Design and Analysis of Approximate Redundant Binary Multipliers

Weiqiang Liu, *Senior Member, IEEE,* Tian Cao, Peipei Yin, Yuying Zhu, Chenghua Wang, Earl E. Swartzlander, Jr., *Life Fellow, IEEE,* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract—As technology scaling is reaching its limits, new approaches have been proposed for computional efficiency. Approximate computing is a promising technique for high performance and low power circuits as used in error-tolerant applications. Among approximate circuits, approximate arithmetic designs have attracted significant research interest. In this paper, the design of approximate redundant binary (RB) multipliers is studied. Two approximate Booth encoders and two RB 4:2 compressors based on RB (full and half) adders are proposed for the RB multipliers. The approximate design of the RB-Normal Binary (NB) converter in the RB multiplier is also studied by considering the error characteristics of both the approximate Booth encoders and the RB compressors. Both approximate and exact regular partial product arrays are used in the approximate RB multipliers to meet different accuracy requirements. Error analysis and hardware simulation results are provided. The proposed approximate RB multipliers are compared with previous approximate Booth multipliers; the results show that the approximate RB multipliers are better than approximate NB Booth multipliers especially when the word size is large. Case studies of error-resilient applications are also presented to show the validity of the proposed designs.

Index Terms—Approximate computing, redundant binary (RB) multiplier, modified Booth encoder, RB compressor, RB-NB converter, partial product array, low power.

1 Introduction

As classic Dennard scaling is coming to an end, on-chip power consumption has become prohibitively high. Therefore, improvement in the performance of computing systems is encountering significant hurdles at the same power level. Recently, approximate computing has been proposed as a new approach for efficient low power design. In this context, efficiency refers to the generation of approximate results and comparable performance at a lower power consumption. Approximate computing can generate results that are good enough rather than always fully accurate. Approximate computing [1] is driven by applications that are related to human perception and inherent error resilience to include digital signal processing (DSP), multimedia, machine learning and pattern recognition [2]. Approximate computing can be applied to these applications due to the large and redundant data sets with significant noise, so numerical exactness can be relaxed. Approximate computing not only reduces power consumption, but also increases performance by reducing the critical path delay. Approximate techniques can be applied at several levels including circuits, architectures and software [3], [4]. The application of approximate computing to deep learning has also been studied [5]. At circuit level, the design of approximate arithmetic units has received significant research interest due to its importance in many computing applications. Typical applications, such as DSP and machine learning, require arithmetic computing in the form of addition (or accumulation) and multiplication. Addition has been extensively studied for approximate circuit implementations; various approximate adders have been proposed to attain reductions in power consumption and delay [6]. Current approximate adder designs include speculative adders [7], [8], [9] and non-speculative transistor-level full adders [10]. Approximate floating-point arithmetic has also been studied [11]. Multiplication is more complex than addition, because it requires the accumulation of the partial product (PP) rows. Approximate design techniques can be applied in four parts of a multiplier:

- Approximation of operands: Multiplication using approximate operands was first proposed by Mitchell with the concept of a logarithmic multiplier (LM) [12]. LM performs multiplication using only shifting and addition by converting the operands to approximate logarithmic numbers. Although the complexity of LM is significantly reduced compared with a conventional multiplier, it results in large errors. Recent designs of LMs aim to improve accuracy using fine piecewise linear approximation [13] or iterative techniques [14]. The use of approximate operands is further developed by an error-tolerant multiplier (ETM) [15] and a dynamic range unbiased multiplier (DRUM) [16]. ETM approximates the lower significant bits in the operand, such that all bits to the right position from the leading one are set to 1. DRUM

- *W. Liu, T. Cao, P. Yin, Y. Zhu and C. Wang are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China. E-mail: liuweiqiang, caotian, ppyin, zhuyuying, chwang@nuaa. edu.cn.*
- *E. E. Swartzlander, Jr. is with 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, 40125. E-mail: lombardi @ece.neu.edu.*

Corresponding authors: Weiqiang Liu and Fabrizio Lombardi.
Manuscript received xx, 2018; revised xx, 2018.

uses the significant segments of the operands, so the most significant k bits to perform multiplication. Generally, the approximation in operands introduces very large errors compared with other approximation techniques.

