR process, we adopt the Dadda tree approach [4], [5], [7] to reduce our PPM to two rows. The main reason is that our PPM is not a conventional PPM. With an analysis, we find that, compared with the Wallace tree approach [4]–[6], the Dadda tree approach can use fewer counters for our PPM. After our PPM is reduced to be two rows, we perform the *(2N-k-1)*-bit final addition. Consequently, as shown in Fig. 1(b), a three-row PPM is obtained. Note that this three-row PPM will be stored in register *accumulation*. Then, in the next cycle, the result of register *accumulation* will be combined with the PPG result to form the next PPM for the PPR process.

Since we use an *(2N-k-1)*-bit adder for the addition of the last two rows obtained by the Dadda tree approach, a larger *k* can have a smaller carry propagation in the *(2N-k-1)*-bit adder. However, since the final addition and accumulation of

k higher significance bits are performed in the PPR process of the next multiplication, a larger k results in a larger PPM for the PPR process. In theory, the value of k can be in the range from 1 to *2N-1*. The choice of the value of k depends on given design constraints (e.g., timing constraints or area constraints). However, in our experiences, the best value of k (with respect to given design constraints) is often near to the value of N.

When our PPM is reduced to be two rows, there are two product terms in the column of the highest significance bit. Note that this column (i.e., the column of the highest significance bit) has exceeded the width of our PPM. Owing to the limitation on the width of our PPM, these two product terms will not be used to form the next PPM. Instead, the values of these two product terms will be sent to the α-bit adder (accumulator) for handling the overflow. Note that a translation circuit is required to translate the values of these two product terms to be a corresponding input value of the α-bit adder.

The inputs to the translation circuit are the values of these two product terms in the column of the highest significance bit. Note that the unsigned MAC unit and the signed MAC unit should have different translation circuits (i.e., different translation functions). The functions of the two translation circuits are below (without loss of generality and for the convenience of presentation, here we assume that $\alpha = 2$ and then express a decimal number as a two-bit binary number).

- The translation circuit of the unsigned MAC unit. Since the two input values correspond to the least significant bit of the α-bit adder, the output is equal to the sum of the two input values. Thus, if both the two input values are 0, the corresponding output value is decimal 0 (i.e., binary 00); if one input value is 0 and the other input value is 1, the corresponding output value is decimal 1 (i.e., binary 01); if both the two input values are 1, the corresponding output value is decimal 2 (i.e., binary 10).
- The translation circuit of the signed MAC unit. The two input values still correspond to the least significant bit of the α-bit adder. However, owing to the Baugh-Wooley algorithm [25], 26], the output is equal to the sum of the two input values minus 1. A more detailed explanation is given in the Appendix. Thus, if both the two input values are 0, the corresponding output value is decimal -1 (i.e., binary 11); if one input value is 0 and the other input value is 1, the corresponding output value is decimal 0 (i.e., binary 00); if both the two input values are 1, the corresponding output value is decimal 1 (i.e., binary 01).

The accumulation result of the α-bit adder (accumulator) is stored in register REG4. Note that the initial value of each bit in register REG4 is '0'. The α-bit adder (accumulator) has two inputs: one is from the result of register REG4 and the other is from the result of the translation circuit (note that the translation circuit is a circuit that translates the two product terms in the column of the highest significance bit to a corresponding value).

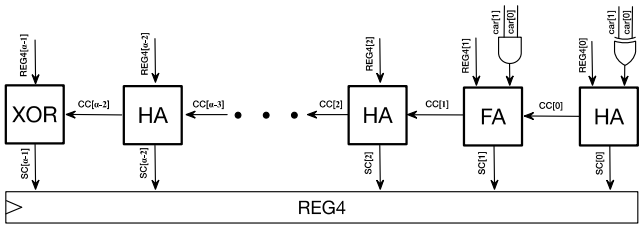

FIGURE 3. The mechanism of α-bit addition in the unsigned MAC unit.

In Fig. 3, we use the circuit that handles the mechanism of α-bit addition (accumulation) in the unsigned MAC unit for illustration. Note that this circuit is applicable to any α value. Inputs car [0] and car [1] denote the two product terms in the column of the highest significance bit. As displayed in Fig. 3, we only need to use one AND gate and one XOR gate to implement the translation circuit. In the addition of each bit i, where $i = 0, 1, 2, \ldots\ldots, \alpha - 1$, SC[$i$] denotes the sum bit and CC[i] denotes the carry bit. Thus, the value of SC[i] will be stored in register REG4.

In the second stage of the proposed MAC unit, we produce the accumulation result. The inputs of the second stage include register *accumulation* (consisting of REG1, REG2 and REG3) and register REG4. Although the total number of columns is *(2N+α-1)*, since each column in the *(2N-k-1)* rightmost columns has only one product term, we only need to use a $(k+\alpha)$-bit adder to perform the addition of the $(k + \alpha)$ leftmost columns.

In the proposed MAC unit, the accumulation has been done in both the α-bit addition and the next PPR process. Thus, if we only need to obtain the final result in the last cycle, we can disable the $(k + \alpha)$-bit addition in other cycles for power saving. As shown in Fig. 2, we use AND gate with an *enable* signal to disable the $(k + \alpha)$-bit addition. The *enable* signal will be '1' the last cycle and '0' in other cycles. As a result, with the gating technique, the second stage will only be executed in the last cycle.

IV. EXAMPLES

In this section, we give two examples. In the first example, we describe the behaviors of the unsigned MAC unit. In the second example, we describe the behavior of the signed MAC unit. Note that the only differences between the unsigned MAC unit and the signed MAC unit are the PPM structure and the α-bit addition mechanism.

The first example is given in Fig. 4. We use the case $13 \times 13 + 12 \times 15$ to demonstrate the detailed process of the unsigned 4-bit MAC unit. In Fig. 4, the values displayed in blue are the results of register accumulation (including REG1, REG2, and REG3). The values displayed in red are the input of α-bit adder (i.e., the output of the translation circuit). Here we assume that $\alpha = 2$. The values displayed in yellow are the result of register REG4.

