A Modified Partial Product Generator for Redundant Binary Multipliers

Xiaoping Cui, Weiqiang Liu, *Senior Member, IEEE*, Xin Chen, Earl E. Swartzlander Jr., *Life Fellow, IEEE*, and Fabrizio Lombardi, *Fellow, IEEE*

Abstract—Due to its high modularity and carry-free addition, a redundant binary (RB) representation can be used when designing high performance multipliers. The conventional RB multiplier requires an additional RB partial product (RBPP) row, because an error-correcting word (ECW) is generated by both the radix-4 Modified Booth encoding (MBE) and the RB encoding. This incurs in an additional RBPP accumulation stage for the MBE multiplier. In this paper, a new RB modified partial product generator (RBMPPG) is proposed; it removes the extra ECW and hence, it saves one RBPP accumulation stage. Therefore, the proposed RBMPPG generates fewer partial product rows than a conventional RB MBE multiplier. Simulation results show that the proposed RBMPPG based designs significantly improve the area and power consumption when the word length of each operand in the multiplier is at least 32 bits; these reductions over previous NB multiplier designs incur in a modest delay increase (approximately 5 percent). The power-delay product can be reduced by up to 59 percent using the proposed RB multipliers when compared with existing RB multipliers.

Index Terms—Redundant binary, modified booth encoding, RB partial product generator, RB multiplier

1 Introduction

Digital multipliers are widely used in arithmetic units of microprocessors, multimedia and digital signal processors. Many algorithms and architectures have been proposed to design high-speed and low-power multipliers [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. A normal binary (NB) multiplication by digital circuits includes three steps. In the first step, partial products are generated; in the second step, all partial products are added by a partial product reduction tree until two partial product rows remain. In the third step, the two partial product rows are added by a fast carry propagation adder. Two methods have been used to perform the second step for the partial product reduction. A first method uses four-two compressors, while a second method uses redundant binary (RB) numbers [5], [6]. Both methods allow the partial product reduction tree to be reduced at a rate of 2:1.

The redundant binary number representation has been introduced by Avizienis [1] to perform signed-digit arithmetic; the RB number has the capability to be represented in different ways. Fast multipliers can be designed using redundant binary addition trees [2], [3]. The redundant binary representation has also been applied to a floating-point processor and implemented in VLSI [4]. High performance RB multipliers have become popular due to the advantageous features, such as high modularity and carry-free addition [5], [6], [7], [8], [9].

A RB multiplier consists of a RB partial product (RBPP) generator, a RBPP reduction tree and a RB-NB converter. A Radix-4 Booth encoding or a modified Booth encoding (MBE) is usually used in the partial product generator of parallel multipliers to reduce the number of partial product rows by half [5], [6], [10], [11], [12], [13]. A RBPP row can be obtained from two adjacent NB partial product rows by inverting one of the pair rows [5], [6]; an N-bit conventional RB MBE (CRBBE-2) multiplier requires $\lceil N/4 \rceil$ RBPP rows. An additional error-correcting word (ECW) is also required by both the RB and the Booth encoding [5], [6] [14]; therefore, the number of RBPP accumulation stages (NRBPPAS) required by a power-of-two word-length (i.e., 2^n-bit) multiplier is given by:

$$\begin{aligned} \mathrm{NRBPPAS} &= \lceil \log_2(N/4+1) \rceil \\ &= n-1, \text{if } N = 2^n. \end{aligned} \tag{1}$$

If the additional ECW can be removed, an RBPP accumulation stage is saved, so resulting in improvements of complexity and critical path delay for a RB multiplier. For example, a conventional 32-bit RB multiplier has four RBPP accumulation stages; if the ECW is removed, then the number of RBPP accumulation stages is reduced to 3, i.e., the stage count is decreased by 25 percent. Note that the problem of extra ECW does not exist in standard significand size (i.e., 24×24-bit and 54×54-bit) RB multipliers as used in floating point-arithmetic units [5], [6].

Alternatively, a high-radix Booth encoding technique can reduce the number of partial products. However, the number of expensive hard multiples (i.e., a multiple that is not a power of two and the operation cannot be performed by simple shifting and/or complementation) increases too [14], [15], [16]. Besli and Desmukh [16] noticed that some hard multiples can be obtained by the differences of two simple power-of-two multiplies. A new radix-16 Booth encoding (RBBE-4) technique without ECW has been proposed in [14]; it avoids the issue of hard multiples. A radix-16 RB Booth encoder can be used to overcome the hard multiple problem and avoid the extra ECW, but at the cost of doubling the number of RBPP rows. Therefore, the number of radix-16 RBPP rows is the same as in the radix-4 MBE. However, the RBPP generator based on a radix-16 Booth encoding has a complex circuit structure and a lower speed compared with the MBE partial product generator [10] when requiring the same number of partial products.

This paper focuses on the RBPP generator for designing a 2^n-bit RB multiplier with fewer partial product rows by eliminating the extra ECW. A new RB modified partial product generator based on MBE (RBMPPG-2) is proposed. In the proposed RBMPPG-2, the ECW of each row is moved to its next neighbor row. Furthermore, the extra ECW generated by the last partial product row is combined with both the two most significant bits (MSBs) of the first partial product row and the two least significant bits (LSBs) of the last partial product row by logic simplification. Therefore, the proposed method reduces the number of RBPP rows from $N/4+1$ to $N/4$, i.e., a RBPP accumulation stage is saved. The proposed method is applied to 8×8-, 16×16-, 32×32-, and 64×64-bit RB multiplier designs; the designs are synthesized using the NanGate 45 nm Open Cell Library. The proposed designs achieve significant reductions in area and power consumption compared with existing multipliers when the word length of each of the operands is at least 32 bits. While a modest increase in delay is encountered (approximately 5 percent), the power-delay product (PDP) at word lengths of at least 32 bits confirms that the proposed designs are the best also by this figure of merit.

This paper is organized as follows. Section 2 introduces radix-4 Booth encoding. The design of the conventional RBPP generator is also reviewed. Section 3 presents the proposed RBMPPG. This section also demonstrates the adoption of the proposed RBMPPG into various word-length RB multipliers. Section 4 provides the evaluation results of the new RB multipliers using the proposed RBMPPG for different word lengths and compares them to

- *X. Cui, W. Liu, and X. Chen are with the College of Electronic Information and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, Jiangsu, China. E-mail: {wnhcxp, liuweiqiang, xin_chen}@nuaa.edu.cn.*
- *E. E. Swartzlander, Jr. is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712. E-mail: eswartzla@aol.com.*
- *F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: lombardi@ece.neu.edu.*

Manuscript received 29 Dec. 2014; revised 27 May 2015; accepted 29 May 2015. Date of publication 3 June 2015; date of current version 17 Mar. 2016.
Recommended for acceptance by J.-M. Muller.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2441711

TABLE 1
MBE Scheme

b_{2i+1}, b_{2i}, b_{2i-1}	Operation
000	0
001	+A
010	+A
011	+2A
100	−2A
101	−A
110	−A
111	0

previous best designs found in the technical literature. The conclusion is provided in Section 5.

