Modified Booth Multipliers With a Regular Partial Product Array

Shiann-Rong Kuang, *Member, IEEE*, Jiun-Ping Wang, and Cang-Yuan Guo

***Abstract*—The conventional modified Booth encoding (MBE) generates an irregular partial product array because of the extra partial product bit at the least significant bit position of each partial product row. In this brief, a simple approach is proposed to generate a regular partial product array with fewer partial product rows and negligible overhead, thereby lowering the complexity of partial product reduction and reducing the area, delay, and power of MBE multipliers. The proposed approach can also be utilized to regularize the partial product array of posttruncated MBE multipliers. Implementation results demonstrate that the proposed MBE multipliers with a regular partial product array really achieve significant improvement in area, delay, and power consumption when compared with conventional MBE multipliers.**

***Index Terms*—Modified booth, multiplier, partial products, posttruncated.**

I. Introduction

ENHANCING the processing performance and reducing the power dissipation of the systems are the most important design challenges for multimedia and digital signal processing (DSP) applications, in which multipliers frequently dominate the system's performance and power dissipation. Multiplication consists of three major steps: 1) recoding and generating partial products; 2) reducing the partial products by partial product reduction schemes (e.g., Wallace tree [1]–[3]) to two rows; and 3) adding the remaining two rows of partial products by using a carry-propagate adder (e.g., carry look-ahead adder) to obtain the final product. There are already many techniques developed in the past years for these three steps to improve the performance of multipliers. In this brief, we will focus on the first step (i.e., partial product generation) to reduce the area, delay, and power consumption of multipliers.

The partial products of multipliers are generally generated by using two-input AND gates or a modified Booth encoding (MBE) algorithm [3]–[7]. The latter has widely been adopted in parallel multipliers since it can reduce the number of partial product rows to be added by half, thus reducing the size and enhancing the speed of the reduction tree. However, as shown in Fig. 1(a), the conventional MBE algorithm generates $n/2+1$ partial product rows rather than $n/2$ due to the extra partial product bit (neg bit) at the least significant bit position of each partial product row for negative encoding, leading to an irregular partial product array and a complex reduction tree. Some approaches [7], [8] have been proposed to generate more regular partial product arrays, as shown in Fig. 1(b) and (c), for the MBE multipliers. Thus, the area, delay, and power consumption of the reduction tree, as well as the whole MBE multiplier, can be reduced.

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PP_0						$\overline{s_0}$	s_0	s_0	p_{07}	p_{06}	p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	p_{00}
PP_1					1	$\overline{s_1}$	p_{17}	p_{16}	p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	p_{10}		neg_0
PP_2			1	$\overline{s_2}$	p_{27}	p_{26}	p_{25}	p_{24}	p_{23}	p_{22}	p_{21}	p_{20}		neg_1		
PP_3	1	$\overline{s_3}$	p_{37}	p_{36}	p_{35}	p_{34}	p_{33}	p_{32}	p_{31}	p_{30}		neg_2				
PP_4										neg_3						

(a)

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PP_0						$\overline{s_0}$	s_0	s_0	p_{07}	p_{06}	p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	τ_{00}
PP_1					1	$\overline{s_1}$	p_{17}	p_{16}	p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	τ_{10}	c_0	
PP_2			1	$\overline{s_2}$	p_{27}	p_{26}	p_{25}	p_{24}	p_{23}	p_{22}	p_{21}	τ_{20}	c_1			
PP_3	1	$\overline{s_3}$	p_{37}	p_{36}	p_{35}	p_{34}	p_{33}	p_{32}	p_{31}	τ_{30}	c_2					
PP_4									c_3							

(b)

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PP_0						$\overline{s_0}$	s_0	s_0	p_{07}	p_{06}	p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	p_{00}
PP_1					1	$\overline{s_1}$	p_{17}	p_{16}	p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	p_{10}		neg_0
PP_2		1	1	$\overline{s_2}$	p_{27}	p_{26}	p_{25}	p_{24}	p_{23}	p_{22}	p_{21}	p_{20}		neg_1		
PP_3	$\overline{s_3}$	s_3	t_7	t_6	t_5	t_4	t_3	t_2	t_1	t_0		neg_2				

(c)

Fig. 1. Conventional MBE partial product arrays for 8×8 multiplication.