- Approximation of PP generation: An underdesigned multiplier (UDM) [17] is based on inaccurate 2×2 multipliers and was proposed by changing one entry of the Karnaugh-map (K-map). For larger size multipliers, the inaccurate 2×2 multipliers are used as basic units to generate approximate PPs that are accumulated with accurate adder trees. A generalized design of UDM has been further studied with carry-in prediction during the PP accumulation stage [18]. However, UDMs can only perform unsigned multiplication. Approximate Booth encoders have also been studied [19], [20], [21]. In [19], approximate radix-8 Booth multipliers have been proposed by using an approximate 2-bit adder that solves the hard multiple (×3) problem. Two efficient radix-4 approximate Booth encoders have been proposed in [21]. In [20], high-radix approximate Booth multipliers are proposed based on a hybrid radix encoding.
- Approximation of PP tree: The truncation scheme applied to a PP tree is usually used to truncate the lower part of the PPs or estimate the least significant PPs as a constant, this scheme is also referred to as a fixed-width multiplier design [22]. The error generated by the truncated PP rows can be rather large. Therefore, error compensation strategies have been proposed to increase the accuracy of truncated multipliers. An inexact array multiplier has been proposed by ignoring some of the least significant columns of the PPs as a constant [10]. In [23], a truncated multiplier has been proposed with a correction constant that is selected according to both the reduction and the rounding errors. However, this truncated multiplier has a large error if the PPs in the least significant columns are all ones or all zeros. Therefore, a truncated multiplier with variable correction has been proposed in [24]. Recently some error compensation strategies have been proposed to further improve the accuracy of fixed-width Booth multipliers [22], [25], [26]. The error is compensated with the outputs of Booth encoders in [22]. The error compensation circuit proposed in [25] mainly uses a simplified sorting network. To compensate for the quantization error of a fixed-width Booth multiplier, an adaptive conditional-probability estimator has been proposed in [26]. A so-called PP perforation [27] technique has been proposed and applied in the PP accumulation tree; successive rows and columns of PPs are removed before accumulation. An approximate Wallace tree has been used in an approximate Booth multiplier by ignoring the negation term in the (N/2+1) row to reduce the critical path [21].
- Approximation of compressors: Compressors or counters are widely used to accelerate the accumulation of PPs in the design of a high-speed multiplier [28], [29], [30], [31], [32]. An inexact 4:2 counter has been used to design an approximate 44 Wallace multiplier that is further used to build larger size multipliers [28]. Approximate 4:2 compressors have been proposed in [29] and used in a Dadda tree of 88 array multipliers. An 88 multiplier using approximate adders that ignore the carry propagation between PPs, has been proposed in [30]. Four multipliers are designed based on the approximate 4:2 compressors [31]. Improved approximate 4:2 compressors have been proposed in [32].

The design of approximate redundant binary (RB) multipliers is firstly studied in this work. RB multipliers use RB adder trees to perform a fast PP reduction [35]. Optimized RB multipliers show better performance in term of energy especially for wide word sizes compared with normal binary (NB) multipliers [36] due to the high modularity and carry-free addition during the PP reduction process. In this paper, radix-4 approximate RB multipliers (R4ARBM) are designed with approximate Booth encoders, approximate RB compressors and an approximate RB-NB converter, *i.e.* the additional novelty of this paper is to assess the compounding effect of multiple and diverse approximate circuits. Efficient approximate Booth encoders and approximate RB compressors are designed and analysed. A regular PP array has been achieved by either ignoring the last row of correction terms, or combining the correction terms into the PPs. By considering the error characteristics from both PP generation and accumulation, NOR-gate based approximate adders are applied in the approximate RB-NB converters to further improve the design of the approximate RB multipliers. Error analysis and hardware evaluation are presented to validate the proposed RB multiplier designs. Case studies with R4ARBM applied to FIR filtering and high dynamic range (HDR) image processing are also provided. This paper has been extended significantly from its previous conference version [37]. The main differences are summarized as follows:

- A new approximate Booth encoder with six errors in the K-map is proposed;
- A new approximate RB 4:2 compressor at a smaller complexity is proposed;
- For small approximate factors, rather than achieving a regular PP array by ignoring the correction term, an exact regular PP array is designed by combining the correction terms into the PPs using logic optimization for more accurate results;
- New approximate RB-NB converters are proposed;
- Case studies are provided with applications to FIR filters, k-mean clustering and HDR image processing.

The paper is organized as follows: Section 2 reviews the exact RB multipliers. The design of approximate RB multipliers is presented in Section 3. Section 4 evaluates the proposed designs with an error analysis and hardware results. Comparison with previous approximate Booth multipliers is also given in this section. Section 5 presents the application of approximate RB multipliers to FIR filters, k-mean clustering and HDR image processing. Section 6 concludes this paper.

2 BACKGROUND

Multiplication using a RB multiplier includes three steps. In the first step, a RB Booth encoder (RBBE-2) generates the PPs, in which the operands are converted from NB to RB. In the second step, all RB PPs are accumulated by a PP reduction tree (PPRT) using RB 4:2 compressors. Finally, in the third step, the RB-NB converter (*i.e.*, a fast adder) adds the two remaining PP rows. In the second step, there are several compression stages. The overall structure of an 8-bit RB multiplier is shown in Fig. 1. The basic principle of a RB multiplier is to use the RB representation during the PP reduction, such that accumulation is carry free. The design of an exact RB multiplier is reviewed in detail next.

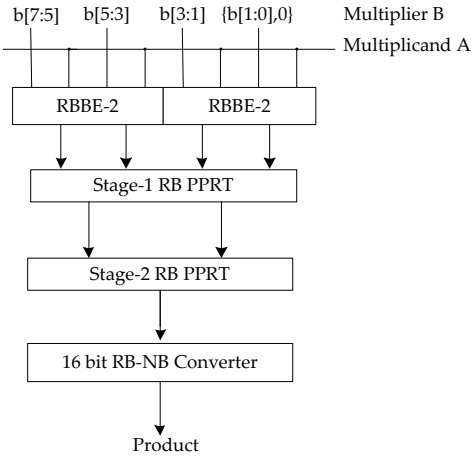


Fig. 1. Overall structure of an 8-bit RB multiplier.