2 Review of Booth Encoding and RB Partial Product Generator

2.1 Radix-4 Booth Encoding

Booth encoding has been proposed to facilitate the multiplication of two's complement binary numbers [17]. It was revised as modified Booth encoding or radix-4 Booth encoding [18]. The MBE scheme is summarized in Table 1, where $A = a_{N-1}a_{N-2}\ldots a_2\ a_1a_0$ stands for the multiplicand, and $B = b_{N-1}b_{N-2}\ldots b_2\ b_1b_0$ stands for the multiplier. The multiplier bits are grouped in sets of three adjacent bits. The two side bits are overlapped with neighboring groups except the first multiplier bits group in which it is {b_1, b_0, 0}. Each group is decoded by selecting the partial product shown in Table 1, where $2A$ indicates twice the multiplicand, which can be obtained by left shifting. Negation operation is achieved by inverting each bit of A and adding '1' (defined as correction bit) to the LSB [10], [11], [12], [13]. Methods have been proposed to solve the problem of correction bits for NB radix-4 Booth encoding (NBBE-2) multipliers. However, this problem has not been solved for RB MBE multipliers.

2.2 RB Partial Product Generator

As two bits are used to represent one RB digit, then a RBPP is generated from two NB partial products [1], [2], [3], [4], [5], [6]. The addition of two N-bit NB partial products X and Y using two's complement representation can be expressed as follows [6]:

$$\begin{aligned} X+Y &= X-\overline{Y}-1 \\ &= \left(-x_N2^N+\sum_{i=0}^{N-1}x_i2^i\right)-\left(-\overline{y_N}2^N+\sum_{i=0}^{N-1}\overline{y_i}2^i\right)-1 \\ &= -\left(x_N-\overline{y_N}\right)2^N+\sum_{i=0}^{N-1}(x_i-\overline{y_i})2^i-1 \\ &= \left(X,\overline{Y}\right)-1, \end{aligned} \tag{2}$$

where $\overline{Y}$ is the inverse of Y, and the same convention is used in the rest of the paper. The composite number $\left(X,\overline{Y}\right)$ can be interpreted as a RB number. The RBPP is generated by inverting one of the two NB partial products and adding -1 to the LSB. Each RB digit X_i belongs to the set $\left\{\overline{1},0,1\right\}$; this is coded by two bits as the pair (X_i^-, X_i^+). Note that $\overline{1}=-1$. RB numbers can be coded in several ways. Table 2 shows one specific RB encoding [6], where the RB digit is obtained by performing $X_i^+ - X_i^-$.

TABLE 2
RB Encoding Used in This Work [6]

X_i^+	X_i^-	RB digit (X_i)
0	0	0
0	1	$\overline{1}$
1	0	1
1	1	0

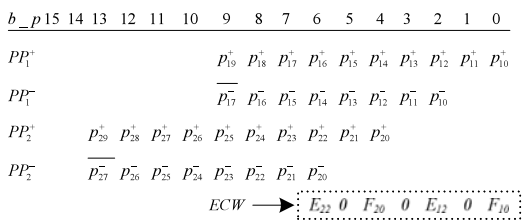

Fig. 1. Conventional RBPP architecture for an 8-bit MBE multiplier.

Both MBE and RB coding schemes introduce errors and two correction terms are required: 1) when the NB number is converted to a RB format, -1 must be added to the LSB of the RB number; 2) when the multiplicand is multiplied by -1 or -2 during the Booth encoding, the number is inverted and $+1$ must be added to the LSB of the partial product. A single ECW can compensate errors from both the RB encoding and the radix-4 Booth recoding. The conventional partial product architecture of an 8-bit MBE multiplier [5], [6] is shown in Fig. 1, where b_p represents the bit position, p_{ij}^+ or p_{ij}^- is generated by using an encoder and decoder (Fig. 2) [10]. An N-bit CRBBE-2 multiplier includes $N/4$ RBPP rows and one ECW; the ECW takes the form as follows:

$$ECW = E_{(N/4)2}0\ F_{(N/4)0}\ldots 0E_{i2}0\ F_{i0}\ldots 0\ E_{12}0\ F_{10}, \tag{3}$$

where i represents the ith row of the RBPPs, $E_{i2}\in\{0,1\}$ and $F_{i0}\in\{0,\overline{1}\}$. In F_{i0}, a -1 correction term is always required by RB coding. If F_{i0} also corrects the errors from the MBE recoding, then the correction term cancels out to 0. That is to say that if the multiplicand digit is inverted and added to 1, then F_{i0} is 0, otherwise F_{i0} is -1. The error-correcting digit E_{i2} is determined only by the Booth encoding:

$$E_{i2}=\begin{cases}0, & \text{no negative encoding}\\ 1, & \text{negative encoding.}\end{cases} \tag{4}$$

As shown in Fig. 1 the first RBPP row, i.e. PP_1, consists of the first partial product row PP_1^+ and the second partial product row PP_1^- i.e., $PP_1^+ = p_{19}^+p_{18}^+\ldots p_{10}^+$and $PP_1^- = p_{17}^-p_{16}^-\ldots p_{10}^-$, where, p_{19}^+ and p_{18}^+ are the sign extension bits, so

$$p_{19}^+ = \overline{p_{18}^+}, \tag{5}$$

$$\begin{aligned} p_{18}^+ =& \overline{b_1}\ \overline{b_0}\cdot 0+\overline{b_1}b_0\cdot a_7+b_1\overline{b_0}\cdot\overline{a_7}+b_1b_0\cdot\overline{a_7} \\ =& \overline{b_1}b_0\cdot a_7+b_1\overline{a_7}. \end{aligned} \tag{6}$$

According to Eq. (2), the sign extension bit p_{29}^+ is also the inverse of p_{28}^+. The p_{17}^- in PP_1^- and the p_{27}^- in PP_2^- are also negated as $\overline{p_{17}^-}$ and $\overline{p_{27}^-}$. Eq. (5) and Eq. (6) are further used in the next section when presenting the proposed modified RBPP generator.