In this brief, we extend the method proposed in [7] to generate a parallelogram-shaped partial product array, which is more regular than that of [7] and [8]. The proposed approach reduces the partial product rows from $n/2+1$ to $n/2$ by incorporating the last neg bit into the sign extension bits of the first partial product row, and almost no overhead is introduced to the partial product generator. More regular partial product array and fewer partial product rows result in a small and fast reduction tree, so that the area, delay, and power of MBE multipliers can further be reduced. In addition, the proposed approach can also be applied to regularize the partial product array of posttruncated MBE multipliers. Posttruncated multiplication, which generates the $2n$-bit product and then rounds the product into n bits, is desirable in many multimedia and DSP systems due to the fixed register size and bus width inside the hardware. Experimental results show that the proposed general and posttruncated MBE multipliers with a regular partial product array can achieve significant improvement in area, delay, and

Manuscript received October 31, 2008; revised January 20, 2009. Current version published May 15, 2009. This work was supported in part by the National Science Council, Taiwan, under Grant NSC 97-2220-E-110-006. This paper was recommended by Associate Editor A.-Y. Wu.

The authors are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan (e-mail: srkuang@cse.nsysu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2009.2019334

TABLE I
MBE TABLE

b_{2i+1}	b_{2i}	b_{2i-1}	Operation	neg_i	two_i	one_i	p_{ij}
0	0	0	+0	0	0	0	0
0	0	1	$+A$	0	0	1	a_j
0	1	0	$+A$	0	0	1	a_j
0	1	1	$+2A$	0	1	0	a_{j-1}
1	0	0	$-2A$	1	1	0	$\overline{a_{j-1}}$
1	0	1	$-A$	1	0	1	$\overline{a_j}$
1	1	0	$-A$	1	0	1	$\overline{a_j}$
1	1	1	-0	0	0	0	0

power consumption when compared with conventional MBE multipliers.

II. CONVENTIONAL MBE MULTIPLIER

Consider the multiplication of two n-bit integer numbers A (multiplicand) and B (multiplier) in 2's complement representation, i.e.,

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$
$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i. \quad (1)$$

In MBE, B in (1) becomes

$$B = \sum_{i=0}^{n/2-1} m_i 2^{2i} = \sum_{i=0}^{n/2-1} (-2b_{2i+1} + b_{2i} + b_{2i-1})2^{2i} \quad (2)$$

where $b_{-1} = 0$, and $m_i \in \{-2, -1, 0, 1, 2\}$. According to the encoded results from B, the Booth selectors choose $-2A$, $-A$, 0, A, or $2A$ to generate the partial product rows, as shown in Table I. The $2A$ in Table I is obtained by left shifting A one bit. Negation operation is achieved by complementing each bit of A (one's complement) and adding "1" to the least significant bit. Adding "1" is implemented as a correction bit neg, which implies that the partial product row is negative $(neg = 1)$ or positive $(neg = 0)$. In addition, because partial product rows are represented in 2's complement representation and every row is left shifted two bit positions with respect to the previous row, sign extensions are required to align the most significant parts of partial product rows. These extra sign bits will significantly complicate the reduction tree. Therefore, many sign extension schemes [3], [9]–[11] have been proposed to prevent extending up the sign bit of each row to the $(2n-1)$th bit position.

Fig. 1(a) illustrates the MBE partial product array for an 8×8 multiplier with a sign extension prevention procedure, where s_i is the sign bit of the partial product row PP_i, $\overline{s_i}$ is the complement of s_i, and b_p indicates the bit position. As can be seen in Fig. 1(a), MBE reduces the number of partial product rows by half, but the correction bits result in an irregular partial product array and one additional partial product row. To have a more regular least significant part of each partial product row PP_i, the authors in [7] added the least significant bit p_{i0} with neg_i in advance and obtained a new least significant bit τ_{i0} and a carry c_i. Note that both τ_{i0} and c_i are generated no later than other partial product bits. Fig. 1(b) depicts the 8×8 MBE partial product array generated by the approach proposed in [7]. Since c_i is at the left one bit position of neg_i, the required additions in the reduction tree are reduced. However, the approach does not remove the additional partial product row PP_4.