In the first step (i.e., the first cycle), 13×13 is performed. Our PPM is formed by two PPMs: one PPM is from the PPG (for 13×13) and the other PPM is from register

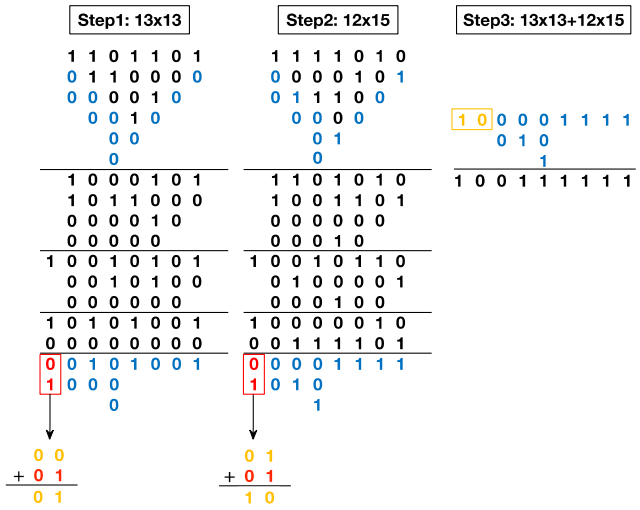

FIGURE 4. The case 13 × 13 + 12 × 15 in an unsigned 4-bit MAC unit.

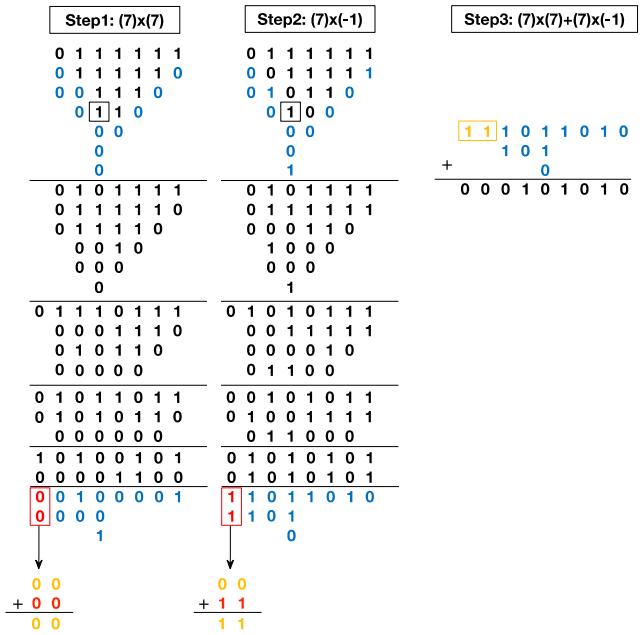

FIGURE 5. The case 7 × 7 + 7 × (−1) in a signed 4-bit MAC unit.

accumulation (including REG1, REG2, and REG3). Note that the initial value of each bit in register accumulation is '0'. After the PPR process is done, the values of REG1, REG2 and REG3 are 0101001, 000 and 0, respectively. In the column of the highest significance bit, the two product terms are 1 and 0, respectively. Thus, the translation circuit translates these two product terms to be binary value 01. Then, binary value 01 is sent to the α-bit adder for the accumulation. After the α-bit addition, the value of register REG4 becomes 01.

In the second step (i.e., the second cycle), 12 × 15 is performed. Our PPM is formed by two PPMs: one PPM is from the PPG (for 12×15) and the other PPM is from register accumulation (including REG1, REG2, and REG3). After the PPR process is done, the values of REG1, REG2 and REG3 are 0001101, 101 and 0, respectively. In the column of the highest significance bit, the two product terms are 1 and 0, respectively. The translation circuit translates these two product terms to be binary value 01. Then, binary value 01 is sent to the α-bit adder for the accumulation. After the α-bit addition, the value of register REG4 becomes 10.

In the third step (i.e., the third cycle), we produce the accumulation result. Each column in the 4 rightmost columns has only one product term. Thus, we only need to use a 5-bit adder to perform the addition of the 5 leftmost columns. Finally, as shown in Fig. 4, the correct result 101011101 is obtained.

The second example is given in Fig. 5. We use the case $7 \times 7 + 7x(-1)$ to demonstrate the detailed process of the signed 4-bit MAC unit. Note that here we use the 2's complement number system to represent a signed value.

In the first step (i.e., the first cycle), 7 × 7 is performed. In the PPG process, we use the Baugh-Wooley algorithm [25], [26] to generate the PPM. Note that, according to the Baugh-Wooley algorithm [25], [26], it is necessary to add an extra '1' in the *(N+1)*-th column. In Fig. 5, we highlight this extra '1' with a black square. After the PPR process is done, the values of REG1, REG2 and REG3 are 0100001, 000 and 1, respectively. In the column of the highest significance bit, the two product terms are 1 and 0, respectively. Thus, the translation circuit translates these two product terms to be binary value 00. Then, binary value 00 is sent to the α-bit adder for the accumulation. After the α-bit addition, the value of register REG4 becomes 00.

In the second step (i.e., the second cycle), 7x(-1) is performed. Our PPM is formed by two PPMs: one PPM is from the PPG (for 7x(-1)) and the other PPM is from register *accumulation* (including REG1, REG2, and REG3). After the PPR process is done, the values of REG1, REG2 and REG3 are 1011010, 101 and 0, respectively. In the column of the highest significance bit, both the two product terms are 0. The translation circuit translates these two product terms to be binary value 11. Then, binary value 11 is sent to the α-bit adder for the accumulation. After the α- bit addition, the value of register REG4 becomes 11.

In the third step (i.e., the third cycle), we produce the accumulation result. We use a 5-bit adder to perform the addition of the 5 leftmost columns. Finally, as shown in Fig. 5, the correct result 000101010 is obtained.