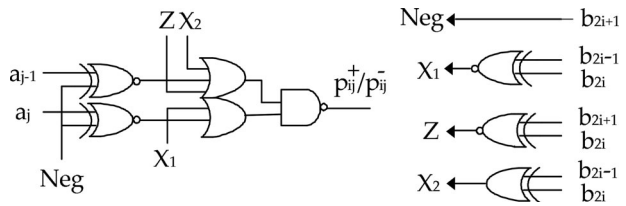

Fig. 2. An encoder and decoder of the MBE scheme [10].

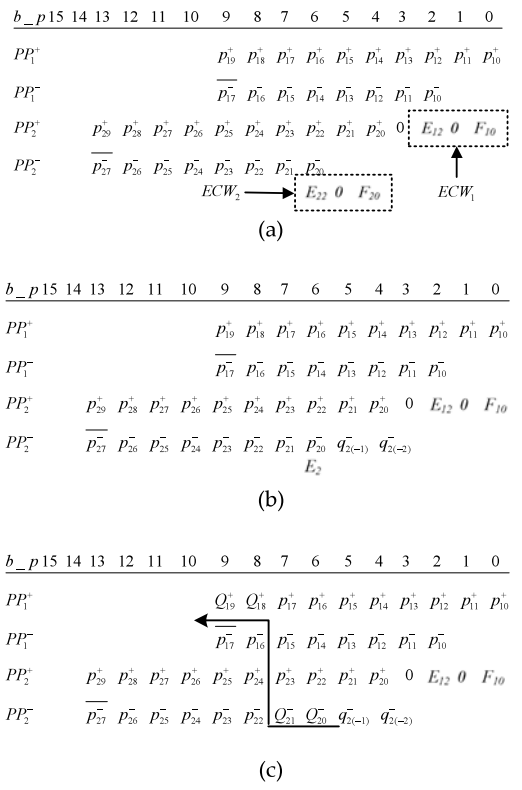

Fig. 3. (a) The first new RBMPPG-2 architecture for an 8-bit MBE multiplier; (b) the further revised RBMPPG-2 architecture by replacing E_{22} and F_{20} with E_2, $q^-_{2(-2)}$, *and* $q^-_{2(-1)}$; (c) the final proposed RBMPPG-2 architecture by totally eliminating ECW_2 and further combing E_2 into Q^+_{19}, Q^+_{18}, Q^-_{21}, and Q^-_{20}.

For a 2^n-bit CRBBE-2 multiplier, one additional RBPP accumulation stage is required due to the ECW. For a 64-bit RB multiplier, there are five RBPP accumulation stages; therefore, the number of RBPP accumulation stages can be reduced by 20 percent when eliminating the ECW in a 64-bit RB multiplier, which improves both the complexity and the critical path delay.

3 Proposed RB Partial Product Generator

A new RB modified partial product generator based on MBE (RBMPPG-2) is presented in this section; in this design, ECW is eliminated by incorporating it into both the two MSBs of the first partial product row (PP^+_1) and the two LSBs of the last partial product row ($PP^-_{(N/4)}$).

3.1 Proposed RBMPPG−2

Fig. 3 illustrates the proposed RBMPPG-2 scheme for an 8×8-bit multiplier. It is different from the scheme in Fig. 1, where all the error-correcting terms are in the last row. ECW_1 is generated by PP_1 and expressed as

$$ECW_1 = 0\ E_{12} 0\ F_{10}. \tag{7}$$

The ECW_2 generated by PP_2 (also defined as an extra ECW) is left as the last row and it is expressed as:

$$ECW_2 = 0\, E_{22}\, 0\, F_{20}. \tag{8}$$

To eliminate a RBPP accumulation stage, ECW_2 needs to be incorporated into PP_1 and PP_2. As discussed in Section 2.2 for F_{i0} and as per Table 1, F_{20} is determined by b_5, b_4, b_3 as follows:

$$F_{20} = \begin{cases} -1, & b_5b_4b_3 = 000, 001, 010, 011, \text{ or } 111 \\ 0, & b_5b_4b_3 = 100, 101, \text{or} 110. \end{cases} \tag{9}$$

TABLE 3
Truth Table of E_2, $q^-_{2(-2)}$, $q^-_{2(-1)}$ and p^-_{21}, p^-_{20}

$b_7b_6b_5$	$E_{22}F_{20}$	$E_2q^-_{2(-2)}q^-_{2(-1)}$	p^-_{21}	p^-_{20}
0 0 0	0 $\bar{1}$	$\bar{1}$ 1 1	0	0
0 0 1	0 0	0 0 0	a_1	a_0
0 1 0	0 $\bar{1}$	$\bar{1}$ 1 1	a_1	a_0
0 1 1	0 0	0 0 0	a_0	0
1 0 0	1 $\bar{1}$	0 1 1	$\overline{a_0}$	1
1 0 1	1 0	1 0 0	$\overline{a_1}$	$\overline{a_0}$
1 1 0	1 $\bar{1}$	0 1 1	$\overline{a_1}$	$\overline{a_0}$
1 1 1	0 0	0 0 0	0	0

As per Table 1, when $b_5b_4b_3 = 111$, $-0 = 0$ can be used. Therefore, F_{20} can be expressed as follows:

$$F_{20} = \begin{cases} -1, & b_5b_4b_3 = 000, 001, 010, \text{or } 011 \\ 0, & b_5b_4b_3 = 100, 101, 110, \text{or } 111. \end{cases} \tag{10}$$

By setting PP^+_2 to all ones and adding +1 to the LSB of the partial product, F_{20} can then be determined only by b_5 as follows:

$$F_{20} = \begin{cases} -1, & b_5 = 0 \\ 0, & b_5 = 1 \end{cases} \tag{11}$$

A modified radix−4 Booth encoding and a decoding circuit for the partial product PP^+_2 are proposed here (Fig. 4); an extra three-input OR gate is then added to the design of [10] (Fig. 2). The three inputs of the additional OR gate are $\overline{b_5}$, $\overline{b_4}$, and $\overline{b_3}$. When $b_5b_4b_3 = 111$, it is clear that $\overline{b_5}\,\overline{b_4}\,\overline{b_3} = 000$, $p^+_{2i} = 1$, and PP^+_2 is set to all ones. So, E_{22} and F_{20} in ECW_2 are now determined by $b_7b_6b_5$ without b_4, b_3. Although the complexity is slightly increased compared with the previous design (Fig. 2), the delay stage remains the same.