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PP_0						α_2	α_1	α_0	p_{07}	p_{06}	p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	τ_{00}
PP_1					1	$\overline{s_1}$	p_{17}	p_{16}	p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	τ_{10}	c_0	
PP_2			1	$\overline{s_2}$	p_{27}	p_{26}	p_{25}	p_{24}	p_{23}	p_{22}	p_{21}	τ_{20}	c_1			
PP_3	1	$\overline{s_3}$	p_{37}	p_{36}	p_{35}	p_{34}	p_{33}	p_{32}	τ_{31}	τ_{30}	c_2					

Fig. 2. Proposed MBE partial product array for 8×8 multiplication.

The problem is overcome in [8] by directly producing the 2's complement representation of the last partial product row $\text{PP}_{n/2-1}$ while the other partial products are produced so that the last neg bit will not be necessary. An efficient method and circuit are developed in [8] to find the 2's complement of the last partial product row in a logarithmic time. As shown in Fig. 1(c), the 10-bit last partial product row and its neg bit in Fig. 1(a) are replaced by the 10-bit 2's complemented partial products $(\overline{s_3}, s_3, t_7, t_6, \ldots, t_0)$ without the last neg bit. Note that one extra "1" is added at the fourteenth bit position to obtain the correct final product. The approach simplifies and speeds up the reduction step, particularly for a multiplier that is in the size of a power of 2 and uses 4–2 compressors [12], [13] to achieve modularity and high performance. However, the approach must additionally develop and design the 2's complement logic, which possibly enlarges the area and delay of the partial product generator and lessens the benefit contributed by removing the extra row.

III. PROPOSED MULTIPLIERS

The proposed MBE multiplier combines the advantages of these two approaches presented in [7] and [8] to produce a very regular partial product array, as shown in Fig. 2. In the partial product array, not only each neg_i is shifted left and replaced by c_i but also the last neg bit is removed by using a simple approach described in detail in the following section.

A. Proposed MBE Multiplier

For MB recoding, at least three signals are needed to represent the digit set $\{-2, -1, 0, 1, 2\}$. Many different ways have been developed, and Table I shows the encoding scheme proposed in [14] that is adopted to implement the proposed MBE multiplier. The Booth encoder and selector circuits proposed in [14] are depicted in Fig. 3(a) and (b), respectively. Based on the recoding scheme and the approach proposed in [7], τ_{i0} and c_i in Fig. 1(b) can be derived from the truth table shown in Table II, as follows:

$$\tau_{i0} = one_i \cdot a_0 = \overline{\overline{one_i} + \overline{a_0}} \quad (3)$$
$$c_i = neg_i \cdot (\overline{one_i} + \overline{a_0}) = \overline{\overline{neg_i} + one_i \cdot a_0}. \quad (4)$$

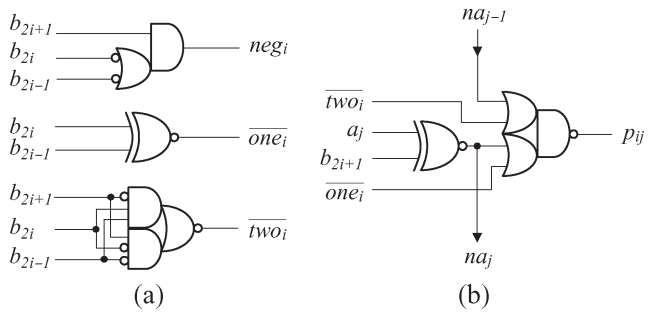

Fig. 3. MBE encoder and selector proposed in [14].