2.1 Review of Radix-4 Booth Encoder

2.1.1 Conventional Modified Booth Encoder (MBE)

The Booth algorithm has been used to improve the sign correction issues of signed number multiplication [38]; however, the original Booth algorithm does not reduce the number of PPs. A Modified Booth Encoding (MBE) method (also known as the radix-4 Booth algorithm) has been further proposed. It reduces the number of PP rows by half. The complexity of the parallel multiplier is reduced significantly by applying MBE. The power consumption and the delay of the entire multiplier are also reduced. Let $A = a_{N-1}a_{N-2}\cdots a_2a_1a_0$ be the multiplicand and $B = b_{N-1}b_{N-2}\cdots b_2b_1b_0$ be the multiplier. The multiplier bits are encoded; so they are grouped in sets of three adjacent bits. The two side bits overlap with neighboring groups, except the first multiplier bit group. As per the encoded results from A, the Booth decoders select -2A, -A, 0, A, or 2A to generate the PP rows. 2A is obtained by a simple 1-bit left shift of the multiplicand. The negation operation is achieved by inverting each bit of A and adding 1 at its least significant bit (LSB) position. This is referred to as the correction term in this work. Therefore, the PP of each line can be easily generated by either shifting or inverting the multiplicand bits. The circuit diagram of the MBE scheme is shown in Fig. 2. Table 1 shows the K-Map of a conventional MBE. Therefore, the output of the Booth encoder pp_{ij} is given as follows:

$$\begin{aligned} pp_{ij} =&(b_{2i} \oplus b_{2i-1})(b_{2i+1} \oplus a_j) + \overline{(b_{2i} \oplus b_{2i-1})} \\ &(b_{2i+1} \oplus b_{2i})(b_{2i+1} \oplus a_{j-1}) \end{aligned} \quad (1)$$

The correction term for the negation operation is as follows:

$$E_i = b_{2i+1}\overline{b_{2i}} + b_{2i+1}\overline{b_{2i-1}} \quad (2)$$

2.1.2 New MBE (NMBE)

As per Eq. (2), the correction term (i.e., E_i) of the negation operation is almost equal to the MSB of the multiplier except when $b_{2i+1}b_{2i}b_{2i-1} = 111$. E_i can be further simplified by reconsidering this entry in the MBE truth table. In [36], it is observed that all the entries in the 6th column of Table 1 can be changed to 1 to achieve a simplified E_i' along with a slight increase in complexity of a pp_{ij}' as follows:

$$\begin{aligned} pp_{ij}' =&(b_{2i} \oplus b_{2i-1})(b_{2i+1} \oplus a_j) + (b_{2i} \oplus b_{2i-1}) \\ &(b_{2i+1} \oplus b_{2i})(b_{2i+1} \oplus a_{j-1}) + b_{2i+1}b_{2i}b_{2i-1} \end{aligned} \quad (3)$$

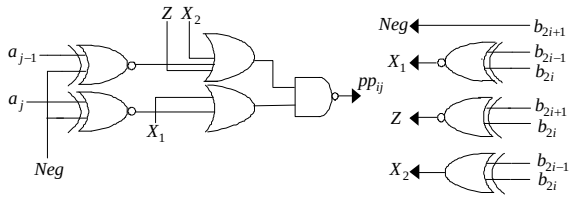


Fig. 2. MBE scheme: encoder and decoder [39].

TABLE 1
K-Map of Conventional MBE

a_ja_{j-1} \ $b_{2i+1}b_{2i}b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	0	1	1
01	0	0	1	0	1	0	1	0
11	0	1	1	1	0	0	0	0
10	0	1	0	1	0	0	0	1

TABLE 2
K-Map of New MBE

a_ja_{j-1} \ $b_{2i+1}b_{2i}b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	1	1	1
01	0	0	1	0	1	1	1	0
11	0	1	1	1	0	1	0	0
10	0	1	0	1	0	1	0	1

$$E_i' = b_{2i+1} \quad (4)$$

This observation is based on the property that the original zeros in this column can be obtained by adding 1 to the revised column with 1 (shown in Table 2). The circuit diagram of the new MBE (NMBE) scheme is shown in Fig. 3.

2.2 Review of RB PP Generator

The redundant binary (RB) representation is one of the signed-digit number representations. It is used for fast PP reduction due to its high modularity and carry-free feature. The RB representation can simplify interconnections, because the RB PPs can be added up by the RB adders with

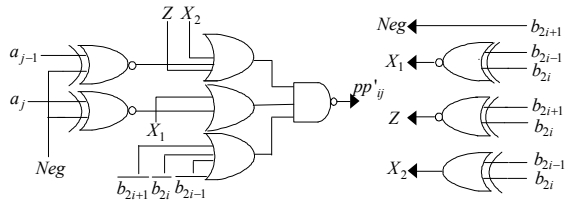


Fig. 3. The NMBE encoder and decoder [36].

TABLE 3
RB Encoding Used in This Work [40]

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

no continuous carry. In the RB signed-digit representation, the RB digit set $\{1, 0, \overline{1}\}$ can be encoded by using two NB bits and represented by the normal binary (NB) bit pair (X_i^+,X_i^-). RB numbers can be coded in several ways [35], [40]. The RB encoding shown in Table 3 [40] is used in this work. It follows the commutative law. An RB digit is given by:

$$X_i = X_i^+ + X_i^- - 1 \tag{5}$$

As two NB bits (i.e., X_i^+ and X_i^-) are used to represent one RB digit, a RB PP is generated from two NB PPs [35]. The addition of two N-bit NB PPs X and Y using two's complement representation is expressed as follows:

$$\begin{aligned} X + Y &= X - \overline{Y} - 1 \\ &= (-x_N 2^N + \sum_{i=0}^{N-1} x_i 2^i) - (-\overline{y_N} 2^N + \sum_{i=0}^{N-1} \overline{y_i} 2^i) - 1 \\ &= -(x_N - \overline{y_N}) 2^N + \sum_{i=0}^{N-1} (x_i - \overline{y_i}) 2^i - 1 \\ &= (X, \overline{Y}) - 1 \end{aligned} \tag{6}$$

where, $\overline{Y}$ is the inverse of Y, the composite number (X,$\overline{Y}$) can be interpreted as a RB number. The RB PP is generated by inverting the MSB of Y and adding -1 to the LSB. As the two MSBs of X are sign extension bits and are inverses of each other, the inverter is the only hardware overhead for the RB PP generation compared with the NB PP generation.

