A New Redundant Binary Booth Encoding for Fast 2^n-Bit Multiplier Design

Yajuan He, *Student Member, IEEE*, and Chip-Hong Chang, *Senior Member, IEEE*

***Abstract*—The use of redundant binary (RB) arithmetic in the design of high-speed digital multipliers is beneficial due to its high modularity and carry-free addition. To reduce the number of partial products, a high-radix-modified Booth encoding algorithm is desired. However, its use is hampered by the complexity of generating the hard multiples and the overheads resulting from negative multiples and normal binary (NB) to RB number conversion. This paper proposes a new RB Booth encoding scheme to circumvent these problems. The idea is to polarize two adjacent Booth encoded digits to directly form an RB partial product to avoid the hard multiple of high-radix Booth encoding without incurring any correction vector. The proposed method leads to lower encoding and decoding complexity than the recently proposed RB Booth encoder. Synthesis results using Artisan TSMC 0.18-μm standard-cell library show that the RB multipliers designed with our proposed Booth encoding algorithm exhibit on average 14% higher speed and 17% less energy-delay product than the existing multiplication algorithms for a gamut of power-of-two word lengths from 8 to 64 b.**

***Index Terms*—Arithmetic circuit, Booth encoding algorithm, digital multiplier, energy-delay product, redundant binary adder (RBA).**

I. Introduction

THE digital multiplier is a ubiquitous arithmetic unit in microprocessors, digital signal processors, and emerging media processors [1]–[4]. It is also a kernel operator in application-specific data path of video and audio codecs, digital filters, computer graphics, and embedded systems [5]–[8]. Compared with many other arithmetic operations, multiplication is time-consuming and power hungry. The critical paths dominated by digital multipliers often impose a speed limit on the entire design. Hence, VLSI design of high-speed multipliers, with low energy dissipation, is still a popular research subject.

Redundant binary (RB) representation is one of the signed digit representations first introduced by Avizienis [9] in 1961 for fast parallel arithmetic. This new arithmetic was applied for fast multiplication by Takagi *et al.* [10] and implemented in VLSI by Edamatsu *et al.* [11]. The RB addition is carry-free, making it a promising substitute for two's complement multi-operand addition in a tree-structured multiplier [12]. Similar to a normal binary (NB) multiplier, an RB multiplier is anatomized into three stages and consists of four modules: the Booth encoder, RB partial product generator (also known as decoder), RB partial product accumulator, and RB-to-NB converter [11], [13]–[17]. The latter is required mainly for communicating the result to the peripheral devices which are largely designed based on the NB number system. The communications among RB adders across different stages of RB partial product summing tree are simpler than those of the full adders in a carry-save adder tree. In addition, the reduction rate of the redundant binary adder (RBA) summing tree is binary logarithmic to the number of RB partial products, which is particular beneficial to the generic power-of-two word size in computing.

Booth encoder and partial product generator affect the efficiency of the partial product generation. The number of partial products that can be saved by this stage impacts the cost, performance, and power consumption of the RB summing tree and the multiplier as a whole. Although the number of partial products can be reduced with a high-radix Booth encoder, the number of hard multiples that are expensive to generate also increases simultaneously [12]. In conventional RB multiplier design, a modified Booth encoding algorithm in NB regime is employed to reduce the number of partial products, and then pairs of NB partial products are encoded to form RB partial products. In this process, an additional constant binary vector is introduced to compensate for the aggregate errors resulting from both the RB and Booth encodings [13], [14], [16]. This correction vector incurs hardware overhead in the RB summing tree and, to a certain extent, offsets the regularity of the layout and increases switching activities.

As 8-, 16-, 32-, and 64-b operands are pervasively used in application-specific data paths and multimedia and very long instruction word (VLIW) processors [1], [2], [5]–[8], [18], this paper focuses on power-of-two word-length RB multipliers to exploit the binary logarithmic partial product reduction rate of the RBA summing tree. By scrutinizing the overheads of existing Booth encoding algorithms, the notion of covalent redundant binary Booth encoding was briefly introduced in [19]. This paper has corrected some shortfalls of our preliminary work [19]. The new multiplier circuits have been enhanced with a different RB coding and more efficient converters with in-depth circuit derivation, analysis and experimental result comparison. The proposed method overcomes the hard multiple generation problem of NB Booth encoders without incurring any correction vector. Compared with the RB Booth encoder [20], our proposed encoder generates the RB partial products more efficiently by consuming two RB digits for every RB partial product it generated. Consequently, our encoder and decoder are less

Manuscript received February 27, 2008; revised June 09, 2008. First published October 31, 2008; current version published June 19, 2009. This paper was recommended by Associate Editor V. De.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: heyajuan@pmail.ntu.edu.sg, echchang@ntu.edu.sg).

Digital Object Identifier 10.1109/TCSI.2008.2008503

complex for the same radix. We demonstrate that the proposed Booth encoder and decoder make high speed RB multipliers for power-of-two operand lengths.

The remainder of this paper is organized as follows. The existing Booth encoding algorithms and their overheads are briefly described in Section II. Section III presents the proposed covalent redundant binary Booth encoding (CRBBE) algorithm. This is followed by the realization of RB multipliers based on the CRBBE algorithm in Section IV to elaborate the design concept. The performance analysis of the proposed RB multiplier and the comparisons with its rivals are presented in Section V. Section VI concludes this paper.

II. Issues of Booth Encoding Algorithms for RB Multiplication

In fast digital multiplier design, modified Booth encoding algorithm is an efficient way to reduce the number of partial products by grouping consecutive bits in one of the two operands to form the signed multiples [21]. The operand that is Booth encoded is called the multiplier and the other operand is called the multiplicand. In this section, two major issues on using the modified Booth encoding algorithm for RB multiplication and some existing solutions are discussed.

A. Hard Multiples Problem

When modified Booth encoding [21] is applied to two's complement number, it is known as normal binary Booth Encoding (NBBE). In radix-r Booth-k encoding ($r = 2^k$), a signed digit c_i is generated from k adjacent binary bits $y_{k(i+1)-1}y_{k(i+1)-2}\cdots y_{ki+1}y_{ki}$ and a borrow bit y_{ki-1} as follows:

$$c_i = -2^{k-1} \times y_{k(i+1)-1} + \sum_{j=2}^{k} 2^{k-j} \times y_{k(i+1)-j} + y_{ki-1}, \quad \text{for } i = 0, 1, 2, \ldots, \left\lceil \frac{N}{k} \right\rceil - 1 \tag{1}$$

where k is an integer, $\lceil \alpha \rceil$ denotes the smallest integer value larger than or equal to α, N is the word length of the multiplier Y, and $y_{-1} = 0$.

As the radix number r of Booth-k encoder increases, the number of Booth encoded digits and hence the number of partial products decreases to approximately $1/k$ of the original number. However, as the number of multiples increases with the radix to $2^k + 1$, the number of hard multiples also increases simultaneously [12]. A hard multiple refers to a multiple that is not a power of two and thus cannot be obtained easily by simple shifting and/or complementation. Table I illustrates the radix-8 Booth encoding. The multiplier is partitioned into 4-b groups with an overlapping borrow bit between two adjacent groups. Each group is encoded in parallel to generate a select signal from the set $\{\pm 4M, \pm 3M, \pm 2M, \pm M, 0\}$. c_iM refers to the select signal for the partial product c_iX, where X is the multiplicand. The partial product $3X$ is a hard multiple, which can only be obtained by adding X and $2X$ by a carry propagation adder (CPA). The existence of hard multiple increases the latency of the multiplier as a whole because the generation of the partial products will not be accomplished until all the hard multiples are produced. Therefore, the advantage of using Booth encoding of radix-8 and above has been greatly offset because of the criticality of generating the hard multiples and the complexity of the decoding logic.