In this work, Q^+_{19}, Q^+_{18}, Q^-_{21}, and Q^-_{20} are used to represent the modified partial products (i.e., replacing p^+_{19}, p^+_{18}, p^-_{21} and p^-_{20}). $q^-_{2(-2)}$, and $q^-_{2(-1)}$ are used to represent the additional partial products that are determined by F_{20}. As -1 can be coded as $\bar{1}11$ in RB format, E_{22} and F_{20} can be represented by E_2, $q^-_{2(-2)}$, $q^-_{2(-1)}$, (Fig. 3b) as follows:

$$E_2 = \begin{cases} E_{22}, & F_{20} = 0 \\ E_{22} - 1, & F_{20} = -1, \end{cases} \tag{12}$$

$$q^-_{2(-2)} = q^-_{2(-1)} = \begin{cases} 0, & F_{20} = 0 \\ 1, & F_{20} = -1. \end{cases} \tag{13}$$

As per Eq. (11) and Eq. (13), $q^-_{2(-2)}$, and $q^-_{2(-1)}$ can also be expressed as follows:

$$q^-_{2(-2)} = q^-_{2(-1)} = \overline{b_5}. \tag{14}$$

This is further explained by the truth table of E_{22}, F_{20} and E_2, $q^-_{2(-2)}$, $q^-_{2(-1)}$ (Table 3). Now ECW_2 only includes E_2 and $E_2 \in \{0, 1, \bar{1}\}$; E_2 can be incorporated into the modified partial products Q^+_{19}, Q^+_{18}, Q^-_{21} and Q^-_{20} by replacing p^+_{19}, p^+_{18} and p^-_{21}, p^-_{20} in

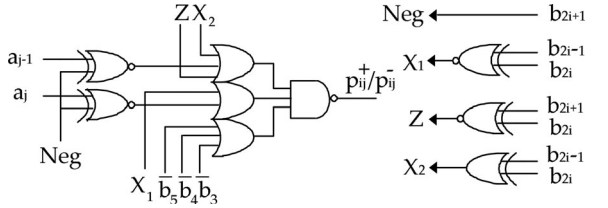

Fig. 4. The modified radix-4 Booth encoding and decoding scheme for PP^+_2.

TABLE 4
The Truth Table of Q_{19}^{+}, Q_{18}^{+}, Q_{21}^{-}, Q_{20}^{-}

$p_{19}^{+}p_{18}^{+}p_{21}^{-}p_{20}^{-}$	$Q_{19}^{+}Q_{18}^{+}Q_{21}^{-}Q_{20}^{-}$ ($E_2=0$)	$Q_{19}^{+}Q_{18}^{+}Q_{21}^{-}Q_{20}^{-}$ ($E_2=1$)	$Q_{19}^{+}Q_{18}^{+}Q_{21}^{-}Q_{20}^{-}$ ($E_2=-1$)
0100	0100	0101	0011
0101	0101	0110	0100
0110	0110	0111	0101
0111	0111	1000	0110
1000	1000	1001	0111
1001	1001	1010	1000
1010	1010	1011	1001
1011	1011	1100	1010

the shortest path Fig. 3c. From the truth table, E_2 can be determined by $b_7b_6b_5$ as follows:

$$E_2=\begin{cases}-1, & b_7b_6b_5=000, \text{or } 010\\ 1, & b_7b_6b_5=101\\ 0, & b_7b_6b_5=001, 011, 100, 110, \text{or } 111.\end{cases} \tag{15}$$

So the following three cases can be distinguished: 1) When $E_2=0$, Q_{19}^{+}, Q_{18}^{+}, Q_{21}^{-} and Q_{20}^{-} remain unchanged as: $Q_{19}^{+}=p_{19}^{+}$, $Q_{18}^{+}=p_{18}^{+}$, $Q_{21}^{-}=p_{21}^{-}$ and $Q_{20}^{-}=p_{20}^{-}$. 2) When $E_2=1$, a 1 is added to $p_{19}^{+}p_{18}^{+}p_{21}^{-}p_{20}^{-}$. 3) When $E_2=-1$, a 1 is subtracted from $p_{19}^{+}p_{18}^{+}p_{21}^{-}p_{20}^{-}$.

The relationships between Q_{19}^{+}, Q_{18}^{+}, Q_{21}^{-}, Q_{20}^{-} and p_{19}^{+}, p_{18}^{+}, p_{21}^{-}, p_{20}^{-} are summarized in Table 4. As the two MSBs of PP_1^{+} *i.e.*, p_{19}^{+} and p_{18}^{+} take complementary values as shown in Eq. (5), the operations of adding or subtracting a 1 will never incur in an overflow. Therefore, as per Eq. (15) and Table 4, the logic functions of Q_{19}^{+}, Q_{18}^{+}, Q_{21}^{-}, and Q_{20}^{-} can be expressed as follows:

$$\begin{aligned}Q_{19}^{+}=&(b_7\oplus b_5+b_7b_6b_5)\cdot p_{19}^{+}+\overline{b_7}\,\overline{b_5}\cdot(\overline{p_{18}^{+}+p_{21}^{-}+p_{20}^{-}}\oplus\\ &p_{19}^{+})+b_7\overline{b_6}b_5\cdot(p_{18}^{+}p_{21}^{-}p_{20}^{-}\oplus p_{19}^{+}),\end{aligned} \tag{16}$$

$$\begin{aligned}Q_{18}^{+}=&(b_7\oplus b_5+b_7b_6b_5)\cdot p_{18}^{+}+\overline{b_7}\,\overline{b_5}\cdot(\overline{p_{21}^{-}+p_{20}^{-}}\oplus p_{18}^{+})\\ &+b_7\overline{b_6}b_5\cdot(p_{21}^{-}p_{20}^{-}\oplus p_{18}^{+}),\end{aligned} \tag{17}$$

$$\begin{aligned}Q_{21}^{-}=&(b_7\oplus b_5+b_7b_6b_5)\cdot p_{21}^{-}+\overline{b_7}\,\overline{b_5}\cdot\overline{p_{21}^{-}\oplus p_{20}^{-}}\\ &+b_7\overline{b_6}b_5\cdot p_{21}^{-}\oplus p_{20}^{-},\end{aligned} \tag{18}$$

$$\begin{aligned}Q_{20}^{-}=&(b_7\oplus b_5+b_7b_6b_5)\cdot p_{20}^{-}+\overline{b_7}\,\overline{b_5}\cdot\overline{p_{20}^{-}}\\ &+b_7\overline{b_6}b_5\cdot\overline{p_{20}^{-}}.\end{aligned} \tag{19}$$