Both MBE and RB coding schemes introduce errors and two correction terms are required: 1) 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 PPs; 2) when the NB number is converted to a RB format, -1 must be added to the LSB of the RB number. These correction terms compensate for errors from both the Booth encoding and the RB encoding.

The conventional PP generation architecture of an exact 8-bit RB multiplier is shown in Fig. 4, where B is encoded, b_p denotes the bit position, p_{ij}^- or p_{ij}^+ is generated by using the Booth encoder, E_i is the correction term from the Booth

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A								a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
B									b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
PP_0^-						$\overline{p_{09}^-}$	p_{09}^-	p_{08}^-	p_{07}^-	p_{06}^-	p_{05}^-	p_{04}^-	p_{03}^-	p_{02}^-	p_{01}^-	p_{00}^-
PP_0^+						$\overline{p_{08}^+}$	p_{07}^+	p_{06}^+	p_{05}^+	p_{04}^+	p_{03}^+	p_{02}^+	p_{01}^+	p_{00}^+	0	0
PP_1^-		$\overline{p_{19}^-}$	p_{19}^-	p_{18}^-	p_{17}^-	p_{16}^-	p_{15}^-	p_{14}^-	p_{13}^-	p_{12}^-	p_{11}^-	p_{10}^-	1	$\mathbf{E}_1$	1	$\mathbf{E}_0$
PP_1^+		$\overline{p_{18}^+}$	p_{17}^+	p_{16}^+	p_{15}^+	p_{14}^+	p_{13}^+	p_{12}^+	p_{11}^+	p_{10}^+	0	0	0	1	0	0
										$\mathbf{E}_3$	1	$\mathbf{E}_2$			
										1	0	0			

Fig. 4. RB PP generation of an 8-bit RB multiplier using a Booth encoder.

encoding and a_7 is a sign bit. As per MBE, when the PP p_{ij}^- or p_{ij}^+ is 2A, a_i is shifted left by 1-bit position.

So, a_7 is lost as a_6 is left shifted. To avoid losing this sign bit, an additional bit a_8 is used to keep the left shifted a_7. The extra a_8 does not change the original value of the multiplicand A.

2.3 Review of RB 4:2 Compressor

To accumulate the RB PPs, RB adders (RBAs) (including RB full and RB half adders) are used in the RB compression tree. As a RBA adds two RB operands (i.e., four NB operands) to produce one RB number (i.e., two NB numbers), it has four inputs and two outputs. Therefore, the RBA acts as a RB 4:2 compressor. The logic expressions of a redundant binary full adder (RBFA) are as follows [35]:

$$g_k = x_k^- \oplus x_k^+ \oplus y_k^- \oplus y_k^+ \tag{7}$$

$$h_k = x_k^- x_k^+ + y_k^- y_k^+ \tag{8}$$

$$C_k^- = (x_k^- + x_k^+)(y_k^- + y_k^+) \tag{9}$$

$$C_k^+ = g_k C_{k-1}^- + \overline{g_k} h_k \tag{10}$$

$$S_k^- = g_k \oplus C_{k-1}^- \tag{11}$$

$$S_k^+ = C_{k-1}^+ \tag{12}$$

Therefore, S_k^- and S_k^+can also be expressed as follows by combing the above equations:

$$S_k^- = x_k^- \oplus x_k^+ \oplus y_k^- \oplus y_k^+ \oplus ((x_{k-1}^- + x_{k-1}^+)(y_{k-1}^- + y_{k-1}^+)) \tag{13}$$

$$\begin{aligned} S_k^+ =& (x_{k-1}^- \oplus x_{k-1}^+ \oplus y_{k-1}^- \oplus y_{k-1}^+)((x_{k-2}^- + x_{k-2}^+) \\ & (y_{k-2}^- + y_{k-2}^+)) + (x_{k-1}^- \oplus x_{k-1}^+ \oplus y_{k-1}^- \oplus y_{k-1}^+) \\ & (x_{k-1}^- x_{k-1}^+ y_{k-1}^- y_{k-1}^+) \end{aligned} \tag{14}$$

The RBHA can be designed with $y_k^- = y_k^+ = 0$. The main advantage of RB multipliers that relay on RBAs, is the continuous carry-free characteristic. The RBA ensures that the addition time is fixed, so it is independent of the word length of the operands [36].

2.4 Review of RB-NB Converter

After the RB PP accumulation, two rows of NB numbers (i.e., one RB number) remain. They must be added by a RB-NB converter to form the final NB product. The RB-NB converter is a fast adder, which can be expressed as follows [46]:

$$S_k = \overline{C_k \oplus (S_k^- \oplus S_k^+)} \tag{15}$$

$$C_{k+1} = \overline{S_k^- + S_k^+} + \overline{S_k^- S_k^+} C_k \tag{16}$$

3 Design of Approximate RB Multipliers

Four approximate RB multipliers are designed in this section based on two approximate Booth encoders, two approximate RB 4:2 compressors, and an approximate RB-NB converter. Both exact and approximate regular PP arrays are used to meet the trade-off between accuracy and complexity.