TABLE I
Radix-8 Normal Binary Booth Encoding (NBBE-3)

Multiplier Bits $y_{3i+2}y_{3i+1}y_{3i}(y_{3i-1})$	c_iM	Multiplier Bits $y_{3i+2}y_{3i+1}y_{3i}(y_{3i-1})$	c_iM
000(0)	$+0$	100(0)	$-4M$
000(1)	$+M$	100(1)	$-3M$
001(0)	$+M$	101(0)	$-3M$
001(1)	$+2M$	101(1)	$-2M$
010(0)	$+2M$	110(0)	$-2M$
010(1)	$+3M$	110(1)	$-M$
011(0)	$+3M$	111(0)	$-M$
011(1)	$+4M$	111(1)	-0

To speed up the generation of hard multiples in high-radix Booth encoding, a partially redundant biased Booth encoding (PRBBE) algorithm was proposed in [22]. Fig. 1 depicts the generation and negation of $3X$ hard multiple. It is generated in a partially redundant form by using a series of small length adders (4-b). The carry bit of each small length adder is not propagated but brought forward to the partial product summing tree. However, when the $3X$ multiple is negated, both the sum and the carry vectors need to be complemented and a "1" is added at the least significant bit (LSB) position. Therefore, the long strings of zeros between carries become strings of ones in the negative multiple. A properly selected biasing constant is introduced to revert the strings of ones into strings of zeros. The "1"s can be combined with the carry and sum bits to form a single compensation vector. The biasing constant of each such partial product introduces an extra compensation vector to the partial product summing tree.

B. Negative Multiples and NB-to-RB Partial Products Conversion Problem

Since negation in two's complement arithmetic requires carry propagation addition, negative partial product is more efficiently generated by the bit inversion of the multiplicand followed by the insertion of a "1" at its LSB position in the partial product summing tree. Therefore, one additional partial product row is generated to complete the two's complement negation of partial products for the negative multiples.

Furthermore, to accumulate the partial products in an RBA summing tree, the NB partial products generated by NBBE and PRBBE need to be converted to RB partial products. An NB number can be encoded into RB representation using either sign-magnitude [16], positive-negative [13], or positive-negative-complement [14] codings. The issue to be discussed here is illustrated by the positive-negative-complement coding adopted in this paper but it is also valid for other binary codings of signed digit set since the architecture designed for one code converter can be easily adapted to the other [23].

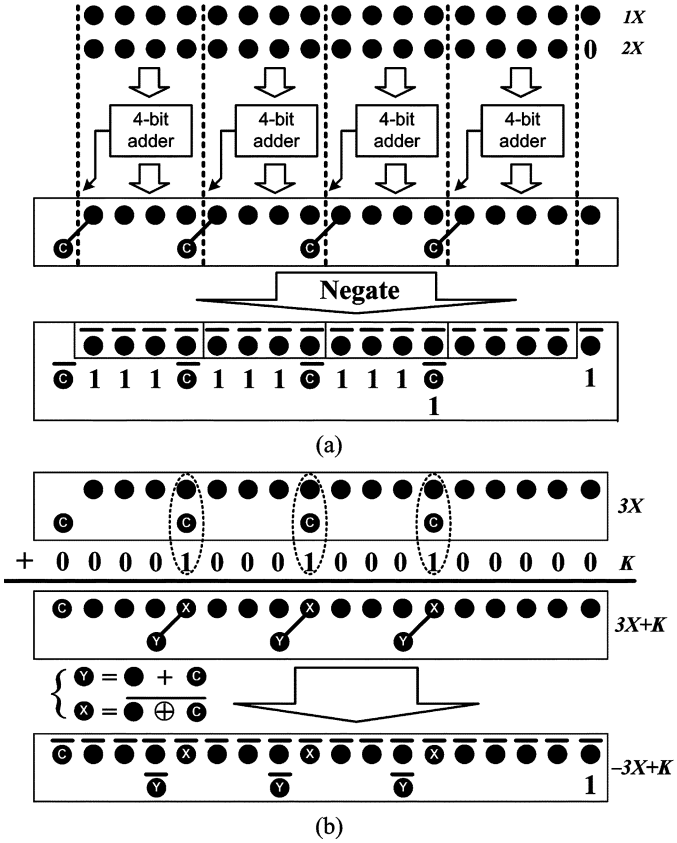


Fig. 1. $3X$ hard multiple generation and negation in partially redundant form [22].

In RB multiplication, the summation of two n-bit NB partial products $A = (a_{n-1}a_{n-2}\ldots a_0)_2$ and $B = (b_{n-1}b_{n-2}\ldots b_0)_2$ can be combined into a single n-digit RB number R by

$$R = A + B = A - (-B). \tag{2}$$

Since $-B = \overline{B} + 1$, substituting it into (2) gives

$$\begin{aligned} R &= A - (\overline{B} + 1) = A - \overline{B} - 1 \\ &= \left(-2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i\right) \\ &\quad - \left(-2^{n-1}\overline{b_{n-1}} + \sum_{i=0}^{n-2} 2^i \overline{b_i}\right) - 1 \\ &= -2^{n-1}(a_{n-1} - \overline{b_{n-1}}) + \sum_{i=0}^{n-2} 2^i (a_i - \overline{b_i}) - 1. \end{aligned} \tag{3}$$

As shown in Table II, an RB digit r can be encoded with two binary bits r^+ and r^- by

$$r = (r^+, r^-) = r^+ - \overline{r^-} \tag{4}$$

where $r^+, r^- \in \{0, 1\}$, and $r \in \{0, 1, \bar{1}\}$.

Therefore, according to (4), the terms $(a_i - \overline{b_i})$ in (3) can be encoded as $r_i = (a_i, b_i)$. To eliminate the hardware required for sign extension, the most significant digit term can be simply negated as $-(a_{n-1}, b_{n-1})$. From (4), it is noted that

$$-(r^+, r^-) = -r^+ + \overline{r^-} = (\overline{r^-}, \overline{r^+}). \tag{5}$$

TABLE II
POSITIVE–NEGATIVE-COMPLEMENT CODING

Coding (r^+, r^-)	RB Digit r
(0, 0)	$\bar{1}$
(0, 1)	0
(1, 0)	0
(1, 1)	1

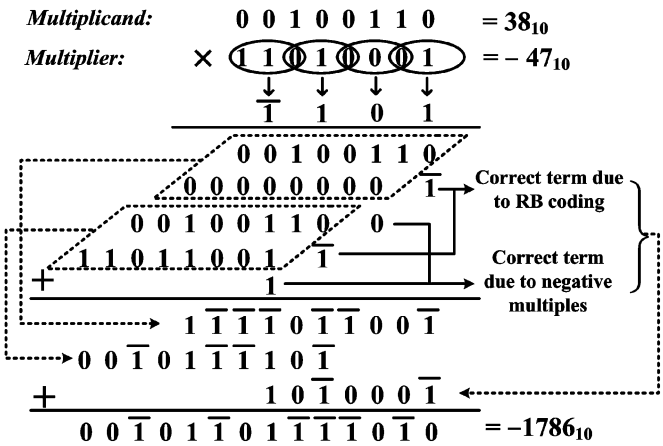


Fig. 2. Illustration of the correction vector generation on an 8 × 8-b multiplication with NBBE-2.

Since the positive-negative-complement coding is symmetric, r^+ and r^- is commutative and $(\overline{r^-}, \overline{r^+}) \equiv (\overline{r^+}, \overline{r^-})$. Therefore, R can be coded as follows:

$$R = (\overline{a_{n-1}}, \overline{b_{n-1}})(a_{n-2}, b_{n-2})\ldots(a_1, b_1)(a_0, b_0) + (0, 0). \tag{6}$$

From (6), it is clear that every RB partial product row thus composed requires one correction constant $(0,0) \equiv \bar{1}$ to be added by an RBA at its LSB position. All of the correction constants generated from the RB partial products, together with those constants from the negative multiples, can be accumulated to form a new RB partial product, collectively called the RB correction vector.