TABLE II
TRUTH TABLE FOR PARTIAL PRODUCT BITS IN THE PROPOSED PARTIAL PRODUCT ARRAY

b_{2i+1}	b_{2i}	b_{2i-1}	τ_{i0}	c_i	τ_{i1}	d_i	$\tilde{\tau}_{i1}$	e_i
0	0	0	0	0	0	0	1	0
0	0	1	a_0	0	a_1	0	$\overline{a_1}$	a_1
0	1	0	a_0	0	a_1	0	$\overline{a_1}$	a_1
0	1	1	0	0	a_0	0	$\overline{a_0}$	a_0
1	0	0	0	1	a_0	$\overline{a_0}$	$\overline{a_0}$	1
1	0	1	a_0	$\overline{a_0}$	$a_0 \oplus a_1$	$\overline{a_0 + a_1}$	$a_0 \odot a_1$	$\overline{a_0 \cdot a_1}$
1	1	0	a_0	$\overline{a_0}$	$a_0 \oplus a_1$	$\overline{a_0 + a_1}$	$a_0 \odot a_1$	$\overline{a_0 \cdot a_1}$
1	1	1	0	0	0	0	1	0

According to (3) and (4), τ_{i0} and c_i can be produced by one NOR gate and one AOI gate, respectively. Moreover, they are generated no later than other partial product bits.

To further remove the additional partial product row $\text{PP}_{n/2}$ [i.e., PP_4 in Fig. 1(b)], we combine the c_i for $i = n/2 - 1$ with the partial product bit p_{i1} to produce a new partial product bit τ_{i1} and a new carry d_i. Then, the carry d_i can be incorporated into the sign extension bits of PP_0. However, if τ_{i1} and d_i are produced by adding c_i and p_{i1}, their arrival delays will probably be larger than other partial product bits. Therefore, we directly produce τ_{i1} and d_i for $i = n/2 - 1$ from A, B, and the outputs of the Booth encoder (i.e., neg_i, two_i, and one_i), as shown in Table II, where $\bigoplus$ and $\odot$ denote the Exclusive-OR and Exclusive-NOR operations, respectively. The logic expressions of τ_{i1} and d_i can be written as

$$\tau_{i1} = one_i \cdot \varepsilon + two_i \cdot a_0 = \overline{(\overline{one_i} + \overline{\varepsilon}) \cdot (\overline{two_i} + \overline{a_0})} \tag{5}$$

$$d_i = \overline{(\overline{b_{2i+1}} + a_0)} \cdot \overline{[(b_{2i-1} + a_1) \cdot (b_{2i} + a_1) \cdot (b_{2i} + b_{2i-1})]} \tag{6}$$

where

$$\overline{\varepsilon} = \begin{cases} a_1, & \text{if } \overline{a_0 \cdot b_{2i+1}} = 0 \\ \overline{a_1}, & \text{otherwise.} \end{cases} \tag{7}$$

Since the weight of d_i is 2^n, which is equal to the weight of s_0 at bit position n, d_i can be incorporated with the sign extension bits $\overline{s_0}s_0s_0$ of PP_0. Let $\alpha_2\alpha_1\alpha_0$ be the new bits after incorporating d_i into $\overline{s_0}s_0s_0$; the relations between them are summarized in Table III. As can be seen in Table III, the maximal value of $\overline{s_0}s_0s_0$ is 100 so that the addition of $\overline{s_0}s_0s_0$ and d_i will never produce an overflow. Therefore, $\alpha_2\alpha_1\alpha_0$ is enough to represent the sum of $\overline{s_0}s_0s_0$ and d_i. According to Table III, α_2, α_1, and α_0 can be expressed as

TABLE III
TRUTH TABLE FOR NEW SIGN EXTENSION BITS

$\overline{s_0}$	s_0	s_0	d_i	α_2	α_1	α_0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0

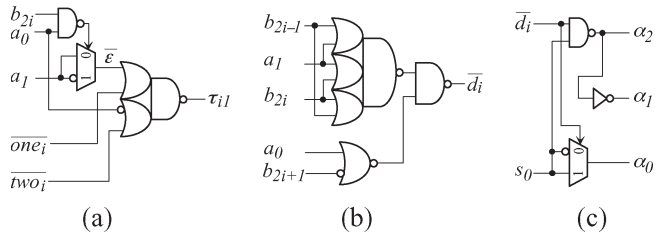

Fig. 4. Proposed circuits to generate τ_{i1}, $\overline{d_i}$, and $\alpha_2\alpha_1\alpha_0$ for $i = n/2 - 1$.