V. EXPERIMENTAL RESULTS

We have implemented a tool (a C++ program) to automatically generate the proposed N-bit MAC in Verilog RTL description. The users can specify the value of N and the value of k for automatic generation, where k denotes the number of higher significance bits whose additions (accumulation) are not performed in the final addition. Note that the value of k is equal to the bit width of register REG2.

In our experiments, we specify the value of N to be 16 (i.e., 16-bit MAC). Besides, we assume that the maximum number of multiplications in each multiply-accumulate operation is

256. Thus, the number of guard bits (i.e., the value of α) is set to be 8.

We have implemented several different configurations of the proposed MAC architecture. For the convenience of presentation, we use the term Ours_k for the naming of each configuration, where k represents the bit width of register REG2. In our experiments, these Verilog RTL descriptions are synthesized to gate-level netlists and targeted to TSMC 40 nm cell library by using Synopsys Design Compiler.

For comparisons, we also implemented the following two MAC architectures: the conventional MAC architecture and the state-of-the-art MAC architecture. In the conventional MAC architecture [11], [12], the MAC unit is composed of two individual blocks (i.e., a multiplier and an accumulator). On the other hand, in the state-of-the-art MAC architecture [16]–[20], the multiplier and the accumulator are tightly integrated (i.e., a carry-save format is sent to the accumulator without being added to only one vector).

According to the two MAC architectures, we implemented five MAC unit models for comparisons: *DT+RC*, *DT+CLA*, *Hoang*, *Peiman*, and *Narendra*. The models *DT+RC* and *DT+CLA* are based on the conventional MAC architecture. The models *Hoang*, *Peiman*, and *Narendra* are based on the state-of-the-art MAC architecture. The details of these five MAC unit models are elaborated below.

- *DT+RC*. Based on the conventional MAC architecture, the PPR process of the multiplier is implemented by the Dadda tree approach (*DT*) and the adders are implemented by the ripple carry adder (*RC*).
- *DT+CLA*. Based on the conventional MAC architecture, the PPR process of the multiplier is implemented by the Dadda tree approach (*DT*) and the adders are implemented by the carry look ahead adder (*CLA*).
- *Hoang*. This model is implemented according to the MAC architecture proposed by Hoang *et al.* [16] (i.e., the state-of-the-art architecture). Note that, in [16], the final adder of the multiplier is implemented by the carry save adder. Then, a carry-save format is sent to the accumulator.
- *Peiman*. This model is also implemented according to the MAC architecture proposed by Hoang *et al.* [16] (i.e., the state-of-the-art architecture). However, this model uses the compressor proposed by Peiman *et al.* [19] for the PPR process.
- *Narendra*. This model is also implemented according to the MAC architecture proposed by Hoang *et al.* [16]. However, this model uses the compressor proposed by Narendra *et al.* 20] for the PPR process.

The first experiment is to perform logic synthesis with the objective of minimizing power consumption under the timing constraint that maximum path delay is at most 1.0 ns. Table 1 tabulates the synthesis results of different unsigned 16-bit MAC unit models. Table 2 tabulates the synthesis results of different signed 16-bit MAC unit models. In Table 1 and Table 2, the column Area denotes circuit area, the column Delay denotes maximum path delay (i.e., minimum possible clock period), the column Power denotes power consumption, and the column Energy denotes energy consumption (note that Energy = Delay × Power). In the column Normalized Energy, the energy consumption of each MAC unit is normalized with respect to that of DT+RC.

TABLE 1. Results of unsigned MAC unit models under timing constraint.

Architecture	Area (μm^2)	Delay (ns)	Power (mW)	Energy (pJ)	Normalized Energy
DT+RC	2943.64	0.98	1.3247	1.30	100.00%
DT+CLA	3050.46	0.98	1.3589	1.33	102.58%
Hoang	2430.61	0.98	1.3065	1.28	98.63%
Peiman	2236.02	0.97	1.1722	1.14	87.59%
Narendra	2222.41	0.97	1.1605	1.13	86.71%
Ours_17	1970.21	0.98	0.8438	0.83	63.70%
Ours_16	1961.81	0.97	0.8440	0.82	63.06%
Ours_15	1949.57	0.97	0.8839	0.86	66.04%
Ours_14	1930.74	0.98	0.8558	0.84	64.60%
Ours_13	1962.50	0.97	0.8543	0.83	63.83%

TABLE 2. Results of signed MAC unit models under timing constraint.

Architecture	Area (μm^2)	Delay (ns)	Power (mW)	Energy (pJ)	Normalized Energy
DT+RC	2804.38	0.98	1.3349	1.31	100.00%
DT+CLA	2786.24	0.98	1.3495	1.32	101.09%
Hoang	2417.46	0.97	1.3451	1.30	99.74%
Peiman	2288.86	0.98	1.2417	1.22	93.02%
Narendra	2203.81	0.98	1.1901	1.17	89.15%
Ours_17	2010.35	0.98	0.8458	0.83	63.36%
Ours_16	1996.66	0.97	0.8410	0.82	62.36%
Ours_15	2000.38	0.97	0.8753	0.85	64.90%
Ours_14	1979.74	0.98	0.8844	0.87	66.25%
Ours_13	2005.14	0.97	0.8745	0.85	64.84%

Since both 16-bit DT+ RC and 16-bit DT+CLA have long carry propagations, these two MAC unit models have large circuit area, large power consumption, and large energy consumption under the timing constraint that maximum path delay is at most 1.0 ns. Therefore, as shown in Table 1 and Table 2, the proposed MAC architecture can greatly reduce circuit area, power consumption, and energy consumption. Using the synthesis results of 16-bit unsigned MAC unit models (i.e., Table 1) as an example, the area of DT+RC is 2943.64 μm^2, while the area of the configuration Ours_16 (i.e., the proposed MAC architecture using 16-bit REG2) is only 1961.81 μm^2. Furthermore, as shown in Table 1, the normalized energy consumption of the configuration Ours_16 is only 63.06%. In other words, compared with DT+RC, the configuration Ours_16 can save 36.94% energy consumption.