3.1 The Proposed Approximate Booth Encoders

Two approximate Booth encoders are designed based on the conventional modified Booth encoding method and the new modified Booth encoding method, respectively.

3.1.1 Radix-4 Approximate MBE

The K-map of the radix-4 approximate modified Booth encoder (R4AMBE6), *i.e.*, app_{ij6-1}, with 6 errors in the K-map is shown in Table 4, where (0) denotes an entry in which a '1' is replaced by a '0' and (1) denotes a '0' entry that has been replaced by a '1'. Only 6 entries are modified to simplify the Booth encoding. This approximate design relies on the property that the truth table is as symmetrical as possible for a design with the least complexity. Therefore, three modifications change a '1' to a '0' and three modifications change a '0' to a '1' in the K-map. The output of R4AMBE6 is given as follows:

$$app_{ij6-1} = (b_{2i} + b_{2i-1})(b_{2i+1} \oplus a_i) \tag{17}$$

$$E_i = (b_{2i+1}\overline{b_{2i}}) + (b_{2i+1}\overline{b_{2i-1}}) \tag{18}$$

Compared with the exact MBE, R4AMBE6 can significantly reduce both the complexity and the critical path delay of Booth encoding. The error rate, denoted by P_{be}, is given by:

$$P_{be} = 6/32 = 18.75\% \tag{19}$$

The gate level structure of R4AMBE6 is shown in Fig. 5. The conventional design of MBE (Fig. 2) consists of four XNOR-2 gates, one XOR-2 gate, one OR-3 gate, one OR-2 gate and one NAND-2 gate. The R4AMBE6 design only requires one XOR-2 gate, one AND-2 gate and one OR-2 gate.

3.1.2 Radix-4 Approximate NMBE

The approximate Radix-4 with the new modified Booth Encoding (R4ANMBE6), *i.e.*, app'_{ij6-1}, with 6 errors in the K-map is shown in Table 5. In this approximate design, there are more entries changed from '0' to '1' than those changed from '1' to '0'. Therefore, the approximate results produced by R4ANMB6 will be usually larger than its exact

TABLE 4
K-Map of R4AMBE6

$a_j a_{j-1}$ \ $b_{2i+1}b_{2i}b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	(1)	1	(0)
01	0	0	(0)	0	1	(1)	1	0
11	0	1	1	1	0	0	0	0
10	0	1	(1)	1	0	0	0	(0)

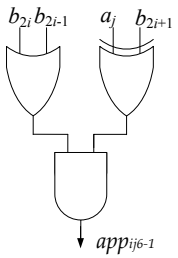


Fig. 5. The gate-level circuit of the proposed R4AMBE6.

TABLE 5
K-Map of R4ANMBE6

$a_j a_{j-1}$ \ $b_{2i+1}b_{2i}b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	(1)	0	1	1	1	1
01	0	0	1	0	1	1	1	(1)
11	(1)	1	1	1	0	1	0	0
10	(1)	1	(1)	1	0	1	0	(0)

counterpart. From Table 5, the approximate pp'_{ij} is derived as follows:

$$app'_{ij6-1} = b_{2i+1} \oplus a_j + b_{2i}b_{2i-1} \tag{20}$$

$$E'_i = b_{2i+1} \tag{21}$$

This design further reduces the complexity of the correction term (*i.e.*, E_i). Its error rate is the same as R4AMBE6:

$$P'_{be} = 6/32 = 18.75\% \tag{22}$$

The gate level circuit of R4ANMBE6 is shown in Fig. 6. The R4AMBE6 design only requires one XOR-2 gate, one AND-2 gate and one OR-2 gate, which has the same complexity as R4AMBE6.

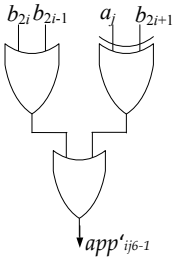


Fig. 6. The gate-level circuit of the proposed R4AMBE6.

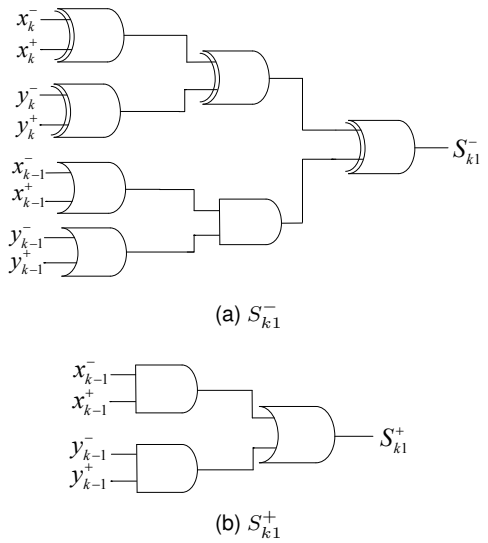


Fig. 7. The gate level circuit of ARBC-1.

3.2 The Proposed Approximate RB 4:2 Compressors

As per Eqs. (13-14), S_k^- and S_k^+ are determined by the following 12 variables: x_k^-, x_k^+, y_k^-, y_k^+, x_{k-1}^-, x_{k-1}^+, y_{k-1}^-, y_{k-1}^+, x_{k-2}^-, x_{k-2}^+, y_{k-2}^-, y_{k-2}^+. Therefore, the number of all possible outputs is 4096 (*i.e.*, 2^{12}). An efficient designs must ensure that the error between the approximate RB compressor and its exact counterpart remains as small as possible.