Fig. 2 exemplifies the correction vector generation procedure. It can be seen that NBBE-2 (radix-4 NB Booth encoding) generates three instead of two RB partial products for an 8 × 8-b multiplication. Owing to the absence of hard multiples, NBBE-2 is attractive especially for the short operand length multiplication. However, the additional delay required to add an extra partial product row critically slows down the short operand length multiplier due to the relatively lower number of adder stages in its partial product summing tree.

The RB correction vector incurs additional hardware for its accumulation. It can even increase the number of stages of the summing tree, if the word length of the multiplier is exactly 2^n, such as the 8-b and 16-b multipliers in application-specific data paths of multimedia and wireless applications [5], [6], and the multipliers for single extended and double extended floating point numbers, whose effective mantissa are 32 and 64 b, respectively [1]. Consequently, the power dissipation and worst case delay are also degraded by the inclusion of this correction vector.

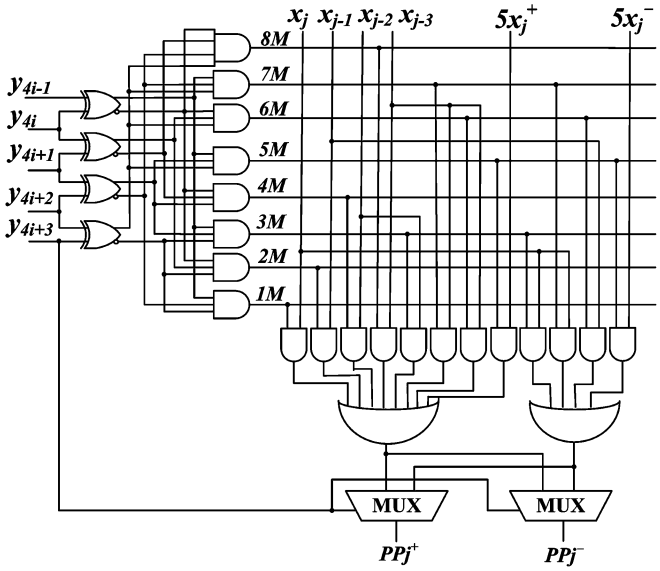


Fig. 3. Radix-16 RBBE and its partial product generator.

C. RB Booth Encoding (RBBE)

In [20], a method was proposed to obtain the hard multiples from the differences of two simple power-of-two multiples. Table III illustrates the radix-16 RB Booth encoding, where the multiplier bits are $y_{4i+3}y_{4i+2}y_{4i+1}y_{4i}(y_{4i-1})$ and each of the original hard multiples selected by $\pm 3M$, $\pm 6M$ and $\pm 7M$ are replaced by $\pm(4X-1X)$, $\pm(8X-2X)$, and $\pm(8X-1X)$, respectively. The partial products generated in this way conform to the format of the RB coding. The only exception is the hard multiples selected by $\pm 5M$, marked with "$*$" in Table III. As the $5X$ hard multiple can not be readily generated in this manner, a simple carry-free RB adder is suggested in [20] to add $4X$ and X. The advantage of this method is the correction vector due to the two's complement arithmetic and the RB coding has been completely eliminated. Comparing with NBBE, the ease of generating the hard multiples by RBBE, to a certain extent has been offset by its complex circuitry. High-radix RBBE requires high fan-in gates in the partial product generator circuit (see Fig. 3). Since the circuit for each digit of the RB partial product will be duplicated in a large number, the overhead of high fan-in gates is more prominent in long operand length multipliers. Besides, as only one Booth encoded digit is consumed for one RB partial product, half of the binary bits representing an RB partial product generated from a simple power-of-two multiple in the RBBE are filled with "0"s, which is rather inefficient.

D. Two's Complementation Method (TCM)

An innovative TCM was recently proposed in [18] to resolve the extra correction vector problem associated with the NBBE-2 algorithm. The TCM algorithm uses a divide-and-conquer approach to perform the two's complement conversion so that five signed partial products (0, $\pm 1X$, $\pm 2X$) are originated for selection. In this way, the correction vector due to the negative multiples in two's complement arithmetic can be eliminated. If TCM is used for the design of RB multiplier, the RB coding induced compensation constants can also be similarly circumvented. With TCM algorithm, the RB multiplier achieves exactly $\lceil N/4 \rceil$ RB partial products as opposed to $\lceil N/4 \rceil + 1$ in NBBE-2 multiplier. Besides, the multiplier is modular and more regularly structured. However, the worst case delay of the TCM algorithm is logarithmically proportional to the operand lengths ($O(\log N)$). Comparing with the constant delay time of conventional Booth encoding algorithms, the dependency of speed on word length of TCM algorithm is a limiting factor for large integer multiplication.

TABLE III
RADIX-16 REDUNDANT BINARY BOOTH ENCODING (RBBE-4)

Multiplier Bits	Multiple		Multiplier Bits	Multiple	
	$+M$	$-M$		$+M$	$-M$
0 0 0 0 (0)	0	0	1 0 0 0 (0)	0	$8M$
0 0 0 0 (1)	M	0	1 0 0 0 (1)	M	$8M$
0 0 0 1 (0)	M	0	1 0 0 1 (0)	M	$8M$
0 0 0 1 (1)	$2M$	0	1 0 0 1 (1)	$2M$	$8M$
0 0 1 0 (0)	$2M$	0	1 0 1 0 (0)	$2M$	$8M$
0 0 1 0 (1)	$4M$	M	1 0 1 0 (1)	0	$5M$ *
0 0 1 1 (0)	$4M$	M	1 0 1 1 (0)	0	$5M$ *
0 0 1 1 (1)	$4M$	0	1 0 1 1 (1)	0	$4M$
0 1 0 0 (0)	$4M$	0	1 1 0 0 (0)	0	$4M$
0 1 0 0 (1)	$5M$ *	0	1 1 0 0 (1)	M	$4M$
0 1 0 1 (0)	$5M$ *	0	1 1 0 1 (0)	M	$4M$
0 1 0 1 (1)	$8M$	$2M$	1 1 0 1 (1)	0	$2M$
0 1 1 0 (0)	$8M$	$2M$	1 1 1 0 (0)	0	$2M$
0 1 1 0 (1)	$8M$	M	1 1 1 0 (1)	0	M
0 1 1 1 (0)	$8M$	M	1 1 1 1 (0)	0	M
0 1 1 1 (1)	$8M$	0	1 1 1 1 (1)	0	0

* Hard multiples.

III. Covalent Redundant Binary Booth Encoding (CRBBE) Algorithm

A new Booth encoding algorithm is presented in this section to simplify the generation of hard multiples and reduce the number of RB partial products without introducing any form of correction vector. The proposed algorithm binds two adjacent Booth encoders to compose an RB partial product by exploiting the RB coding. The common bit of the two adjacent Booth encoders is used as an enabler for the polarization of two equally weighted partial product bits. As the formation of an RB partial product digit is analogous to the charge sharing of two oppositely charged atoms in a covalent bond, we name the algorithm the covalent redundant binary Booth encoding.

A. Radix-4 Covalent Redundant Binary Booth Encoding (CRBBE-2)

Fig. 4 shows the simplest radix-2 Booth encoded multiplier. From (1), the signed digit $d_i = -y_i + y_{i-1}$ is encoded from $y_i(y_{i-1})$, where the borrow bit is bracketed. Since the borrow bit from which d_{i+1} is encoded is the MSB of the binary bits from which d_i is encoded, not all combinations of two digits

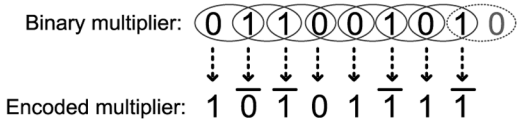


Fig. 4. Radix-2 Booth encoded multiplier.