Note that the synthesis results displayed in Table 1 and Table 2 are under the timing constraint. From Table 1 and Table 2, we have the following two observations.

TABLE 3. Results of unsigned MAC unit models under area constraint.

Architecture	Area (μm^2)	Delay (ns)	Power (mW)	Energy (pJ)	Normalized Energy
DT+RC	1843.90	1.38	0.7224	1.00	100.00%
DT+CLA	1988.36	1.18	0.8816	1.04	104.35%
Hoang	1995.16	0.96	1.1473	1.10	110.48%
Peiman	1996.75	1.18	0.9032	1.07	106.91%
Narendra	1978.15	1.22	0.8742	1.07	106.98%
Ours_17	1996.97	0.91	0.8965	0.82	81.83%
Ours_16	1995.84	0.92	0.8986	0.83	82.93%
Ours_15	1998.33	0.92	0.9400	0.86	86.75%
Ours_14	1980.19	0.93	0.8859	0.82	82.64%
Ours_13	1972.48	0.95	0.8796	0.84	83.82%

TABLE 4. Results of signed MAC unit models under area constraint.

Architecture	Area (μm^2)	Delay (ns)	Power (mW)	Energy (pJ)	Normalized Energy
DT+RC	1996.29	1.21	0.8700	1.05	100.00%
DT+CLA	1991.30	1.14	0.9525	1.09	103.15%
Hoang	1997.65	0.96	1.1527	1.11	105.12%
Peiman	1983.59	1.20	0.9180	1.10	104.65%
Narendra	1972.02	1.20	0.9144	1.10	104.23%
Ours_17	1994.48	0.99	0.8430	0.83	79.28%
Ours_16	1989.26	1.01	0.8237	0.83	79.03%
Ours_15	1999.92	0.95	0.8847	0.84	79.84%
Ours_14	1998.56	0.95	0.8747	0.83	78.94%
Ours_13	1968.62	0.98	0.8559	0.84	79.68%

- *Comparisons on circuit areas*. Compared with the conventional MAC architecture (i.e., *DT+RC* and *DT+CLA*), both the state-of-the-art MAC architecture (i.e., *Hoang*, *Peiman*, and *Narendra*) and the proposed MAC architecture (i.e., *Ours_17*, *Ours_16*, *Ours_15*, *Ours_14*, and *Ours_13*) can reach smaller circuit areas. Especially, the proposed MAC architecture has a significant reduction on circuit area.
- *Comparisons on power consumption and energy consumption.* Compared with the conventional MAC architecture, both the state-of-the-art architecture and the proposed MAC architecture can achieve smaller power consumption and smaller energy consumption. Especially, the reduction of the proposed MAC architecture on power consumption (energy consumption) is significant. Using Table 1 as an example, compared with the model *DT+RC*, the model *Narendra* can save only 13.29% energy consumption, while the configuration *Ours_16* can save 36.94% energy consumption.

The second experiment is to perform logic synthesis with the objective of minimizing maximum path delay under the area constraint that circuit area is at most 2000 μm^2. Table 3 tabulates the synthesis results of different unsigned 16-bit MAC unit models. Table 4 tabulates the synthesis results of different signed 16-bit MAC unit models. As shown in Table 3 and Table 4, the proposed MAC architecture can greatly reduce maximum path delay (i.e., minimum possible clock period), power consumption, and energy consumption.

Note that the synthesis results displayed in Table 3 and Table 4 are under the area constraint. From Table 3 and Table 4, we have the following four observations.

- *Comparisons on the models based on the conventional MAC architecture*. The models *DT+RC* and *DT+CLA* are based on the conventional MAC architecture. Compared with *DT+RC*, *DT+CLA* can achieve a smaller maximum path delay with a larger power consumption (owing to the mechanism of carry look ahead).
- *Comparisons on the models based on the state-of-the-art architecture*. The models *Hoang*, *Peiman*, and *Narendra* are based on the state-of-the-art MAC architecture. Compared with *Hoang*, both *Peiman* and *Narendra* can achieve smaller power consumption and smaller energy consumption with larger maximum path delays. The reason is that both the compressors used in *Peiman* and the compressors used in *Narendra* are 4:2 compressors (for the PPR process).
- *Comparisons on maximum path delays*. Compared with the model *DT+RC*, both the state-of-the-art MAC architecture and the proposed MAC architecture can achieve smaller maximum path delays (i.e., smaller minimum possible clock periods). Especially, the proposed MAC architecture has a significant reduction on maximum path delay.
- *Comparisons on power consumption and energy consumption*. Compared with *DT+RC*, the state-of-the-art MAC architecture achieves smaller maximum path delay with larger power consumption. Consequently, the energy consumption of the state-of-the-art MAC architecture is even slightly larger than that of *DT+RC*. On the other hand, the proposed MAC architecture can reduce both maximum path delay and power consumption at the same time. As a result, the proposed MAC architecture has a great reduction on energy consumption. Using Table 3 as an example, the configuration *Ours_17* (i.e., the proposed MAC architecture using 17-bit REG2) can save 18.17% energy consumption.

From these experiments, we find that, no matter timing constraint or area constraint, the proposed MAC architecture can always have a large reduction on energy consumption. Thus, the proposed approach works well in practice.

VI. APPLICATION TO A SYSTOLIC ARRAY

The systolic array [28], [29] has been widely used in the hardware acceleration for matrix multiplication. In recent years, several research efforts [30], [31] have been paid to map the inference of a convolutional neural network to a systolic array. Note that a systolic array is composed of multiple processing elements (PEs). Each PE corresponds to a MAC unit. In this section, we address the application of the proposed MAC architecture to a systolic array.