The final results of compression are the same when (x_k^-,x_k^+) is equal to either (1, 0) or (0, 1). So, when the result of the approximate RB compressor is (x_k^-,x_k^+) = (1, 0) rather than the exact compression result (x_k^-,x_k^+) = (0, 1), the result is still correct. Therefore, the following four types of compression results are equivalent: (0, 0) = (0, 0), (0, 1) = (1, 0), (1, 0) = (0, 1) and (1, 1) = (1, 1).

3.2.1 Approximate RB 4:2 Compressor 1

S_k^+ can be simplified by ignoring the asymmetric part of the exact RB compressor (ERBC) in Eq. (14). The first approximate RB compressor (ARBC-1) is given by the following expressions:

$$S_{k1}^- = x_k^- \oplus x_k^+ \oplus y_k^- \oplus y_k^+ \oplus ((x_{k-1}^- + x_{k-1}^+)(y_{k-1}^- + y_{k-1}^+)) \quad (23)$$

$$S_{k1}^+ = x_{k-1}^- x_{k-1}^+ + y_{k-1}^- y_{k-1}^+ \quad (24)$$

The error rate of this proposed approximate RB compressor is:

$$P_{ce} = 1024/4096 = 25\% \quad (25)$$

The gate level circuit of the approximate RB compressor is given in Fig. 7. The approximate S_{k1}^+ has only 3 gates, while the exact S_{k1}^+ requires 12 gates. In total, ARBC-1 reduces the gate count from 19 to 10.

3.2.2 Approximate RB 4:2 Compressor 2

S_k^- and S_k^+ can be further simplified as an approximate RB compressor. The second approximate RB 4:2 compressor (ARBC-2) is given by:

$$S_{k2}^- = x_k^- \oplus x_k^+ \oplus y_k^- \oplus y_k^+ \oplus (y_{k-1}^- + y_{k-1}^+) \quad (26)$$

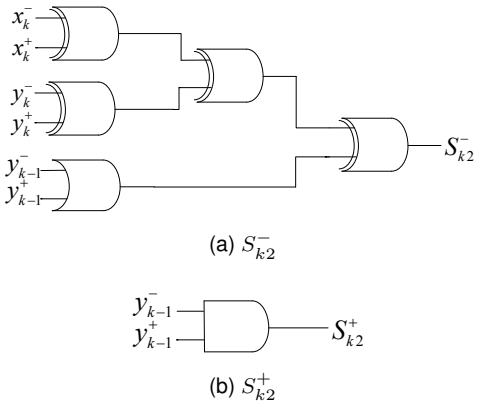


Fig. 8. The gate level circuit of ARBC-2.

$$S_{k2}^+ = y_{k-1}^- y_{k-1}^+ \quad (27)$$

The error rate of the proposed ARBC-2 is as follows:

$$P_{ce}^{'} = 1296/4096 = 31.6\% \quad (28)$$

Also ARBC-2 generates results that are larger than its exact counterpart. The gate level design of the approximate RB compressor is given in Fig. 8. ARBC-2 further reduces the gate count of S_{k2}^- from 7 to 5. Therefore, ARBC-2 reduces the gate count from 19 to 6, which is significantly simpler than ERBC.

3.3 The Proposed Approximate RB-NB Converter

As the approximate Booth encoders and approximate RB compressors generate results that are generally larger than the exact results, the biased approximate results can be compensated using ARNC with smaller values. The principle of compensation is to use an approximate adder that produces results that are smaller than its exact results. Therefore, the complexity of the RB-NB converter can be reduced, while the overall accuracy of the approximate RB multipliers is also increased. The truth table of a possible approximate RB-NB converter is given by Table 6, a simple NOR gate is used in the approximate RB-NB digit converter as follows:

$$S_k^{'} = \overline{S_k^- + S_k^+} \quad (29)$$

The approximate RB multipliers using ARNC can reduce the error of an entire multiplier, as further analyzed in Section 4.3.

3.4 Design of Approximate RB Multipliers

In this section, the approximate RB multipliers are designed as follows. The proposed approximate Booth encoders, *i.e.*, R4AMBE6 and R4ANMBE6, are used to generate approximate PPs. Approximate RB compressors, *i.e.*, ARBC-1 and ARBC-2, are used for RB PP reduction, which can reduce the delay for compression and significantly improve speed performance when the operand size is a power of 2. The approximate RB-NB converter (made of NOR gates) is used to convert the RB digit to the NB digit.

An approximation factor p (p=1, 2, ..., 2N) that has been proposed in [21] is used. This is defined as the number of least significant PP columns that are generated by the

TABLE 6
The Truth Table of RB-NB Conversion (ENB: Exact Normal Binary Digit, ANBI: Approximate Normal Binary Digit from ARNCI)

RB Digit (S_k^+,S_k^-)	Carry-in(C_k)	ENB (S_k)	ANB1 (S'_{k1})	ANB2 (S'_{k2})	ANB3 (S'_{k3})
(0,0)	0	1	1	1	1
(0,0)	1	0	1	1	1
(0,1)	0	0	0	1	0
(0,1)	1	1	0	1	0
(1,0)	0	0	0	1	0
(1,0)	1	1	0	1	0
(1,1)	0	1	0	0	1
(1,1)	1	0	0	0	1

approximate Booth encoders. As p column PPs are already approximate, the approximate PPs can be accumulated with an approximate RB 4:2 compressor to further improve speed and reduce power consumption. For the same reason, the p least significant RB digits are also converted by the approximate RB-NB converter to calculate the final product.