The delay of the RBMPPG-2 can be further reduced by generating Q_{19}^{+}, Q_{18}^{+}, Q_{21}^{-}, Q_{20}^{-} directly from the multiplicand A and the multiplier B. The relationships between p_{19}^{+}, p_{18}^{+} and A, B have been discussed in Section 2.2 as Eq. (5) and Eq. (6). The relationships between p_{21}^{-}, p_{20}^{-} and A, B are also shown in Table 3 according to the MBE scheme. Therefore, Q_{19}^{+}, Q_{18}^{+}, Q_{21}^{-}, and Q_{20}^{-} can be expressed as follows by replacing p_{19}^{+}, p_{18}^{+}, p_{21}^{-}, and p_{20}^{-} with the multiplicand bits (a_i) and the multiplier bits (b_i) after simplification:

$$\begin{aligned}Q_{19}^{+}=&\overline{\overline{b_1}b_0a_7+b_1\overline{a_7}}\left(\overline{\overline{b_5}\cdot\overline{b_7+b_6a_0+b_6a_1}}\right)\\ &+\left(\overline{b_1}b_0a_7+b_1\overline{a_7}\right)\cdot b_5\cdot b_7\overline{b_6}\,\overline{a_1}\;\overline{a_0},\end{aligned} \tag{20}$$

$$\begin{aligned}Q_{18}^{+}=&\overline{\overline{b_1}b_0a_7+b_1\overline{a_7}}\cdot\left(\overline{b_5}\cdot\overline{b_7+b_6a_0+b_6a_1}+b_5b_7\overline{b_6}\,\overline{a_1}\;\overline{a_0}\right)\\ &+\left(\overline{b_1}b_0a_7+b_1\overline{a_7}\right)\cdot[\overline{b_5}\cdot(b_7+b_6a_0+b_6a_1)\\ &+b_5\cdot\overline{b_7\overline{b_6}\,\overline{a_1}\;\overline{a_0}}],\end{aligned} \tag{21}$$

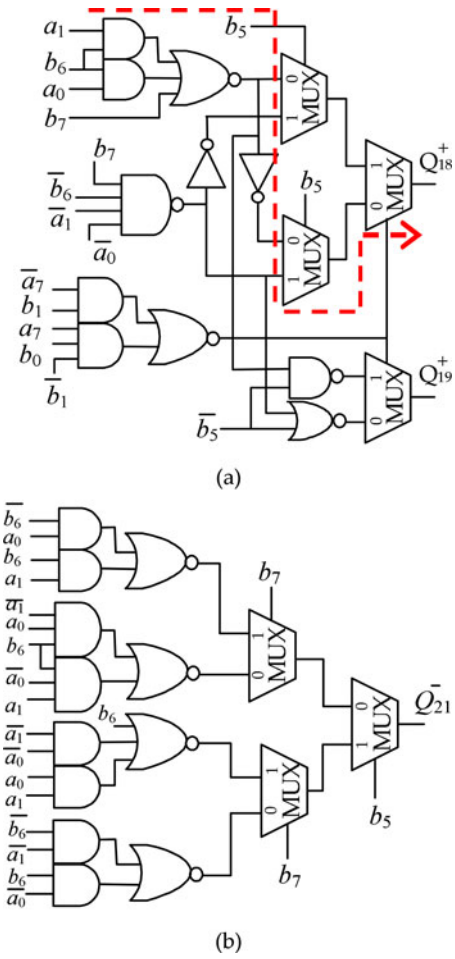

Fig. 5. The circuit diagram of the modified partial product variables: (a) Q_{18}^{+} and Q_{19}^{+}, (b) Q_{21}^{-}.

$$\begin{aligned}Q_{21}^{-}=&\overline{b_5}\cdot(\overline{b_7}\cdot\overline{b_6a_0\overline{a_1}+b_6\overline{a_0}a_1}+b_7\cdot\overline{\overline{b_6}a_0+b_6a_1})\\ &+b_5\cdot(\overline{b_7}\cdot\overline{\overline{b_6}\overline{a_1}+b_6\overline{a_0}}+b_7\cdot\overline{b_6+\overline{a_1}\;\overline{a_0}+a_1a_0}),\end{aligned} \tag{22}$$

$$Q_{20}^{-}=\overline{b_6}a_0+\overline{b_5}\overline{a_0}. \tag{23}$$

The circuit diagrams of the modified partial product variables Q_{18}^{+}, Q_{19}^{+} and Q_{21}^{-} are shown in Fig. 5. It is clear that Q_{18}^{+} has the longest delay path. It is well known that the inverter, the 2-input NAND gate and the transmission gate (TG) are faster than other gates. So, it is desirable to use TGs when designing the multiplexer [5], [6]. As shown in Fig. 5a, the critical path delay (the dash line) consists of a 1-stage AND-OR-Inverter gate, a one-stage inverter, and two-stage TGs. Therefore, RBMPPG-2 just increases the TG delay by one-stage compared with the MBE partial product of Fig. 2.

The above discussion is only an example; the above technique can be applied to design any 2^n-bit RB multipliers. It eliminates the extra $ECW_{N/4}$ and saves one RBPP accumulation stage, i.e., three XOR gate delays, while only slightly increasing the delay of the partial product generation stage. In general, an N-bit RB multiplier has $N/4$ RBPP rows using the proposed RBMPPG-2. The partial product variables $p_{1(N+1)}^{+}$, p_{1N}^{+}, $p_{(N/4)1}^{-}$ and $p_{(N/4)0}^{-}$ can be replaced by $Q_{1(N+1)}^{+}$, Q_{1N}^{+}, $Q_{(N/4)1}^{-}$, and $Q_{(N/4)0}^{-}$. The radix-4 Booth decoding of a PPR ($PP_{N/4}^{+}$) needs additional three-input OR gates (Fig. 4). Therefore, the extra $ECW_{N/4}$ is removed by the transformation of four partial product variables $Q_{1(N+1)}^{+}$, Q_{1N}^{+}, $Q_{(N/4)1}^{-}$, $Q_{(N/4)0}^{-}$ and one partial product row is saved in RB multipliers with any power-of-two word-length.