TABLE IV
PERMISSIBLE DUPLET (d_{i+1}, d_i) IN RADIX-2 BOOTH ENCODED NUMBER

$d_{i+1}=1$	$d_{i+1}=0$	$d_{i+1}=\bar{0}$	$d_{i+1}=\bar{1}$
$(1,\bar{0})$	$(0,1)$	$(\bar{0},\bar{0})$	$(\bar{1},1)$
$(1,\bar{1})$	$(0,0)$	$(\bar{0},\bar{1})$	$(\bar{1},0)$

from $\{0,\bar{0},1,\bar{1}\}$ are permissible for any pair of adjacent digits in an encoded number. The following properties are observed.

1) Property 1: No two consecutive nonzero digits are of the same sign, i.e., $d_{i+1}\times d_i=-1$, $i\in[0,N-2]$, where d_{i+1} and d_i are two adjacent nonzero digits and N is the word length of radix-2 Booth encoded number.

Property 1 implies that the signs of the nonzero digits alternate in the encoded multiplier.

2) Property 2: Any zero between a leading 1 and a trailing $\bar{1}$ is a negative zero, $\bar{0}$.

Table IV shows all permissible combinations of two contiguous encoded digits d_{i+1} and d_i, which are grouped into four categories based on the left digit d_{i+1}. From the analysis of Section II, it is evident that if two adjacent NBBEs always generate signed digits of opposite polarity, their corresponding NB partial products can be directly combined to form a single positive-negative-complement coded RB partial product without any correction vector. This is only possible if contiguous digits of the Booth encoded multiplier alternate in signs. The duplets in the middle two columns of Table IV obviously do not fulfill this criterion.

Since the sign digit representation of a number is not canonic and the neutral polarity zero can be expressed in both positive and negative forms, we can map all possible duplets in Table IV to (p_l^+, p_l^-), such that one digit of the pair is positive and the other digit is negative without changing the compound multiple coefficient c_l

$$c_l = 2d_{i+1} + d_i = 2p_l^+ + p_l^- \tag{7}$$

where $p_l^+, p_l^- \in \{\pm 0, \pm 1\}$, and $\text{sign}(p_l^+) \neq \text{sign}(p_l^-)$. $l = 0, 1, \ldots, \lceil N/2 \rceil - 1$ and $i = 2l$.

The multiple $c_l X$ is an RB partial product composed from the two adjacent NB partial products, $2d_{i+1}X$ and d_iX. For ease of exposition, the digit pair, (p_l^+, p_l^-) is called a dipole and the mapping $\theta : (d_{i+1}, d_i) \rightarrow (p_l^+, p_l^-)$ is called polarization. For example, in the second column of Table IV, $(d_{i+1}, d_i) = (0,1)$ can be mapped to a dipole of either $(1,\bar{1})$ or $(\bar{0},1)$. Table V shows all the dipoles. The dipole allows an RB partial product $c_l X$ to be composed from the difference of two multiples in the partial product generator. Due to the symmetry, for every positive–negative dipole in the shaded column of Table V, there is always a corresponding negative-positive dipole in the unshaded column with their coefficients, c_l of (7) differ only in sign. This property can be used to reduce the hardware for the CRBBE

TABLE V
POLARIZATION OF (d_{i+1}, d_i) FOR RADIX-4 CRBBE

$d_{i+1}=1$	$d_{i+1}=0$	$d_{i+1}=\bar{0}$	$d_{i+1}=\bar{1}$
$(1,\bar{0})=(1,\bar{0})$	$(0,1)=(\bar{0},1)$	$(\bar{0},\bar{0})=(0,\bar{0})$	$(\bar{1},1)=(\bar{1},1)$
$(1,\bar{1})=(1,\bar{1})$	$(0,0)=(\bar{0},0)$	$(\bar{0},\bar{1})=(0,\bar{1})$	$(\bar{1},0)=(\bar{1},0)$

circuit so that only one selector logic of each distinct signed multiple magnitude needs to be generated. The positive–negative-complement encoded RB partial product corresponding to the dipole, (p_l^+, p_l^-) is denoted by $\left(pp_{l,j}^+, \overline{pp_{l,j}^-}\right)$ as

$$\left(pp_{l,j}^+, \overline{pp_{l,j}^-}\right) = pp_{l,j}^+ - pp_{l,j}^- \equiv \left(2p_l^+ + p_l^-\right)\cdot X_j \tag{8}$$

where $pp_{l,j}^+, pp_{l,j}^- \in \{0,1\}$, and the subscripts l and j are the indexes of the multiplier and the multiplicand bits, respectively.

A multiple in the unshaded column can be generated from its corresponding multiple in the shaded column by simply swapping the values of $pp_{l,j}^+$ and $pp_{l,j}^-$ without generating any correction vector.

Radix-4 CRBBE produces $\lceil N/2 \rceil$ RB partial products without the correction vector problems of NBBE and yet the encoder logic and RB partial product generator (RBPPG) circuit is simpler than RBBE of the same radix. It is interesting to note that radix-8 CRBBE can be created in a similar manner from binding two heterogenous Booth encoders. The encoded digits from a radix-2 and a radix-4 NBBE can be "polarized" to avoid the generation of all the hard multiples of radix-8. With a simple tweak, CRBBE can be easily extended to radix-16 to achieve even higher RB partial product reduction rate, which will be illustrated in Section III-B.

B. Radix-16 Covalent Redundant Binary Booth Encoding (CRBBE-4)

From (1), two contiguous radix-4 NBBE encoded digits, d_{i+1} and d_i share a common bit, $y_{k(i+1)-1}$ from the multiplier and it exhibits the following property.

1) Property 3: If the LSB $y_{k(i+1)-1}$ that encodes d_{i+1} is 0, d_i is nonnegative. Otherwise, if $y_{k(i+1)-1}=1$, d_i is nonpositive.

The above property is actually a generalization of Property 2. It indicates that, irrespective of the radix of Booth encoding, only restricted combinations of contiguous digit pairs from the set $\{\pm 0, \pm 1, \ldots, \pm(r/2)\}$ are permissible in an encoded number.

With this restriction on the legitimacy of the encoded digits, two contiguous digits, $d_{i+1}d_i$, of radix-4 Booth encoding can be similarly mapped from three contiguous bits $y_{2i+3}y_{2i+2}y_{2i+1}$ and $y_{2i+1}y_{2i}y_{2i-1}$ of the multiplier as shown in Table VI, where $i = 0, 1, \ldots, \lceil N/2 \rceil - 1$. In Table VI, all legitimate duplets are mapped to the dipoles (p_l^+, p_l^-) for $l = 0, 1, \ldots, \lceil N/4 \rceil - 1$ such that

$$c_l = 4d_{i+1} + d_i = 4p_l^+ + p_l^- \tag{9}$$

where the multiple $c_l X$ is an RB partial product composed from the two adjacent NB partial products, $4d_{i+1}X$ and d_iX, and $i = 2l$.

TABLE VI
POLARIZATION OF (d_{i+1}, d_i) FOR RADIX-16 CRBBE

$d_{i+1}=2$	$d_{i+1}=1$	$d_{i+1}=0$	$d_{i+1}=\bar{0}$	$d_{i+1}=\bar{1}$	$d_{i+1}=\bar{2}$
$(2,\bar{0})=(2,0)$	$(1,2)=(2,\bar{2})$	$(0,2)=(\bar{0},2)$	$(\bar{0},\bar{0})=(0,\bar{0})$	$(\bar{1},2)=(\bar{1},2)$	$(\bar{2},2)=(\bar{2},2)$
$(2,\bar{1})=(2,\bar{1})$	$(1,1)^*$	$(0,1)=(\bar{0},1)$	$(\bar{0},\bar{1})=(0,\bar{1})$	$(\bar{1},1)=(\bar{1},1)$	$(\bar{2},1)=(\bar{2},1)$
$(2,\bar{2})=(2,\bar{2})$	$(1,0)=(1,\bar{0})$	$(0,0)=(\bar{0},0)$	$(\bar{0},\bar{2})=(0,\bar{2})$	$(\bar{1},0)=(\bar{1},0)$	$(\bar{2},0)=(\bar{2},0)$
	$(1,\bar{0})=(1,\bar{0})$			$(\bar{1},\bar{0})=(\bar{1},0)$	
	$(1,\bar{1})=(1,\bar{1})$			$(\bar{1},\bar{1})^*$	
	$(1,\bar{2})=(1,\bar{2})$			$(\bar{1},\bar{2})=(\bar{2},2)$	

* Hard multiples.