Four approximate RB multipliers are proposed. They use the exact regular PP array when $p \leq (N-4)$ (as detailed in [36]), and the approximate regular PP array when $p > (N-4)$ where the bit pairs (E_2, 0) and (E_3, 1) of Fig. 4 can be ignored in the approximate design of the RB Booth multipliers; however they all use the proposed approximate RB-NB converter. For the 2N-p most significant PP columns, the exact design is used for the final results.

The four RB multipliers are different in the p PP columns as follows:

1) The first approximate RB multiplier (R4ARBM1) uses R4AMBE6 to generate the p least significant PP columns and ARBC-1 to perform the approximate PP accumulation.
2) The second approximate RB multiplier (R4ARBM2) uses R4AMBE6 to generate the p least significant PP columns and ARBC-2 for the corresponding approximate PP accumulation.
3) The third approximate RB multiplier (R4ARBM3) uses R4ANMBE6 to generate the p least signifi-cant PP columns and ARBC-1 to perform the approximate PP accumulation.
4) The fourth approximate RB multiplier (R4ARBM4) uses R4ANMBE6 to generate the p least significant PP columns and ARBC-2 to perform the approximate PP accumulation.

As the error can be controlled by the approximation factor p, a reasonable accuracy can be achieved for different applications. Fig. 9 shows an approximate 8-bit RB multiplier with p=4 using an approximate Booth encoder, an approximate RB compressor, an approximate RB-NB converter, and an exact regular PP. A box with a solid line denotes the use of an exact RB compressor, and a box with a dotted line denotes an approximate RB 4:2 compressor. The exact PP is represented by ● , the modified PP after logic simplification is represented by ▼ , while the approximate PP term is represented by ■. ◎ represents E_i.

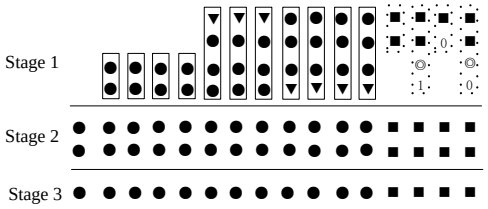


Fig. 9. The dot diagram of the proposed 8-bit approximate RB multiplier (R4ARBM) with p=4.

TABLE 7
Comparison between Exact and Approximate Booth Encoders to Generate a 1-Bit PP

Booth Encoder	Power (μW)	Delay (ps)	Area (μm^2)	Energy (aJ)	Error Rate
MBE	1.99	80	9.84	159.20	0%
R4AMBE4 [21]	0.93	70	3.99	65.10	12.5%
R4AMBE6	0.69	70	3.19	48.30	18.75%
R4ANMBE6	0.68	70	3.19	47.60	18.75%
R4AMBE8 [21]	0.23	50	1.60	11.50	25.0%

4 Error Analysis and Hardware Evaluation

4.1 Booth Encoder Evaluation

The two proposed approximate Booth encoders are compared with the exact MBE and two approximate Booth encoders. They have been proposed in [21] with 4 errors (*i.e.*, R4AMBE4) and 8 errors (*i.e.*, R4AMBE8) in the K-map of MBE.

All designs in this work are described at gate-level in Verilog HDL with a non-pipelined version and verified by Synopsys VCS. Both designs are then synthesized by the Synopsys Design Compiler using the NanGate 45 nm Open Cell Library, in which the FO4 delay is given by 22ps and the area of an inverter (*i.e.* INVX1) is 1.41 μm^2. 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 drivers and the output loads. The average power consumption is found using the Synopsys Power Compiler with a back annotated switching activity file generated from the random input vectors.

Table 7 summarizes the power, delay, area, energy and error rates of the exact and approximate Booth encoders. As expected, the approximate Booth encoders significantly reduce the energy compared with the exact MBE. R4AMBE8 has the smallest energy and the largest error, while R4AMBE4 is the opposite. The proposed R4AMBE6 and R4ANMBE6 have moderate error and energy, so achieving a better tradeoff between error and performance.

4.2 RB 4:2 Compressor Evaluation

The proposed approximate RB 4:2 compressors, *i.e.*, ARBC-1 and ARBC-2 are compared with exact RB converter (ERBC) in Table 8. ARBC-1 and ARBC-2 reduce the energy by over 47% and 64%, respectively, compared with ERBC.

TABLE 8
Comparison between Exact and Approximate RB Compressors

RB compressors	Power (μW)	Delay (ps)	Area (μm^2)	Energy (aJ)	Error Rate
ERBC	4.54	180	17.56	816.6	0.0%
ARBC-1	3.07	140	12.77	429.8	25.0%
ARBC-2	2.06	140	9.31	288.4	31.6%