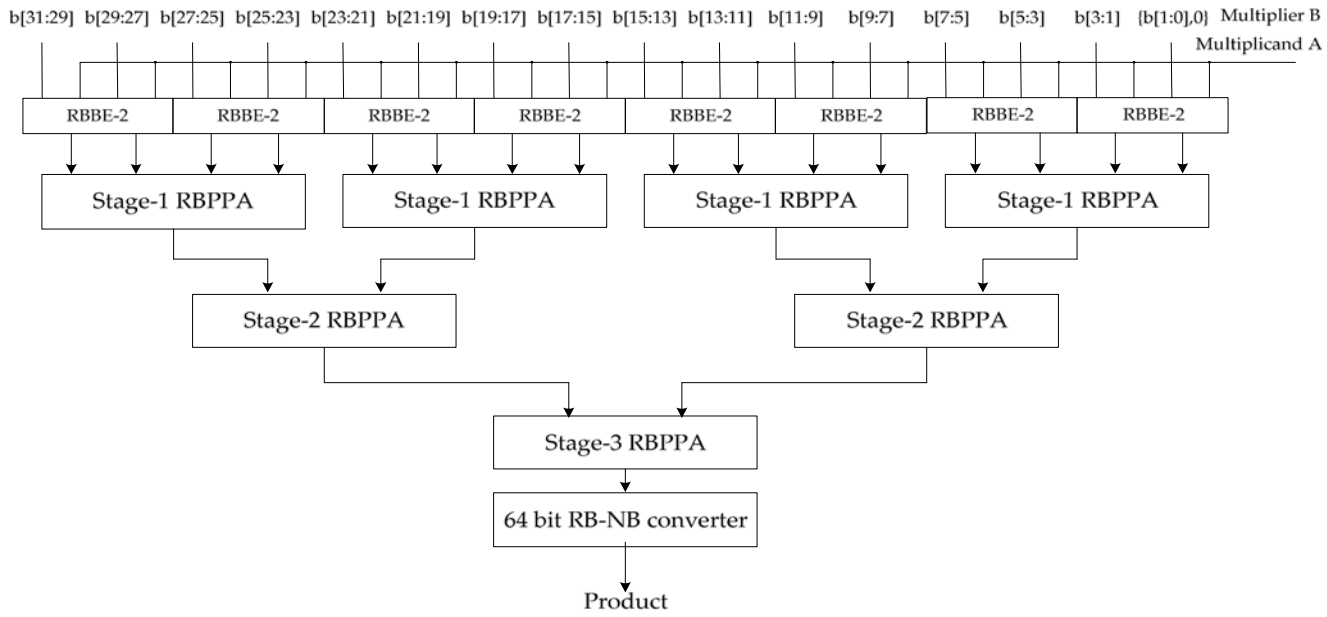

Fig. 6. The block diagram of a 32-bit RB multiplier using the proposed RBMPPG-2.

3.2 Design of RBMPPG-2-Based High-Speed RB Multipliers

The proposed RBMPPG-2 can be applied to any 2^n-bit RB multipliers with a reduction of a RBPP accumulation stage compared with conventional designs. Although the delay of RMPPG-2 increases by one-stage of TG delay, the delay of one RBPP accumulation stage is significantly larger than a one-stage TG delay. Therefore, the delay of the entire multiplier is reduced. The improved complexity, delay and power consumption are very attractive for the proposed design.

A 32-bit RB MBE multiplier using the proposed RBPP generator is shown in Fig. 6. The multiplier consists of the proposed RBMPPG-2, three RBPP accumulation stages, and one RB-NB converter. Eight RBBE-2 blocks generate the RBPP (p_i^+, p_i^-); they are summed up by the RBPP reduction tree that has three RBPP accumulation stages. Each RBPP accumulation block contains RB full adders (RBFAs) and half adders (RBHAs) [7]. The 64-bit RB-NB converter converts the final accumulation results into the NB representation, which uses a hybrid parallel-prefix/carry select adder [25] (as one of the most efficient fast parallel adder designs).

There are four stages in a conventional 32-bit RB MBE multiplier architecture; however, by using the proposed RBMPPG-2, the number of RBPP accumulation stages is reduced from 4 to 3 (i.e., a 25 percent reduction). These are significant savings in delay, area as well as power consumption. The improvements in delay, area and power consumption are further demonstrated in the next section by simulation.

Table 5 compares the number of RBPP accumulation stages in different 2^n-bit RB multipliers, i.e., 8×8-bit, 16×16-bit, 32×32-bit, 64×64-bit multipliers.

For a 64-bit multiplier, the proposed design has four RBPP accumulation stages; it reduces the partial product accumulation delay time by 20 percent compared with CRBBE-2 multipliers. Although both the proposed design and RBBE-4 have the same number of RBPP accumulation stages, RBBE-4 is more complex, because it uses radix-16 Booth encoding [14].

TABLE 5
Comparison of RBPP Accumulation Stages in RBPP Reduction Tree

Methods	64×64	32×32	16×16	8×8
CRBBE-2	5	4	3	2
RBBE-4 [14]	4	3	2	1
Proposed	4	3	2	1

4 Performance Evaluation

The performance of various 2^n-bit RB multipliers using the proposed RBMPPG-2 is assessed; the results are compared with NBBE-2, CRBBE-2 and RBBE-4 [14] multipliers that are the latest and best designs found in the technical literature.

All designs of RB multipliers use the RBFA and RBHA of [7]. An RB-NB converter is required in the final stage of the RB multiplier to convert the summation result in RB form to a two's complement number. It has been shown that the constant-time converter in [7] does not exist [19], [20], [21]. However, there is a carry-free multiplier that uses redundant adders in the reduction of partial products by applying on-the-fly conversion [22] in parallel with the reduction and generates the product without a carry-propagate adder [23], [24].

A hybrid parallel-prefix/carry-select adder [25] is used for the final RB-NB converter. The NBBE-2 multiplier design uses the same encoder and decoder as shown in Fig. 2. Four-two compressors [26], [27], [28] are used in the partial product reduction tree. The extra ECW in the NB multiplier designs is also modified as proposed in [11].

The multiplier designs are described at gate level in Verilog HDL and verified by Synopsys VCS using randomly generated input patterns; the designs are synthesized by the Synopsys Design Compiler using the NanGate 45 nm Open Cell Library. In the simulation of each design, a supply voltage of 1.25 V and room temperature are assumed. Standard buffers of a 2X strength are used for both the input drive and the output load. The option for logic structuring is turned off to prevent the tool from changing the structure of the unit cells. The average power consumption is found using the Synopsys Power Compiler with back annotated switching activity files generated from 2,500 random input vectors.