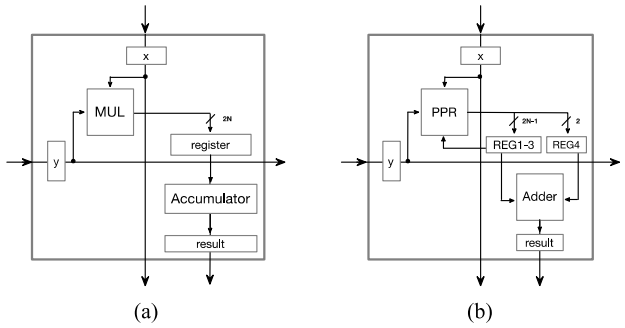

FIGURE 6. (a) The conventional PE. (b) The proposed PE.

Fig. 6(a) gives the block diagram of the PE based on the conventional MAC architecture [11], [12]. Note that the PE is a two-stage (i.e., two-cycle) pipeline design. The inputs of the PE are x and y. The block MUL denotes the multiplier. In the first stage, the multiplier performs the multiplication. Then, the output of the multiplier is stored in a register. In the second stage, the accumulator performs the accumulation. Then, the accumulation result is stored in register *result*.

Fig. 6(b) gives the block diagram of the PE based on the proposed MAC architecture. Note that the PE is a two-stage (i.e., two-cycle) pipeline design. The inputs of the PE are x and y. The block PPR denotes the PPR process. In the first stage, the PPR process is performed. Then, as described in Section III, the output of the PPR process is stored in REG1, REG2, REG3 and REG4. In the second stage, the adder is used to produce the accumulation result. Note that the second stage (the adder) is only enabled in the last cycle of the entire sequence of multiply-accumulate operations.

Here we use 3×3 matrix multiplication as an example to explain the differences between the two systolic arrays, i.e., the systolic array based on the conventional PE (i.e., the conventional MAC architecture) and the systolic array based on the proposed PE (i.e., the proposed MAC architecture). As shown in Fig. 7, a 3×3 systolic array consists of 9 PEs (PE1∼PE9). In Fig. 7, we use the term $a_{i,j}$ and the term $b_{i,j}$, respectively, to represent the element of the two matrices, where i denotes the row number (i.e., $i = 0, 1, 2$) and j denotes the column number (i.e., $j = 0, 1, 2$).

For the systolic array based on the conventional PE (i.e., the conventional MAC architecture), Table 5 displays the detailed operations of each PE (PE1∼PE9) in each cycle. In Table 5, the term *MUL* denotes the multiplication operation (i.e., the first stage of a conventional multiply-accumulate operation) and the term *ACC* denotes the accumulation operation (i.e., the second stage of a conventional multiply-accumulate operation).

For the systolic array based on the conventional PE (i.e., the conventional MAC architecture), Table 5 displays the detailed operations of each PE (PE1∼PE9) in each cycle. In Table 5, the term *MUL* denotes the multiplication operation (i.e., the first stage of a conventional multiply-accumulate operation) and the term *ACC* denotes the accumulation operation

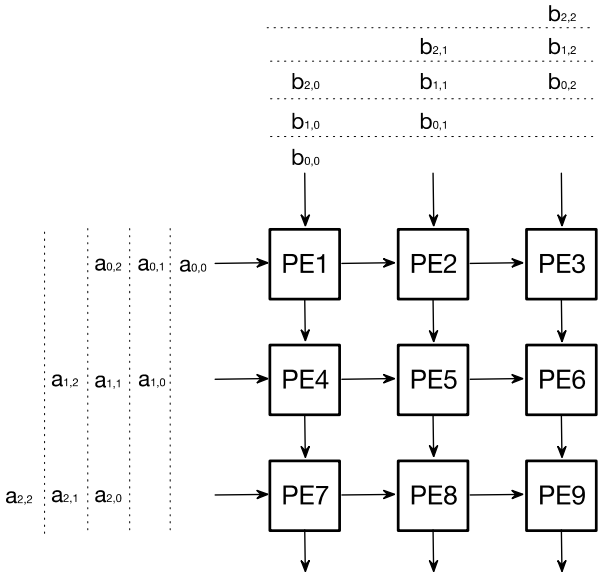

FIGURE 7. Block diagram of 3 × 3 systolic array.

(i.e., the second stage of a conventional multiply-accumulate operation).

We use the detailed operations of PE1 shown in Table 5 as an example for explanation. Note that PE1 is responsible for producing the result of $a_{0,0}\text{x}b_{0,0} + a_{0,1}\text{x}b_{1,0} + a_{0,2}\text{x}b_{2,0}$. In the first cycle, the multiplication of $a_{0,0}$ and $b_{0,0}$, i.e., MUL($a_{0,0}$, $b_{0,0}$), is performed. Then, in the second cycle, MUL($a_{0,1}$, $b_{1,0}$) is performed and the multiplication result of the first cycle, i.e., the result of MUL($a_{0,0}$, $b_{0,0}$), is accumulated. In the third cycle, MUL($a_{0,2}$, $b_{2,0}$) is performed and the result of MUL($a_{0,1}$, $b_{1,0}$) is accumulated. Finally, in the fourth cycle, the result of MUL($a_{0,2}$, $b_{2,0}$) is accumulated.

For the systolic array based on the proposed PE (i.e., the proposed MAC architecture), Table 6 displays the detailed operations of each PE in each cycle. In Table 6, the term *PPR* denotes the PPR process (i.e., the first stage of our multiply-accumulate operation) and the term *ADD* denotes the addition operation (i.e., the second stage of our multiply-accumulate operation).

We use the detailed operations of PE1 shown in Table 6 for the explanation. Note that PE1 is responsible for producing the result of $a_{0,0}\text{x}\ b_{0,0} + a_{0,1}\text{x}b_{1,0} + a_{0,2}\text{x}b_{2,0}$. In the first cycle, the PPR process is performed with respect to inputs $a_{0,0}$ and $b_{0,0}$, i.e., PPR($a_{0,0}$, $b_{0,0}$). Then, in the second cycle, PPR($a_{0,1}$, $b_{1,0}$) is performed. In the third cycle, PPR($a_{0,2}$, $b_{2,0}$) is performed. Finally, in the fourth cycle, an addition (ADD) is performed to produce the accumulation result.