$$\alpha_2 = \overline{(s_0 \cdot \overline{d_i})} \tag{8}$$

$$\alpha_1 = s_0 \cdot \overline{d_i} = \overline{\alpha_2} \tag{9}$$

$$\alpha_0 = s_0 \odot \overline{d_i}. \tag{10}$$

The corresponding circuits to generate τ_{i1}, $\overline{d_i}$, and $\alpha_2\alpha_1\alpha_0$ are depicted in Fig. 4(a)–(c), respectively. The partial product array generated by the proposed approach for the 8×8 multiplier is shown in Fig. 2. This regular array is generated by only slightly modifying the original partial product generation circuits and introducing almost no area and delay overhead.

B. Proposed Posttruncated MBE Multiplier

As mentioned earlier, the product of an $n \times n$ multiplier is frequently rounded to n bits. A posttruncated multiplier can be accomplished by adding a "1" at the $(n-1)$th bit position of the partial product array and then truncating the least significant n-bit of the final $2n$-bit product, as shown in Fig. 5(a). Unfortunately, this extra "1" will result in one additional partial product row like $neg_{n/2-1}$, and it cannot be removed by directly producing the 2's complement representation of the last partial product row $\text{PP}_{n/2-1}$.

On the other hand, the proposed approach to remove $neg_{n/2-1}$ can easily be extended to simultaneously remove this extra "1." Because the weight of the extra "1" is equal to the weight of p_{i1} for $i = n/2 - 1$, we add the two least significant bits $p_{i1}p_{i0}$ with the extra "1" and $neg_{n/2-1}$ beforehand to obtain a 2-bit sum $\tilde{\tau}_{i1}\tau_{i0}$ and a carry e_i. τ_{i0} can be generated according to (3). Similar to τ_{i1} and d_i, $\tilde{\tau}_{i1}$ and e_i for $i = n/2 - 1$ are directly produced from A, B, and the outputs of the Booth encoder to shorten their arrival delays. The relations between them are also listed in Table II, and $\tilde{\tau}_{i1}$ and e_i can be obtained as follows:

$$\tilde{\tau}_{i1} = \overline{\tau_{i1}} \tag{11}$$

$$e_i = \overline{(\overline{\kappa} + one_i) \cdot (\overline{\pi} + \overline{one_i})} \tag{12}$$

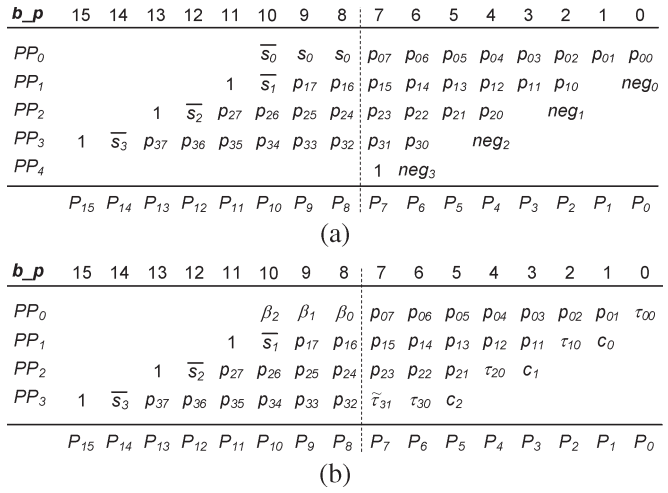

Fig. 5. MBE partial product arrays for 8 × 8 posttruncated multiplication.

where

$$\overline{\kappa} = \overline{(\overline{b_{2n+1}} \cdot b_{2n} \cdot a_0 + b_{2n+1} \cdot \overline{b_{2n}})} \tag{13}$$

$$\overline{\pi} = b_{2n+1} \cdot a_1 \cdot a_0 + \overline{b_{2n+1}} \cdot \overline{a_1}. \tag{14}$$

Subsequently, the carry e_i must also be incorporated with the sign extension bits $\overline{s_0}s_0s_0$ of PP_0 to remove the additional partial product row. Let $\beta_2\beta_1\beta_0$ be the result of incorporating e_i into $\overline{s_0}s_0s_0$; they can be obtained by the similar method as $\alpha_2\alpha_1\alpha_0$ shown in (8)–(10). The partial product array generated by the proposed approach for the 8 × 8 posttruncated multiplier is shown in Fig. 5(b).