Table 6 summarizes the delay, area, power and power-delay product of the NB and RB multiplier designs; the delay, area, power and PDP metrics are compared separately.

Consider the delay first (Fig. 7); compared with CRBBE-2, the proposed designs can reduce the delay (for example up to 16.6

TABLE 6
Design Results of RB Multipliers (Using NanGate 45 nm Open Cell Library)

N-bits	NB and RB Multipliers	Delay (ns)	Area(μm^2)	Power (μW)	PDP (pJ)
8	NBBE-2	0.95	1,210	301	0.285
	CRBBE-2	1.20	1,322	485	0.582
	RBBE-4 [14]	1.32	1,071	546	0.721
	Proposed	1.00	1,258	496	0.496
16	NBBE-2	1.20	4,055	1,128	1.353
	CRBBE-2	1.48	4,165	1,549	2.293
	RBBE-4 [14]	1.62	3,897	2,498	4.047
	Proposed	1.26	4,004	1,500	1.890
32	NBBE-2	1.51	14,420	4,215	6.364
	CRBBE-2	1.79	13,925	3,227	5.776
	RBBE-4 [14]	2.09	14,454	5,745	12.007
	Proposed	1.57	13,589	3,090	4.851
64	NBBE-2	1.92	54,120	16,047	30.810
	CRBBE-2	2.29	48,624	11,852	27.141
	RBBE-4 [14]	2.47	55,119	20,517	50.677
	Proposed	2.05	47,903	11,199	22.958

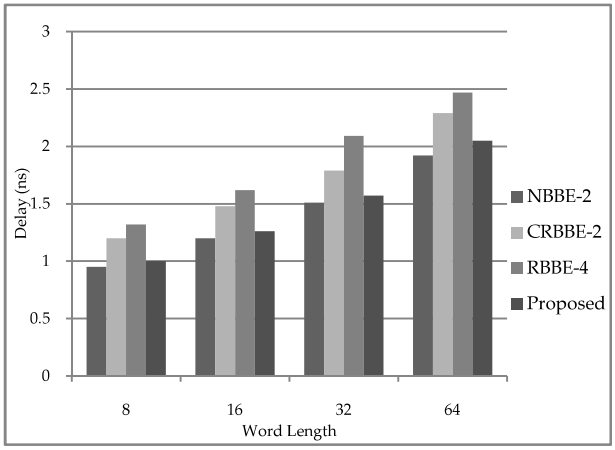

Fig. 7. Delay comparison of the NB and RB MBE multipliers at different word-lengths.

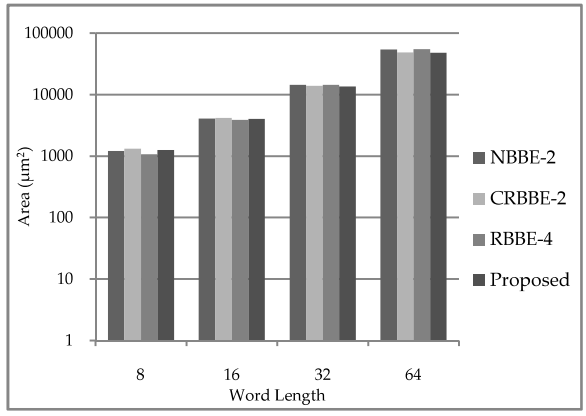

Fig. 8. Area comparison of the NB and RB MBE multipliers at different word-lengths.

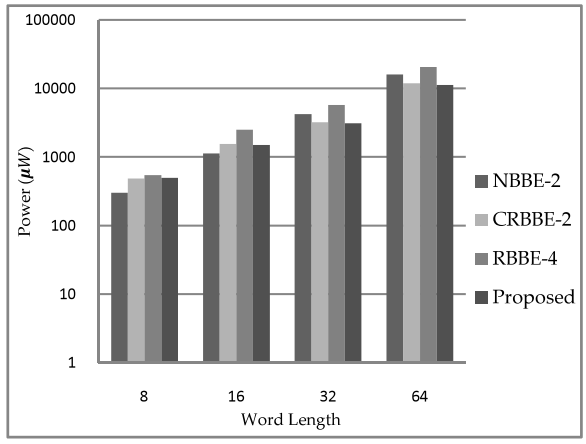

Fig. 9. Power comparison of the NB and RB MBE multipliers at different word-lengths.

percent for the case of 8×8-bit multiplier; for all cases of word-length, the delay is reduced by at least 10 percent. Compared with RBBE-4, the proposed designs can reduce the delay by up to 24.8 percent for the case of 32×32-bit and the delay is reduced by at least 17 percent for all cases of word-length. The delay improvement is achieved by the reduced critical path due to the elimination of one RBPP accumulation stage.

The delay of the proposed RB multipliers is slightly larger (approximately 5 percent) compared with the best NB multiplier, i.e., NBBE-2. However, its area and power are significantly lower than NBBE-2 for large word length designs (32- and 64-bit), as discussed next.

Compared with CRBBE-2, the RB multiplier using the proposed RBMPPG-2 has the smallest area for all cases (Fig. 8). For 8×8-bit and 16×16-bit multipliers, the area of RBBE-4 RB multipliers is smaller than that of the proposed RB multipliers because RBBE-4 based designs don't require extra ECW, while the area is slightly increased by the modified partial product in the proposed RB multipliers. Compared with NBBE-2 and RBBE-4, the proposed designs can reduce the area by up to 11.5 and 13.0 percent, respectively, for the case of a 64×64-bit multiplier and it is especially pronounced for large size designs, thus confirming the area efficiency of the proposed approach.

Power consumptions of NB and RB multipliers are also considered and compared (Fig. 9). The proposed designs can reduce the power for a 64×64-bit multiplier by up to 30.2, 5.5 and 45.4 percent, respectively, compared with NBBE-2, CRBBE-2, and RBBE-4.

PDP is a commonly used metric for combined performance in terms of delay and power consumption. In Fig. 10, the RB multipliers using the proposed RBMPPG-2 have the smallest PDP under all cases of RB multipliers. Compared with CRBBE-2, the proposed designs can reduce the PDP by over 14 percent for all cases. Compared with RBBE-4, the proposed designs can reduce the PDP by up to 59.6 percent for the case of a 32×32-bit multiplier, and overall cases the proposed designs can reduce the PDP by over 30 percent. Thus, these results confirm the proposed

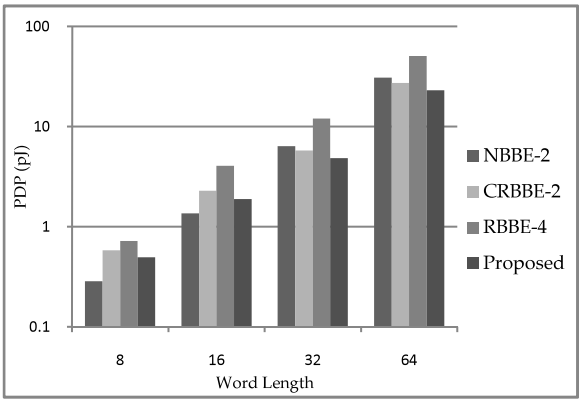

Fig. 10. PDP comparison of the NB and RB MBE multipliers at different word-lengths.