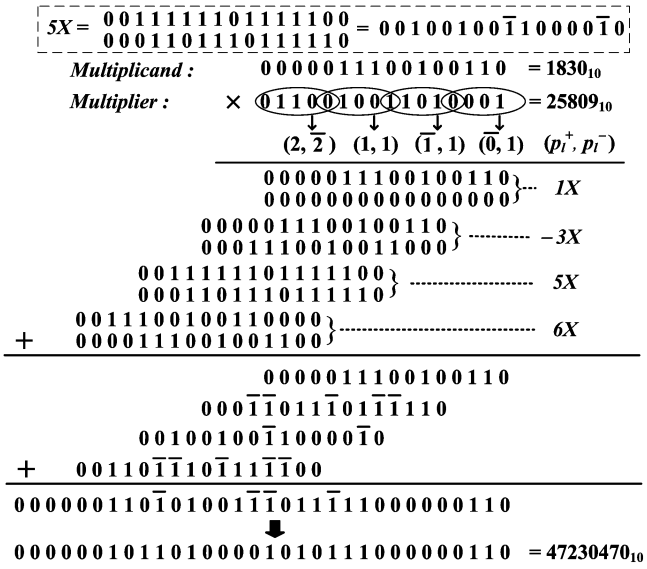


Fig. 5. 16 × 16-bit RB multiplication with CRBBE-4.

The positive–negative dipoles are listed in the shaded columns of Table VI while their negative–positive counterparts appear in the unshaded columns. The only exception is when $(d_{i+1}, d_i) = (1,1)$ and $(\bar{1},\bar{1})$. These two cases correspond to the special hard multiples, $\pm 5X$, which are marked with "*" in Table VI. This hard multiple can be generated using the dedicated carry-free RBA of [20]. It turns out that this RBA does not lie in the critical path of the CRBBE encoder. Thus, the RB partial product, $\left(pp_{l,j}^{+}, pp_{l,j}^{-}\right)$ generated by the dipole $\left(p_l^{+}, p_l^{-}\right)$ is expressed as follows:

$$\left(pp_{l,j}^{+}, \overline{pp_{l,j}^{-}}\right) = pp_{l,j}^{+} - pp_{l,j}^{-} = \left(4p_l^{+} + p_l^{-}\right) \cdot X_j. \qquad (10)$$

A detailed work-out example for a 16 × 16-bit multiplication based on CRBBE-4 algorithm is shown in Fig. 5. The generation of the hard multiple $5X$ by an RBA is shown at the top of the figure. Except the hard multiple (1,1), the three dipoles, $(2,\bar{2})$, $(\bar{1},1)$ and $(\bar{0},1)$ are used to generate the RB partial products $6X$, $-3X$ and $1X$, respectively.

The RB partial products reduction rate of radix-16 CRBBE is 1/4. A higher radix CRBBE algorithm can be similarly derived without introducing additional row of correction vector.

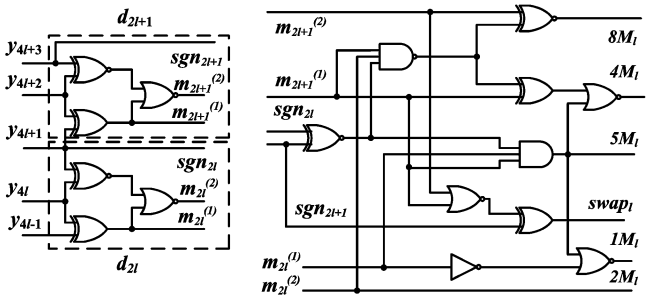


Fig. 6. Implementation of CRBBE-4 encoder.

Although most hard multiples for radix-32 can be more readily resolved than NBBE of the same radix, there exist some hard multiples which can not be generated as efficiently as the $5X$ multiple in this manner. Thus, CRBBE algorithm with $k \geqslant 5$ will not be pursued in this paper.

IV. DESIGN OF CRBBE-4-BASED RB MULTIPLIER

A. Realization of CRBBE-4 Algorithm

As discussed in Section III-B, CRBBE-4 is composed of two adjacent radix-4 Booth encoders. Its gate-level implementation is shown in Fig. 6(a), where the sign and magnitude of the radix-4 Booth encoded digit d_i are represented with three binary bits, sgn_i, $m_i^{(2)}$, and $m_i^{(1)}$ as follows:

$$d_i = (-1)^{sgn_i}\left(m_i^{(2)} + m_i^{(1)}\right). \qquad (11)$$

Fig. 6 shows the lth slice of a radix-16 CRBBE-4 circuit for the generation of the control signals, $c_l M_l$. The indexes i and l are related by $i = 2l$. The lower encoder takes three consecutive bits $y_{2i+1}y_{2i}y_{2i-1} \equiv y_{4l+1}y_{4l}y_{4l-1}$ from the multiplier to generate the magnitude bits $m_{2l}^{(2)}$ and $m_{2l}^{(1)}$ of d_i. Its sign bit $sgn_i = y_{4l+1}$. The upper encoder takes the binary bits $y_{2i+3}y_{2i+2}y_{2i+1} \equiv y_{4l+3}y_{4l+2}y_{4l+1}$, and generates the magnitude bits $m_{2l+1}^{(2)}$ and $m_{2l+1}^{(1)}$ of d_{i+1}. Its sign bit $sgn_{i+1} = y_{4l+3}$. All of these output signals are mapped by the polarization circuit as shown in Fig. 6(b). The control signals, $c_l M_l$ it generated are used to select the RB partial products correspond to the multiples, $c_l X$.

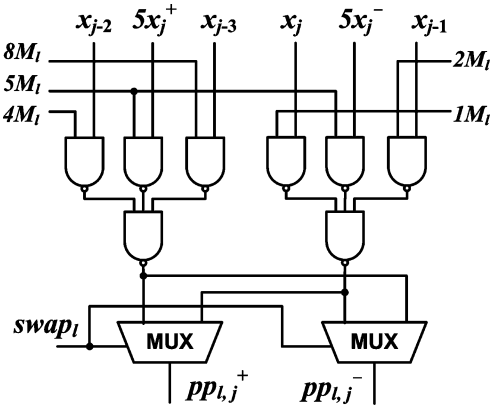


Fig. 7. RB partial product generator (RBPPG) of CRBBE-4.