IV. Experimental Results

For comparison, we have implemented several MBE multipliers whose partial product arrays are generated by using different approaches. Except for a few of partial product bits that are generated by different schemes to regularize the partial product array, the other partial product bits are generated by using the similar method and circuits for all multipliers. In addition, the partial product array of each multiplier is reduced by a Wallace tree scheme, and a carry look-ahead adder is used for the final addition. These multipliers were modeled in Verilog HDL and synthesized by using Synopsys Design Compiler with an Artisan TSMC 0.18-μm 1.8-V standard cell library. The synthesized netlists were then fed into Cadence SOC Encounter to perform placement and routing [15]. Delay estimates were obtained after RC extraction from the placed and routed netlists. Power dissipation was estimated from the same netlists by feeding them into Synopsys Nanosim to perform full transistor-level power simulation at a clock frequency of 50 MHz with 5000 random input patterns.

The implementation results, including the hardware area, critical path delay, and power consumption for these multipliers with $n = 8, 16,$ and 32, are listed in Table IV, where $CBMa$, $CBMb$ [7], and $CBMc$ [8] denote the MBE multipliers with the partial product array shown in Fig. 1(a)–(c), respectively. Moreover, $A(-)$, $D(-)$, and $P(-)$ denote the area, delay, and power decrements when compared with the $CBMa$ multiplier.

TABLE IV
Experimental Results of MBE Multipliers

n	Multiplier	*Area* (μm^2)	$A(-)$ (%)	*Delay* (*ns*)	$D(-)$ (%)	*Power* (μW)	$P(-)$ (%)
8	*CBMa*	7311	-	4.19	-	728.3	-
	CBMb [7]	6796	7.0	4.10	2.1	660.2	9.4
	CBMc [8]	6560	10.3	3.98	5.0	615.3	15.5
	Proposed	5921	19.0	3.81	9.1	489.0	32.9
16	*CBMa*	28541	-	5.73	-	3740.9	-
	CBMb [7]	27706	2.9	5.61	2.1	3506.7	6.3
	CBMc [8]	26877	5.8	5.69	0.7	3478.0	7.0
	Proposed	25045	12.3	5.60	2.3	3057.3	18.3
32	*CBMa*	112549	-	6.86	-	19004.9	-
	CBMb [7]	110689	1.7	6.78	1.2	18650.1	1.9
	CBMc [8]	107187	4.8	6.69	2.5	16932.7	10.9
	Proposed	103607	7.9	6.65	3.1	16803.4	11.6

TABLE V
Experimental Results of Posttruncated Multipliers

n	Multiplier	*Area* (μm^2)	$A(-)$ (%)	*Delay* (*ns*)	$D(-)$ (%)	*Power* (μW)	$P(-)$ (%)
8	*Conventional*	6623	-	4.02	-	643.9	-
	Proposed	5828	12.0	3.83	4.6	525.0	18.5
16	*Conventional*	25271	-	5.76	-	3135.3	-
	Proposed	22040	12.8	5.55	3.7	2503.9	20.1
32	*Conventional*	108457	-	6.90	-	18496.4	-
	Proposed	98794	8.9	6.88	0.3	15794.1	14.6