So far, we have not discussed the systolic array based on the state-of-the-art PE (i.e., the state-of-the-art MAC architecture [16]–[20]). In fact, from the viewpoint of functionalities, the two stages of the state-of-the-art MAC architecture are the same as those of the conventional MAC architecture, i.e., multiplication operation (MUL) followed by accumulation operation (ACC). Therefore, for the systolic array based on

TABLE 5. The detailed operations of each conventional PE in each cycle.

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8
PE1	MUL(a_{00},b_{00})	MUL(a_{01},b_{10}) ACC	MUL(a_{02},b_{20}) ACC	ACC				
PE2		MUL(a_{00},b_{01})	MUL(a_{01},b_{11}) ACC	MUL(a_{02},b_{21}) ACC	ACC			
PE3			MUL(a_{00},b_{02})	MUL(a_{01},b_{12}) ACC	MUL(a_{02},b_{22}) ACC	ACC		
PE4		MUL(a_{10},b_{01})	MUL(a_{11},b_{11}) ACC	MUL(a_{12},b_{21}) ACC	ACC			
PE5			MUL(a_{10},b_{01})	MUL(a_{11},b_{11}) ACC	MUL(a_{12},b_{21}) ACC	ACC		
PE6				MUL(a_{10},b_{02})	MUL(a_{11},b_{12}) ACC	MUL(a_{12},b_{22}) ACC	ACC	
PE7			MUL(a_{20},b_{01})	MUL(a_{21},b_{11}) ACC	MUL(a_{22},b_{21}) ACC	ACC		
PE8				MUL(a_{20},b_{01})	MUL(a_{21},b_{11}) ACC	MUL(a_{22},b_{21}) ACC	ACC	
PE9					MUL(a_{20},b_{02})	MUL(a_{21},b_{12}) ACC	MUL(a_{22},b_{22}) ACC	ACC

TABLE 6. The detailed operations of each proposed PE in each cycle.

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8
PE1	PPR(a_{00},b_{00})	PPR(a_{01},b_{10})	PPR(a_{02},b_{20})	ADD				
PE2		PPR(a_{00},b_{01})	PPR(a_{01},b_{11})	PPR(a_{02},b_{21})	ADD			
PE3			PPR(a_{00},b_{02})	PPR(a_{01},b_{12})	PPR(a_{02},b_{22})	ADD		
PE4		PPR(a_{10},b_{01})	PPR(a_{11},b_{11})	PPR(a_{12},b_{21})	ADD			
PE5			PPR(a_{10},b_{01})	PPR(a_{11},b_{11})	PPR(a_{12},b_{21})	ADD		
PE6				PPR(a_{10},b_{02})	PPR(a_{11},b_{12})	PPR(a_{12},b_{22})	ADD	
PE7			PPR(a_{20},b_{01})	PPR(a_{21},b_{11})	PPR(a_{22},b_{21})	ADD		
PE8				PPR(a_{20},b_{01})	PPR(a_{21},b_{11})	PPR(a_{22},b_{21})	ADD	
PE9					PPR(a_{20},b_{02})	PPR(a_{21},b_{12})	PPR(a_{22},b_{22})	ADD

the state-of-the-art PE, to deal with 3 × 3 matrix multiplication, the detailed operations of each PE (PE1∼PE9) in each cycle are the same as Table 5. However, it is noteworthy to mention that, in the state-of-the-art MAC architecture, the output of the multiplier is in the carry-save format.

We have used TSMC 40 nm cell library to implement three 3 × 3 systolic arrays based on these three different PE architectures: the conventional PE, the state-of-the-art PE, and the proposed PE. Note that, for low power, we adopt the model *DT+RC* for the conventional PE and the model *Narendra* for the state-of-the-art PE. The clock rate is assumed to be 1 GHz. Table 7 gives the implementation results. The row *Conventional* denotes the systolic array based on the conventional PE. The row *SOTA* denotes the systolic array based on the state-of-the-art PE. The row *Ours* denotes the systolic array based on the proposed PE. As shown in Table 7, compared with the systolic array based on the conventional PE, the systolic array based on the proposed PE can save 28.8% circuit area and 29.4% power consumption; compared with the systolic array based on the state-of-the-art PE, the systolic array based on the proposed PE can save 9.4% circuit area and 20.8% power consumption.

TABLE 7. Comparisons on 3 × 3 systolic array.

Systolic Array	Area (μm^2)	Power (mW)
Conventional	25239.42	36.0423
SOTA	19834.29	32.1327
Ours	17969.94	25.4412

Further, according to these three different PE architectures, we also have used TSMC 40 nm cell library to implement three 5 × 5 systolic arrays. The clock rate is also assumed to be 1 GHz. Table 8 gives the implementation results. As shown in Table 8, compared with the systolic array based on the conventional PE, the systolic array based on the proposed PE

TABLE 8. Comparisons on 5 × 5 systolic array.

Systolic Array	Area (μm^2)	Power (mW)
Conventional	70109.50	166.86
SOTA	55095.25	148.76
Ours	49916.50	114.35

can save 28.8% circuit area and 31.5% power consumption; compared with the systolic array based on the state-of-the-art PE, the systolic array based on the proposed PE can save 9.4% circuit area and 23.1% power consumption.

From Table 7 and Table 8, we find that, compared with both the systolic array based on the conventional PE (i.e., the conventional MAC architecture) and the systolic array based on the state-of-the-art PE (i.e., the state-of-the-art MAC architecture), the systolic array based on the proposed PE (i.e., the proposed MAC architecture) can greatly reduce both circuit area and power consumption under the same timing constraint.

VII. CONCLUSION

This paper presents a low-power high-speed two-stage pipeline MAC architecture for real-time DSP applications.