The polarization circuit performs the mapping, $\theta : (d_{i+1}, d_i) \rightarrow (p_l^+, p_l^-)$. The control signals $1M_l$, $2M_l$, $4M_l$, and $8M_l$ are computed as follows:

$$1M_l = m_{2l}^{(1)} \cdot \overline{5M_l} \tag{12}$$

$$2M_l = m_{2l}^{(2)} \tag{13}$$

$$4M_l = \left(\overline{m_{2l+1}^{(1)} \cdot m_{2l}^{(2)} \cdot (sgn_{2l+1} \odot sgn_{2l})} \odot m_{2l+1}^{(1)}\right) \cdot \overline{5M_l} \tag{14}$$

$$8M_l = \overline{m_{2l+1}^{(1)} \cdot m_{2l}^{(2)} \cdot (sgn_{2l+1} \odot sgn_{2l})} \odot m_{2l+1}^{(2)}. \tag{15}$$

The special $5M_l$ multiple is generated by

$$5M_l = (sgn_{2l+1} \odot sgn_{2l}) \cdot m_{2l+1}^{(1)} \cdot m_{2l}^{(1)}. \tag{16}$$

The control flag, *swap* is used to exchange p_l^+ and p_l^- in the partial product generator to negate the selected RB partial product. When d_{i+1} is 0, the sign bit of d_{i+1} is complemented before it is used as an active high swap flag to the RBPPG. Otherwise, the original sign of d_{i+1} is used as the swap flag. Therefore, the *swap* signal can be generated by:

$$swap_l = \overline{\left(m_{2l+1}^{(1)} + m_{2l+1}^{(2)}\right)} \oplus sgn_{2l+1} \tag{17}$$

Fig. 7 shows a slice of the RBPPG circuit for the generation of the j-th digit of the l-th RB partial product, $pp_{l,j}^+$ and $pp_{l,j}^-$. Comparing with Fig. 3, the RBPPG circuit of CRBBE-4 is less complex.

B. Architecture of CRBBE-4 Based RB Multiplier

This section exemplifies the use of CRBBE-4 for the design of a 64×64-bit RB multiplier. Fig. 8 shows the block diagram of a 64×64-bit CRBBE-4 multiplier, which consists of three stages, Booth encoder and RBPPG, RBA summing tree and RB-to-NB converter.

In the first stage, 16 CRBBE-4 slices are used to generate the control signals from the multiplier. The $5X$ hard multiple is generated. The multiplicand bits are shifted and selected into 16 rows of RB partial products in 16 slices of RBPPG.

In the second stage, a 4-stage RBA summing tree is used to sum 16 RB partial products. Only the multi-digit RBA blocks, annotated with the number of RB partial product digits input to each block, are shown in Fig. 8. Each RBA block contains 64 RB full adder (RBFA) cells and a varying number of RB half adder (RBHA) cells depending on where it is located. The RBA block in the i-th level, designated RBA_i (i =1 to 4) contains 2^{i+1} RBHA cells in its most significant digit positions. The RBFA and RBHA cells modified from [14] are shown in Fig. 9. According to (10), due to the positive-negative-complement coding, the second binary bit, $pp_{l,j}^-$ of the RB partial product generated from CRBBE-4 and RBPPG circuit should be inverted before it is input to the RBA. In [13], a preprocessing circuit is needed for each RB digit to avoid the inconsistent representations of "0" prior to the RBA summing tree stage. An important benefit of the coding format adopted in this design is that these preprocessing circuits can be completely eliminated due to its symmetry.

An RB-to-NB converter converts the final accumulation result to NB representation. Due to the unequal delay profile of the final RB result bits, the conversion can be carried out in uneven groups of consecutive digits according to their arrival time. Groups of 4, 4, 8, 16 and 96 digits from the least significant digit position are evaluated concurrently. The first three groups of 4, 4, and 8 digits can be independently converted with ripple-carry adders to reduce the circuit complexity. The carry generation of the next group of 16 digits can be evaluated with a carry-lookahead adder as they do not depend on the final summation results in the RBA tree stage. Therefore, the conversion speed of the RB-to-NB stage depends solely on the conversion time of the most significant 96-digit group. This group is converted with a hybrid carry-lookahead/carry-select adder since it is widely known as one of the most efficient structures for fast parallel adder design [15], [24]–[26].

V. Performance Evaluations and Discussions

To examine the effects of using the proposed CRBBE-4 algorithm in RB multiplier design, the overall performance of the proposed radix-16 covalent redundant binary Booth encoded (CRBBE-4) multiplier is evaluated for power-of-two operand lengths varying from 8 to 64 b. The results are compared with the RB multipliers designed with radix-4, radix-8, radix-16 normal binary Booth encoding (NBBE-2, NBBE-3, NBBE-4) [12], radix-8 partially redundant biased Booth encoding (PRBBE-3) [22], radix-16 redundant binary Booth encoding (RBBE-4) [20], and the two's complementation method (TCM) [18] as reviewed in Section II. For a fair and legitimate comparison, the same RBA summing tree and RB-to-NB converter circuits are used for all the multipliers.

Each design is described at gate level in VHDL. The functionalities of the algorithms are verified by ModelSim for randomly generated input patterns. The designs are synthesized and mapped to Artisan TSMC 0.18-μm standard-cell library [27] using the Synopsys Design Compiler. All experiments are carried out at a supply voltage of 1.8V and a room temperature of 25 °C. Standard buffers of strength $2X$ are used for both input drive and output load. The option for logic structuring is turned off to prevent the tool from changing the structure of the unit

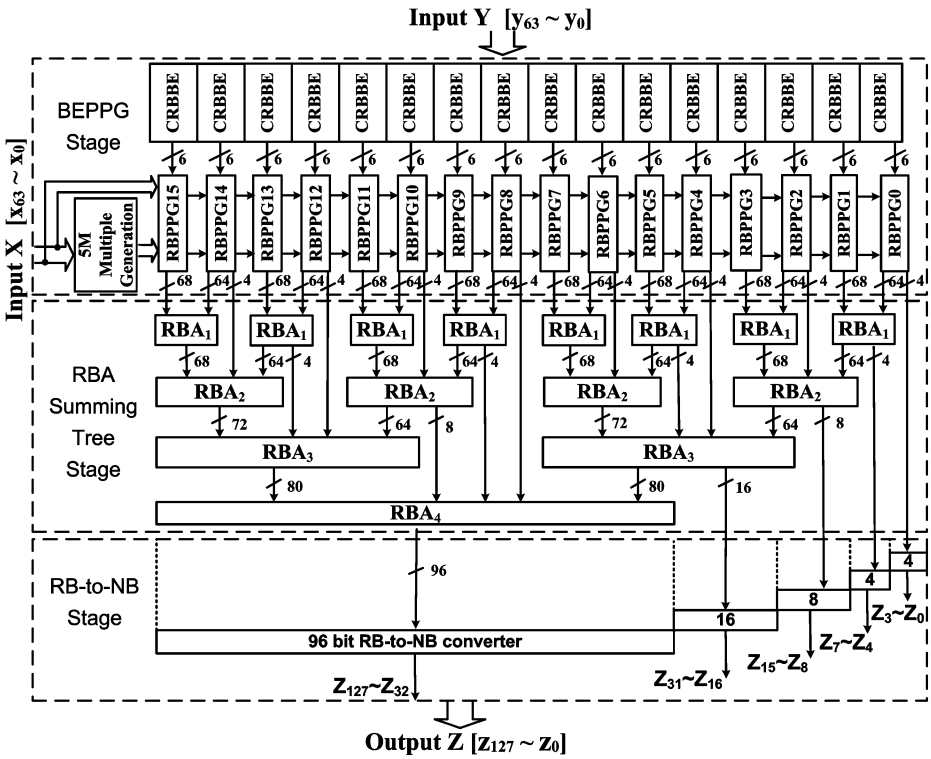


Fig. 8. Block diagram of 64 × 64-bit RB multiplier.

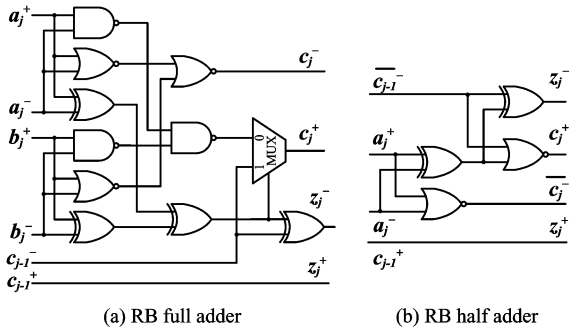


Fig. 9. Schematic of redundant binary adder (RBA).