As can be seen in Table IV, the conventional multiplier $CBMa$ typically has the largest area, delay, and power consumption due to its irregular product array and complex reduction tree. The $CBMb$ and $CBMc$ multipliers can really achieve improvement in area, delay, and power consumption when compared with the $CBMa$ multiplier. Moreover, the proposed multiplier offers more area, delay, and power reduction over the $CBMb$ and $CBMc$ multipliers. For example, the proposed 8 × 8 multiplier gives 12.0%, 7.0%, and 23.5% improvement over the $CBMb$ multiplier in area, delay, and power, respectively. In addition, it offers 8.7%, 4.1%, and 17.4% savings over the $CBMc$ multiplier. The achievable improvement in area, delay, and power by the proposed 32 × 32 multiplier is also respectable. It gives 6.2%, 1.9%, and 9.7% and 3.1%, 0.6%, and 0.7% improvement over the $CBMb$ and $CBMc$ multipliers in area, delay, and power, respectively.

On the other hand, for comparison, we also implemented and synthesized the conventional posttruncated multiplier with the partial product array shown in Fig. 5(a) and the proposed posttruncated multiplier with the partial product array shown in Fig. 5(b). In these posttruncated multipliers, the cells in the carry look-ahead adder for producing the least significant n-bit product (i.e., the circuits for generating the sum) are removed to reduce the area and power consumption. Table V shows the implementation results, and the proposed 8 × 8 posttruncated multiplier can offer 12.0%, 4.6%, and 18.5% decrement over the conventional posttruncated multiplier in area, delay, and power, respectively. In addition, the proposed 32 × 32 posttruncated multiplier gives 8.9%, 0.3%, and 14.6%

improvement over the conventional posttruncated multiplier in area, delay, and power, respectively. The results show that the proposed approach can also efficiently reduce the area, delay, and power consumption of posttruncated multipliers.

V. Conclusion

In this brief, a simple approach has been proposed to generate a regular partial product array with fewer partial product rows, thereby reducing the area, delay, and power of MBE multipliers. The proposed approach has also been extended to regularize the partial product array of posttruncated MBE multipliers. Experimental results have demonstrated that the proposed MBE and posttruncated MBE multipliers with regular partial product arrays can achieve significant improvement in area, delay, and power consumption when compared with conventional multipliers.

References

[1] C. S. Wallace, "A suggestion for parallel multipliers," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[2] O. Hasan and S. Kort, "Automated formal synthesis of Wallace tree multipliers," in *Proc. 50th Midwest Symp. Circuits Syst.*, 2007, pp. 293–296.

[3] 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.

[4] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified Booth algorithm," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 9, pp. 902–908, Sep. 2000.

[5] K. Choi and M. Song, "Design of a high performance 32 $\times$ 32-bit multiplier with a novel sign select Booth encoder," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2001, vol. 2, pp. 701–704.

[6] Y. E. Kim, J. O. Yoon, K. J. Cho, J. G. Chung, S. I. Cho, and S. S. Choi, "Efficient design of modified Booth multipliers for predetermined coefficients," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 2717–2720.

[7] W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 692–701, Jul. 2000.

[8] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1253–1258, Oct. 2006.

[9] O. Salomon, J.-M. Green, and H. Klar, "General algorithms for a simplified addition of 2's complement numbers," *IEEE J. Solid-State Circuits*, vol. 30, no. 7, pp. 839–844, Jul. 1995.

[10] E. de Angel and E. E. Swartzlander, Jr., "Low power parallel multipliers," in *Workshop VLSI Signal Process. IX*, 1996, pp. 199–208.

[11] A. A. Farooqui and V. G. Oklobdzija, "General data–path organization of a MAC unit for VLSI implementation of DSP processors," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1998, vol. 2, pp. 260–263.

[12] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, "Design of high-speed low-power 3–2 counter and 4–2 compressor for fast multipliers," *Electron. Lett.*, vol. 34, no. 4, pp. 341–343, Feb. 1998.

[13] C.-H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4–2 and 5–2 compressors for fast arithmetic circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

[14] Z. Huang and M. D. Ercegovac, "High-performance low-power left-to-right array multiplier design," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 272–283, Mar. 2005.

[15] *CIC Referenced Flow for Cell-based IC Design*, 2008, Taiwan: Chip Implementation Center, CIC. Document no. CIC-DSD-RD-08-01.