Our basic idea is to integrate a part of additions (including a part of the final addition in the multiplication and a part of the addition in the accumulation) into the PPR process. As a result, critical path delays and power dissipations caused by carry propagations can be reduced. To correctly deal with the overflow during the PPR process, an α-bit accumulator is used to count the total number of carries. Experimental results consistently show that the proposed approach works well in practice.

The proposed MAC architecture is applicable to both the design of an unsigned MAC unit and the design of a signed MAC unit. Note that the only differences between the unsigned MAC unit and the signed MAC unit are the PPM structure and the α-bit addition mechanism.

Moreover, the proposed MAC architecture is also applicable to the systolic array (for performing the matrix multiplication). Implementation data show that, compared with the systolic array based on the conventional PE (i.e., the conventional MAC architecture), the systolic array based on the proposed PE (i.e., the proposed MAC architecture) can greatly reduce both circuit area and power consumption under the same timing constraint.

APPENDIX

In this Appendix, we give a detailed explanation to the translation circuit of the signed MAC. Here, without loss of generality, we suppose X and Y are two 2's complement N-bit integers. Thus, we can express X and Y as follows:

$$X = -x_{N-1}2^{2N-1} + \sum_{i=0}^{N-2} x_i 2^i$$
$$Y = -y_{N-1}2^{2N-1} + \sum_{i=0}^{N-2} y_i 2^i$$

where $x_i, y_i \in \{0, 1\}$.

Let P be the product of X and Y. From the Baugh-Wooley algorithm [25], [26], the product P can be expressed below:

$$\begin{aligned} P = {} & x_{N-1}y_{N-1}2^{2N-2} + \sum_{i=0}^{N-2}\sum_{j=0}^{N-2} x_i y_j 2^{i+j} \\ & + 2^{N-1}\sum_{i=0}^{N-2} \overline{y_{N-1}x_i}\,2^i \\ & + 2^{N-1}\sum_{j=0}^{N-2} \overline{x_{N-1}y_j}\,2^j + 2^N - 2^{2N-1} \end{aligned}$$

Note that our PPM generated in the PPG process has only 2N-1 columns (e.g., as displayed in Fig. 5, our PPM has only 7 columns for the signed 4-bit MAC unit). In other words, our PPM does not deal with the 2N-th column. Therefore, the product P' obtained by our PPM can be expressed below:

$$\begin{aligned} P' = {} & x_{N-1}y_{N-1}2^{2N-2} + \sum_{i=0}^{N-2}\sum_{j=0}^{N-2} x_i y_j 2^{i+j} \\ & + 2^{N-1}\sum_{i=0}^{N-2} \overline{y_{N-1}x_i}\,2^i \\ & + 2^{N-1}\sum_{j=0}^{N-2} \overline{x_{N-1}y_j}\,2^j + 2^N \end{aligned}$$

As a result, we have $P = P' - 2^{2N-1}$. In order to obtain P, we need to add -2^{2N-1} into P'. Note that the weight of the least significant bit of the α-bit adder is 2^{2N-1}. Therefore, -2^{2N-1} corresponds to decimal -1 for the α-bit adder.

The inputs to the translation circuit are the two carries (i.e., the two product terms) generated by the PPR process. The α-bit adder is responsible to accumulate the sum of the two carries. However, to compensate for -2^{2N-1}, the output value of the translation circuit (i.e., the input value to the α-bit adder) should be the sum of the two carries minus 1.

Note that the translation circuit is used to translate the two carries to be a corresponding input value of the α-bit adder. From the above discussions, the function of the translation circuit of the signed MAC is derived as below.

- If both the two input values (i.e., the two carries) are 0, the sum of the two input values is 0. Thus, after subtracting 1, the output value is decimal -1.
- If one input value is 0 and the other input value is 1, the sum of the two input values is 1. Thus, after subtracting 1, the output value is decimal 0.
- If both the two input values are 1, the sum of the two input values is 2. Thus, after subtracting 1, the output value is decimal 1.

REFERENCES

[1] A. Abdelgawad, "Low power multiply accumulate unit (MAC) for future wireless sensor networks," in *Proc. IEEE Sensors Appl. Symp. Proc.*, Galveston, TX, USA, Feb. 2013, pp. 129–132.
[2] H. O. Ahmed, M. Ghoneima, and M. Dessouky, "Concurrent MAC unit design using VHDL for deep learning networks on FPGA," in *Proc. IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Penang, Malaysia, Apr. 2018, pp. 31–36.
[3] V. Camus, C. Enz, and M. Verhelst, "Survey of precision-scalable multiply-accumulate units for neural-network processing," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Hsinchu, Taiwan, Mar. 2019, pp. 57–61.
[4] W. J. Townsend, E. E. Swartzlander, and J. A. Abraham, "A Comparison of Dadda and Wallace Multiplier Delays," in *Proc. SPIE Annu. Meeting Opt. Sci. Technol.*, San Diego, CA, USA, 2003, pp. 552–560.