TABLE VII
SYNTHESIS RESULTS OF DIFFERENT BOOTH ENCODED RB MULTIPLIERS

RB Multipliers	Delay (ns)				Energy Dissipation (pJ)			
	8*b*	16*b*	32*b*	64*b*	8*b*	16*b*	32*b*	64*b*
CRBBE-4	**1.588**	**2.159**	**2.969**	**3.938**	**5.084**	**16.25**	**58.37**	**237.02**
RBBE-4	1.712	2.396	3.246	4.295	5.646	19.07	68.03	259.87
NBBE-2	1.809	2.468	3.286	4.297	4.916	15.52	55.73	229.78
TCM	1.591	2.426	3.568	5.134	5.363	17.08	61.15	246.91
NBBE-3	2.143	2.743	3.511	4.672	5.480	16.54	53.51	205.19
PRBBE-3	1.885	2.502	3.301	4.419	5.544	17.02	56.63	220.96

cells. Each design is optimized for speed to obtain their minimum achievable delay. The average power consumptions are simulated by Synopsys Power Compiler with back annotated switching activity files generated from random input vectors to each design. It is clearly not feasible to estimate the power by exhaustive simulation of the circuit. To alleviate the pattern-dependence and dimension-dependence problems, the Monte Carlo statistical model [28] is adopted to obtain the mean power dissipation of each design with more than 99.9% confidence level that the error is bounded below 3%. The energy per operation of each design is obtained by dividing the average power dissipation by the input rate of the test vectors, which is the maximum frequency that each individual multiplier is capable to function. This index is used to compare the power consumptions of two designs working at different frequencies.

Table VII summarizes the worst case delay and energy dissipation of the RB multipliers. It shows that the proposed CRBBE-4 multiplier is the fastest design. On average, it is 8.50%, 10.68%, 12.82%, 19.58%, and 12.60% faster than RBBE-4, NBBE-2, TCM, NBBE-3, and PRBBE-3, respectively, with a penalty of 10.83% area overhead when compared to the most compact multiplier design.

To account for the good performance of CRBBE multiplier, the multiplier architectures are decomposed into three key constituent components and the stacked bar graphs showing the critical path delay distributions of all the multipliers for sizes $N = 8$, 16, 32, and 64 are analyzed in Fig. 10. The delays due to the Booth encoder and partial product generator (black), the RBA summing tree (gray) and the RB-to-NB converter (white) are clearly demarcated on each bar of the histograms in Fig. 10. Among these multipliers, CRBBE-4, RBBE-4, NBBE-2, and TCM have the same reduction rate of 1/4 in terms of the number of RB partial products. Due to the correction vector, the speed of NBBE-2 multiplier is obviously degraded by the additional stage in the partial product summation network. TCM is comparable to CRBBE-4 for small word lengths, however, its speed

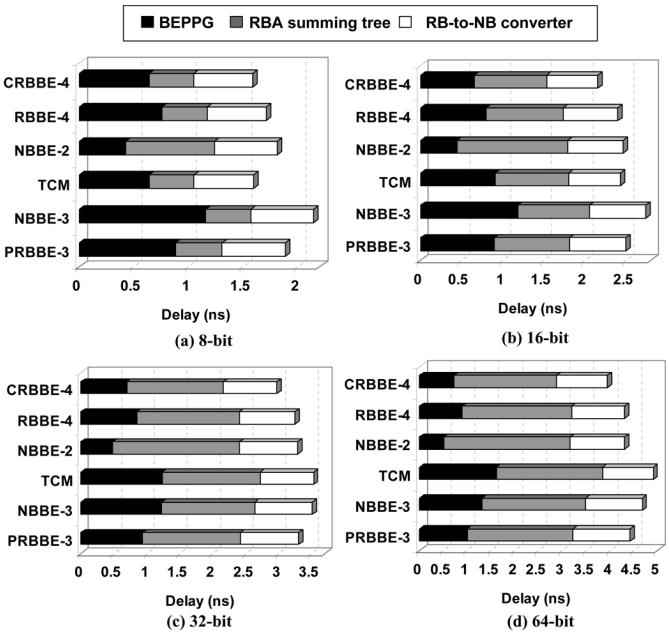


Fig. 10. Critical path delay distributions of different Booth encoded RB multiplier schemes.

TABLE VIII
ENERGY-DELAY PRODUCTS OF DIFFERENT BOOTH ENCODED RB MULTIPLIERS

RB Multipliers	EDP (fJ/MHz)			
	8*b*	16*b*	32*b*	64*b*
CRBBE-4	**8.073**	**35.08**	**173.30**	**933.38**
RBBE-4	9.666	45.69	220.83	1,116.1
NBBE-2	8.893	38.30	183.13	987.36
TCM	8.533	41.44	218.18	1,267.6
NBBE-3	11.74	45.37	187.87	958.65
PRBBE-3	10.45	42.58	186.94	976.42

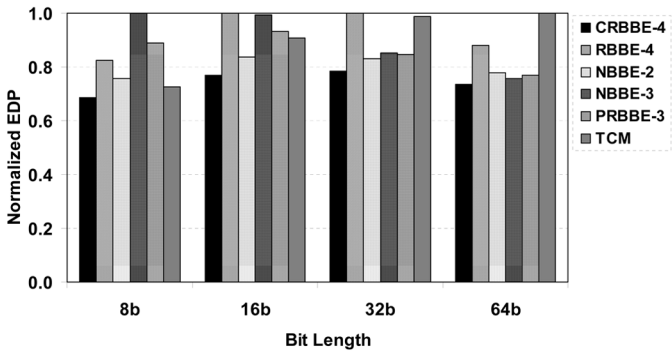


Fig. 11. Comparison of normalized EDP of different Booth encoded RB multipliers.

slows down when the multiplier size increases due to the logarithm time dependency on word length. Although RBBE-4 has no hard multiple and correction vector issues, its far more complex Booth encoder and partial product generator circuits result in an inferior performance to CRBBE-4. Compared with CRBBE-4, NBBE-3, and PRBBE-3 are able to reduce more RB partial products, making the effect of correction vector negligible. However, these higher radix Booth multipliers suffer from a more severe hard multiple problem due to the inevitable carry propagation in generating the hard multiples, which in turn make them slower. In practice, timing errors due to manufacturing variability is kept from the design process by the built-in implicit guard bands provided in the device and process models by the foundries and the library developers. Since the designs in comparison are synthesized with the same process model and cell library, this relative speed improvement of the proposed CRBBE-4 multiplier is a substantive merit despite the process variations.

From Table VII, the RB multiplier with NBBE-2 dissipates the least energy in 8- and 16-b multiplications due to the absence of hard multiple and its simplest Booth encoder and partial product generator circuits. For larger operand length of 32 and 64 b, NBBE-3 consumes the least energy among all NBBE multipliers in view of a better tradeoff between the complexity of the RBA summing tree and the number of CPAs required for the generation of hard multiples. Despite the lower complexity of Booth encoder and partial product generator of NBBE-2, its RBAs in the summing tree outnumber that of CRBBE-4, which accounts for the reduced ascendancy in energy dissipation. This is because the number of RBA stages of NBBE-2 is comparatively larger than that of CRBBE-4 due to its extra correction vector. The complexity of hard multiple generation and the extra partial product compensation terms of NBBE-3 and PRBBE-3 cause higher switching activities in the 8- and 16-b multipliers. As the operand length increases to 64 b, the energy consumption margin due to these overheads reduces and the switching activities become dominated by the complexity of the RBA summing tree. CRBBE-4 dissipates less energy than RBBE-4 and TCM for all word lengths. The better energy dissipation of CRBBE-4 over these two schemes is primarily due to the power reduction over a large number of partial product generators.