RBMPP-2 can be very useful for designing area and PDP efficient RB multipliers.

Although the PDP of the proposed design is larger than that of NBBE-2 for small sizes, it is much better for large word lengths. Compared with NBBE-2, the proposed designs reduce the PDP by 23.8 and 25.5 percent for 32×32-bit and 64×64-bit multipliers, respectively.

5 Conclusions

A new modified RBPP generator has been proposed in this paper; this design eliminates the additional ECW that is introduced by previous designs. Therefore, a RBPP accumulation stage is saved due to the elimination of ECW. The new RB partial product generation technique can be applied to any 2^n-bit RB multipliers to reduce the number of RBPP rows from $N/4+1$ to $N/4$. Simulation results have shown that the performance of RB MBE multipliers using the proposed RBMPPG-2 is improved significantly in terms of delay and area. The proposed designs achieve significant reductions in area and power consumption when the word length is at least 32 bits. The PDP can be reduced by up to 59 percent using the proposed RB multipliers when compared with existing RB multipliers. Hence, the proposed RBPP generation method is a very useful technique when designing area and PDP efficient power-of-two RB MBE multipliers.

Acknowledgments

This work is supported by grants from the National Natural Science Foundation of China (No. 61401197 and No. 61106029). The corresponding author of this paper is Weiqiang Liu.

References

[1] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, 1961.
[2] N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Comput.*, vol. C-34, no. 9, pp. 789–796, Sep. 1985.
[3] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high speed multiplier using a redundant binary adder tree," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 1, pp. 28–34, Feb. 1987.
[4] H. Edamatsu, T. Taniguchi, T. Nishiyama, and S. Kuninobu, "A 33 MFLOPS floating point processor using redundant binary representation," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 1988, pp. 152–153.
[5] H. Makino, Y. Nakase, and H. Shinohara, "A 8.8-ns 54x54-bit multiplier using new redundant binary architecture," in *Proc. Int. Conf. Comput. Des.*, 1993, pp. 202–205.
[6] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Makino, "An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 773–783, Jun. 1996.
[7] Y. Kim, B. Song, J. Grosspietsch, and S. Gillig, "A carry-free 54b×54b multiplier using equivalent bit conversion algorithm," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1538–1545, Oct. 2001.
[8] Y. He and C. Chang, "A power-delay efficient hybrid carry-lookahead carry-select based redundant binary to two's complement converter," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 1, pp. 336–346, Feb. 2008.
[9] G. Wang and M. Tull, "A new redundant binary number to 2's-complement number converter," in *Proc. Region 5 Conf.: Annu. Tech. Leadership Workshop*, 2004, pp. 141–143.
[10] W. Yeh and C. Jen, "High-speed booth encoded parallel multiplier design," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 692–701, Jul. 2000.
[11] S. Kuang, J. Wang, and C. Guo, "Modified Booth multiplier with a regular partial product array," *IEEE Trans. Circuits Syst. II*, vol. 56, no. 5, pp. 404–408, May 2009.
[12] J. Kang and J. Gaudiot, "A simple high-speed multiplier design," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1253–1258, Oct. 2006.
[13] F. Lamberti, N. Andrikos, E. Antelo, and P. Montuschi, "Reducing the computation time in (short bit-width) two's complement multipliers," *IEEE Trans. Comput.*, vol. 60, no. 2, pp. 148–156, Feb. 2011.
[14] Y. He and C. Chang, "A new redundant binary booth encoding for fast –bit multiplier design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 6, pp. 1192–1199, Jun. 2009.
[15] Y. He, C. Chang, J. Gu, and H. Fahmy, "A novel covalent redundant binary booth encoder," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, vol. 1, pp. 69–72.
[16] N. Besli and R. Deshmukh, "A novel redundant binary signed-digit (RBSD) Booth's encoding," in *Proc. IEEE Southeast Conf.*, 2002, pp. 426–431.
[17] A. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, pp. 236–240, 1951.
[18] O. MacSorley, "High-speed arithmetic in binary computers," *IRE Proc.*, vol. 49, pp. 67–91, 1961.
[19] Y. Kim, B. Song, J. Grosspietsch, and S. Gillig, "Correction to 'a carry-free 54b×54b multiplier using equivalent bit conversion algorithm'," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, p. 159, Jan. 2003.
[20] W. Rulling, "A remark on carry-free binary multiplication," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 159–160, Jan. 2003.
[21] M. Ercegovac and T. Lang, "Comments on 'a carry-free 54b×54b multiplier using equivalent bit conversion algorithm'," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 160–161, Jan. 2003.
[22] M. Ercegovac and T. Lang, "On-the-fly conversion of redundant to conventional representations," *IEEE Trans. Comput.*, vol. C-36, no. 7, pp. 895–897, Jul. 1987.
[23] M. Ercegovac and T. Lang, "Fast multiplication without carry-propagate addition," *IEEE Trans. Comput.*, vol. C-39, no. 11, pp. 1385–1390, Nov. 1990.
[24] L. Ciminiera and P. Montuschi, "Carry-save multiplication schemes without final addition," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 1050–1055, Sep. 1996.
[25] G. Dimitrakopoulos and D. Nikolos, "High-speed parallel- prefix VLSI Ling adders," *IEEE Trans. Comput.*, vol. 54, no. 2, pp. 225–231, Feb. 2005.
[26] S. Hsiao, M. Jiang, and J. Yeh, "Design of high-speed low-power 3–2 counter and 4–2 compressor for fast multipliers," *Electron. Lett.*, vol. 34, pp. 341–343, Feb. 1998.
[27] C. 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.
[28] D. Radhakrishnan and A. Preethy, "Low power CMOS pass logic 4–2 compressor for high-speed multiplication," in *Proc. IEEE Midwest Symp. Circuits Syst.*, 2000, vol. 3, pp. 1296–1298.