[5] K. L. S. Swee and L. H. Hiung, "Performance comparison review of 32-bit multiplier designs," in *Proc. 4th Int. Conf. Intell. Adv. Syst. (ICIAS)*, Kuala Lumpur, Malaysia, Jun. 2012, pp. 836–841.
[6] S. Asif and Y. Kong, "Design of an algorithmic wallace multiplier using high speed counters," in *Proc. 10th Int. Conf. Comput. Eng. Syst. (ICCES)*, Cairo, Egypt, Dec. 2015, pp. 133–138.
[7] C.-W. Tung and S.-H. Huang, "Low-power high-accuracy approximate multiplier using approximate high-order compressors," in *Proc. 2nd Int. Conf. Commun. Eng. Technol. (ICCET)*, Nagoya, Japan, Apr. 2019, pp. 163–167.
[8] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 689–702, Oct. 1996.
[9] J. Saini, S. Agarwal, and A. Kansal, "Performance, analysis and comparison of digital adders," in *Proc. Int. Conf. Adv. Comput. Eng. Appl.*, Ghaziabad, India, Mar. 2015, pp. 81–83.
[10] L. Pilato, S. Saponara, and L. Fanucci, "Performance of digital adder architectures in 180 nm CMOS standard-cell technology," in *Proc. Int. Conf. Appl. Electron. (AE)*, Pilsen, Czech Republic, Sep. 2016, pp. 211–214.
[11] P. Jebashini, R. Uma, P. Dhavachelv, and H. K. Wye, "A survey and comparative analysis of multiply-accumulate (MAC) block for digital signal processing application on ASIC and FPGA," *J. Appl. Sci.*, vol. 15, no. 7, pp. 934–946, Jul. 2015.
[12] P. A. Patil and C. Kulkarni, "A survey on multiply accumulate unit," in *Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, Pune, India, Aug. 2018, pp. 1–5.
[13] P. F. Stelling and V. G. Oklobdzija, "Implementing multiply-accumulate operation in multiplication time," in *Proc. 13th IEEE Sympsoium Comput. Arithmetic*, Asilomar, CA, USA, Jul. 1997, pp. 99–106.
[14] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient multiply accumulate (MAC) unit for digital signal prossing applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, USA, May 2007, pp. 3199–3202.
[15] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. San Mateo, CA, USA: Morgan Kaufmann, 2003.
[16] T. T. Hoang, M. Sjalander, and P. Larsson-Edefors, "A high-speed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 12, pp. 3073–3081, Dec. 2010.
[17] B. Liebig, J. Huthmann, and A. Koch, "Architecture exploration of high-performance floating-point fused multiply-add units and their automatic use in high-level synthesis," in *Proc. IEEE Int. Symp. Parallel Distrib. Process., Workshops Phd Forum*, Cambridge, MA, USA, May 2013, pp. 134–143.
[18] A. Wahba and H. Fahmy, "Area efficient and fast combined Binary/Decimal floating point fused multiply add unit," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 226–239, Feb. 2017.
[19] P. Aliparast, Z. D. Koozehkanani, and F. Nazari, "An ultra high speed digital 4-2 compressor in 65-nm CMOS," *Int. J. Comput. Theory Eng.*, vol. 5, no. 4, pp. 593–597, Aug. 2013.
[20] C. P. Narendra and K. M. R. Kumar, "Low power compressor based MAC architecture for DSP applications," in *Proc. IEEE Int. Conf. Signal Process., Informat., Commun. Energy Syst. (SPICES)*, Kozhikode, India, Feb. 2015, pp. 1–5.
[21] A. Rezai and P. Keshavarzi, "High-throughput modular multiplication and exponentiation algorithms using Multibit-Scan–Multibit-Shift technique," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1710–1719, Sep. 2015.
[22] A. Rezai and P. Keshavarzi, "High-performance scalable architecture for modular multiplication using a new digit-serial computation," *Microelectron. J.*, vol. 55, pp. 169–178, Sep. 2016.
[23] A. Rezai and P. Keshavarzi, "Compact SD: A new encoding algorithm and its application in multiplication," *Int. J. Comput. Math.*, vol. 94, no. 3, pp. 554–569, Mar. 2017.
[24] D. Nguyen, D. Kim, and J. Lee, "Double MAC: Doubling the performance of convolutional neural networks on modern FPGAs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 890–893.
[25] M. Sjalander and P. Larsson-Edefors, "High-speed and low-power multipliers using the Baugh–Wooley algorithm and HPM reduction tree," in *Proc. 15th IEEE Int. Conf. Electron., Circuits Syst.*, St. Julien's, Malta, Aug. 2008, pp. 33–36.
[26] L.-D. Van and J.-H. Tu, "Power-efficient pipelined reconfigurable fixed-width Baugh–Wooley multipliers," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1346–1355, Oct. 2009.
[27] K. Tsoumanis, S. Xydis, C. Efstathiou, N. Moschopoulos, and K. Pekmestzi, "An optimized modified booth recoder for efficient design of the add-multiply operator," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 4, pp. 1133–1143, Apr. 2014.
[28] F. Moldovan, "Partitioning and mapping algorithms into fixed size systolic arrays," *IEEE Trans. Comput.*, vol. C-35, no. 1, pp. 1–12, Jan. 1986.
[29] N. Petkov, *Systolic Parallel Processing*. New York, NY, USA: Elsevier, 1992.
[30] X. Wei, C. H. Yu, P. Zhang, Y. Chen, Y. Wang, H. Hu, Y. Liang, and J. Cong, "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *Proc. 54th Annu. Design Autom. Conf.*, Austin, TX, USA, Jun. 2017, pp. 1–6.
[31] J. J. Zhang, K. Basu, and S. Garg, "Fault-tolerant systolic array based accelerators for deep neural network execution," *IEEE Des. Test. Comput.*, vol. 36, no. 5, pp. 44–53, Oct. 2019.

CHE-WEI TUNG received the B.S. degree in electronic engineering from Chung Yuan Christian University, Taoyuan City, Taiwan, in 2018, where he is currently pursuing the M.S. degree in electronic engineering, and the dual M.S. degree in electrical engineering from the University of Wisconsin-Milwaukee. His current research interests include high-performance circuit, system design, approximate computing, and VLSI arithmetic circuits.

SHIH-HSU HUANG (Senior Member, IEEE) received the B.S. degree in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1989, the M.S. degree in computer science from National Tsing Hua University, Hsinchu, in 1991, and the Ph.D. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1995.

From 1995 to 2000, he was with the Computer and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, rising to the position of deputy manager of the IC Design Department, responsible for the design of high performance ICs. In 2000, he joined the Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan City, Taiwan, as a Faculty Member, where he is currently a Full Professor. From 2008 to 2012, He served as the Chairman for the Department of Electronic Engineering, Chung Yuan Christian University, where he serves as the Director of the Research Center for Intelligent Electronics, since 2013. His current research interests include high-performance circuit and system design, electronic design automation, and design for testability. He received the Distinguished Professor Award from Chung Yuan Christian University, in 2018.