For the same rate of partial product reduction, it is interesting to note that with small length adders and additional compensation vector, PRBBE-3 multiplier achieves higher speed than NBBE-3 with a penalty of more energy dissipation. Therefore, if both speed and battery life need to be optimized simultaneously, the energy per operation has to be minimized in the same time as the delay. The energy-delay product (EDP) is a better metric than the energy per operation for benchmarking the energy efficiency of a circuit [29]. This metric makes the evaluation less sensitive to the reduction of either energy or delay by simply changing the supply voltage than optimizing circuit topology. The EDPs of all multipliers being compared are tabulated in Table VIII. For ease of comparison, the bar chart of the normalized EDP is plotted in Fig. 11, where the EDP for each operand length is normalized so that the multiplier with the largest EDP has an EDP of one. The results show that the proposed CRBBE-4 multiplier is the most energy efficient one. On average, its EDP is 19.40%, 7.11%, 16.91%, 16.08%, and 13.02% less than RBBE-4, NBBE-2, TCM, NBBE-3, and PRBBE-3, respectively.

VI. Conclusion

In this paper, a high-speed and energy-efficient RB multiplier designed based on a new covalent RB Booth encoding algorithm is presented. The idea is to polarize two adjacent Booth-encoded digits into a differential pair to restore the effective RB partial product reduction rate without the NB-to-RB conversion overhead. The proposed method fully exploits the characteristics of the positive–negative complement coding of RB number to directly generate an RB partial product from two adjacent Booth-encoded digits. Consequently, it shares the same advantages of RB Booth encoder for the ease of generating hard multiples and avoidance of error compensation vector, the two problems that are confronted by RB multiplier with normal binary Booth encoding. Six RB multipliers with different Booth encoding schemes have been prototyped for evaluation. The synthesis results show that the RB multiplier designed based on CRBBE-4 is the fastest and the most energy-efficient one among its rivals for the de facto power-of-two word lengths of computing.

References

[1] C. Shi, W. Wang, L. Zhou, L. Gao, P. Liu, and Q. Yao, "32B RISC/DSP media processor: MediaDSP3201," *SPIE Embedded Processors for Multimedia and Communications II*, vol. 5683, pp. 43–52, Mar. 2005.

[2] N. Slingerland and A. J. Smith, "Measuring the performance of multimedia instruction sets," *IEEE Trans. Comput.*, vol. 51, no. 11, pp. 1317–1332, Nov. 2002.

[3] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined fft processor," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 53, no. 7, pp. 585–589, Jul. 2006.

[4] H.-Y. Lee and I.-C. Park, "Balanced binary-tree decomposition for area-efficient pipelined fft processing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 4, pp. 889–900, Apr. 2007.

[5] O. T.-C. Chen, L.-H. Chen, N.-W. Lin, and C.-C. Chen, "Application-specific data path for highly efficient computation of multistandard video codecs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 1, pp. 26–42, Jan. 2007.

[6] M. Katona, A. Pizurica, N. Teslic, V. Kovacevic, and W. Philips, "A real-time wavelet-domain video denoising implementation in FPGA," *EURASIP J. Embedded Syst.*, pp. 1–12, 2006.

[7] S. Perri, P. Corsonello, and G. Cocorullo, "A 64-bit reconfigurable adder for low power media processing," *Electron. Lett.*, vol. 38, no. 9, pp. 397–399, Apr. 2002.

[8] A. Beaumont-Smith, J. Tsimbinos, C. C. Lim, and W. Marwood, "A VLSI chip implementation of an A/D converter error table compensator," *Computer Standard & Interfaces*, vol. 23, pp. 111–122, 2001.

[9] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Computers*, vol. EC-10, pp. 389–400, Sep. 1961.

[10] 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.

[11] H. Edamatsu, T. Taniguchi, T. Nishiyama, and S. Kuninobu, "A 33 MFLOPS floating point processor using redundant binary representation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, Feb. 1988, pp. 152–153.

[12] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*. New York: Oxford Univ. Press, 2000.

[13] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "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.

[14] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "A carry-free 54b $\times$ 54b multiplier using equivalent bit conversion algorithm," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1538–1545, Oct. 2001.

[15] Y. He and C. H. 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. 2, pp. 336–346, Feb. 2008.

[16] S. Kuninobu, T. Nishiyama, and T. Taniguchi, "High speed MOS multiplier and divider using redundant binary representation and their implementation in a microprocessor," *IEICE Trans. Electron.*, vol. E76-C, no. 3, pp. 436–445, Mar. 1993.

[17] S.-H. Lee, S.-J. Bae, and H.-J. Park, "A compact radix-64 54×54 CMOS redundant binary parallel multiplier," *IEICE Trans. Electron.*, vol. E85-C, no. 6, pp. 1342–1350, Jun. 2002.

[18] 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.

[19] Y. He, C. H. Chang, J. Gu, and H. A. H. Fahmy, "A novel covalent redundant binary Booth encoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'2005)*, Kobe, Japan, May 2005, vol. 1, pp. 69–72.

[20] N. Besli and R. G. Deshmukh, "A novel redundant binary signed-digit (RBSD) Booth's encoding," in *Proc. IEEE Southeast Conf.*, Columbia, SC, Apr. 2002, pp. 426–431.

[21] O. L. MacSorley, "High-speed arithmetic in binary computers," *IRE Proc.*, vol. 49, pp. 67–91, Jan. 1961.

[22] G. W. Bewick, "Fast multiplication: Algorithms and implementation," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1994.

[23] P. Kornerup, "Digit-set conversions: Generalizations and applications," *IEEE Trans. Comput.*, vol. 43, no. 5, pp. 622–629, May 1994.

[24] T. Lynch and E. E. Swartzlander, Jr, "A spanning tree carry lookahead adder," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 931–939, Aug. 1992.

[25] V. Kantabutra, "A recursive carry-lookahead/carry-select hybrid adder," *IEEE Trans. Comput.*, vol. 42, no. 12, pp. 1495–1499, Dec. 1993.

[26] Y. Wang, C. Pai, and X. Song, "The design of hybrid carry-lookahead/carry-select adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 1, pp. 16–24, Jan. 2002.

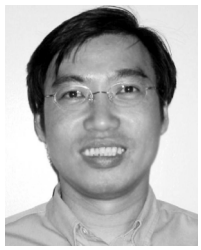
[27] *"TSMC 0.18 μm Process 1.8-Volt SAGE-X™ Standard Cell Library Databook,"* Artisan Components, Inc., 2001.

[28] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 1, pp. 63–71, Mar. 1993.

[29] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, Aug. 1997.

Yajuan He (S'05) received the B.S. degree in electronic science and technology from East China Normal University (ECNU), Shanghai, China, in 2001, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2008.

From 2001 to 2002, she was an RTP Digital IC Designer with the Institute of Microelectronics (IME), Singapore. In 2007, she joined STMicroelectronics, Singapore, as a Digital IC Design Engineer in the Smartcards Division, Asia-Pacific Design Center (APDC). Her research interests include computer arithmetic circuits and low-power IC design.



Chip-Hong Chang (S'92–M'98–SM'03) received the B.Eng. (Hons) degree from National University of Singapore, Singapore, in 1989, and the M.Eng. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

He served as a Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering, NTU, in 1999, where he is now an Associate Professor. He holds joint appointments at the university as Assistant Chair, Alumni, School of Electrical and Electronics Engineering since June 2008, Deputy Director of the Centre for High Performance Embedded Systems (CHiPES) since 2000, and Program Director of the Centre for Integrated Circuits and Systems (CICS) since 2003. His current research interests include low-power arithmetic circuits, application-specific digital signal processing, and digital watermarking for IP protection. He has published three book chapters and more than 130 research papers in refereed international journals and conferences.

Dr. Chang is a Fellow of the IET. He serves as an Editorial Advisory Board Member of *The Open Electrical and Electronic Engineering Journal*. He is listed in *Marquis Who's Who in the World* 2008 and 2009 and is appointed the Charter Fellow of Advisory Directorate International by the American Biographical Institute, Inc. (ABI).