TABLE 9
Multipliers and Their Abbreviations

Abbreviation	Designs
R4ERBM	Radix-4 exact RB multiplier
R4AMBE6	Radix-4 approximate modified Booth encoder with 6 errors in the K-Map
R4ANMBE6	Approximate radix-4 the new modified Booth encoder with 6 errors in the K-Map
ARBC-1	The first approximate RB compressor
ARBC-2	The second approximate RB compressor
R4TBM	Naive radix-4 truncated Booth multiplier without error compensation
R4ARBM1	Design 1 using R4AMBE6 to generate the p least significant PP columns and ARBC-1 to perform the approximate PP accumulation
R4ARBM2	Design 2 using R4AMBE6 to generate the p least significant PP columns and ARBC-2 to perform the approximate PP accumulation
R4ARBM3	Design 3 using R4ANMBE6 to generate the p least significant PP columns and ARBC-1 to perform the approximate PP accumulation
R4ARBM4	Design 4 using R4ANMBE6 to generate the p least significant PP columns and ARBC-2 to perform the approximate PP accumulation
R4ARBM04 [22]	Radix-4 approximate Booth multiplier with PP truncation
R4ABM11 [25]	Radix-4 approximate Booth multiplier with PP truncation and error compensation
R4ABM12 [26]	Radix-4 approximate Booth multiplier with an adaptive conditional-probability estimator
R8ABM2-C15 [19]	Radix-8 approximate Booth multiplier with approximate recoding adder and error com-pensation with 15 bits truncation
R8ABM2-C30 [19]	Radix-8 approximate Booth multiplier with approximate recoding adder and error com-pensation with 30 bits truncation
RAD256 [20]	Hybrid high radix approximate 16-bit multiplier
RAD2^{24} [20]	Hybrid high radix approximate 32-bit multiplier

4.3 Error Analysis of Approximate RB Multipliers

Although the error rate of each approximate circuit (or module) has been presented, the error characteristics of the entire approximate Booth multiplier must be also considered. For approximate designs, several metrics have been proposed to measure the error of approximate adders and multipliers including the mean error distance , the relative error distance and the normalization of MED (NMED) [42]. Three main error metrics, *i.e.*, NMED, MAE and MRED, are used to compare different approximate designs of various sizes:

- The NMED is defined as the normalized MED by the maximum output of the accurate design.
- MAE is defined as the maximum absolute error.
- The MRED is defined as the mean relative error distance, and the relative error distance (RED) is defined as the ED over the absolute accurate result.

The acronyms of the different approximate multipliers are shown in Table 9. The error metrics of the proposed approximate multipliers are shown in Table 10 for the NMED, MAE and MRED of 16-bit and 32-bit designs, respectively.

TABLE 10
Errors of Proposed 16-Bit and 32-Bit Approximate Multipliers at Different Approximation Factors

Designs	16-bit p	16-bit NMED (10^{-5})	16-bit MAE (10^{5})	16-bit MRED (10^{-2})	32-bit p	32-bit NMED (10^{-10})	32-bit MAE (10^{10})	32-bit MRED (10^{-4})
R4ARBM1	4	2.79×10^{-4}	2.8×10^{-4}	7.15×10^{-5}	8	1.93×10^{-7}	1.02×10^{-7}	7.01×10^{-8}
	8	8.43×10^{-3}	1.02×10^{-2}	2.49×10^{-2}	16	7.92×10^{-5}	4.82×10^{-5}	2.52×10^{-5}
	12	1.68×10^{-1}	2.50×10^{-1}	3.37×10^{-1}	24	2.54×10^{-2}	1.43×10^{-2}	1.16×10^{-2}
	14	1.16	1.08	1.39	28	4.62×10^{-1}	2.53×10^{-1}	1.42×10^{-1}
	16	3.62	4.56	9.94	32	6.78	3.99	1.83
	18	13.3	14.89	12.4	36	103	6.88×10^{1}	23.2
	20	50.8	53.17	52	44	21400	1.54×10^{4}	5350
	24	703	594.85	1021	52	4.16×10^{6}	2.13×10^{6}	5.79×10^{5}
	28	4420	5949.93	1133	60	1.05×10^{9}	1.62×10^{8}	6.92×10^{8}
R4ARBM2	4	2.79×10^{-4}	2.5×10^{-4}	7.15×10^{-5}	8	1.79×10^{-7}	8.93×10^{-8}	5.86×10^{-8}
	8	7.78×10^{-3}	8.93×10^{-3}	2.46×10^{-2}	16	6.98×10^{-5}	4.06×10^{-5}	2.76×10^{-5}
	12	1.50×10^{-1}	1.94×10^{-1}	3.27×10^{-1}	24	2.21×10^{-2}	1.26×10^{-2}	6.48×10^{-2}
	14	7.60×10^{-1}	9.89×10^{-1}	2.61	28	3.83×10^{-1}	2.17×10^{-1}	9.82×10^{-2}
	16	3.21	3.79	5.44	32	5.5	3.46	2.59
	18	10.4	13.07	44.1	36	91	6.62×10^{1}	44.7
	20	37.8	44.41	178	44	2.15×10^{4}	1.38×10^{4}	103
	24	536	404.07	2948	52	4.38×10^{6}	1.96×10^{6}	2.12×10^{6}
	28	7540	3423.97	49090	60	1.21×10^{9}	1.15×10^{8}	6.79×10^{8}
R4ARBM3	4	4.66×10^{-4}	3.9×10^{-4}	2.20×10^{-4}	8	3.30×10^{-7}	1.14×10^{-7}	1.16×10^{-7}
	8	1.23×10^{-2}	1.14×10^{-2}	2.41×10^{-2}	16	1.44×10^{-4}	5.55×10^{-5}	7.37×10^{-5}
	12	2.65×10^{-1}	2.64×10^{-1}	1.16	24	5.44×10^{-25}	1.98×10^{-2}	2.69×10^{-2}
	14	7.09×10^{-1}	9.00×10^{-1}	3.95	28	8.86×10^{-1}	3.40×10^{-1}	4.46×10^{-1}
	16	2.93	3.45	20.5	32	18.2	5.53	9.96
	18	9.8	11.85	83.8	36	2.92×10^{2}	8.88×10^{1}	1.22×10^{2}
	20	36.8	44.51	34.25	44	5.65×10^{4}	1.93×10^{4}	1.81×10^{4}
	24	489	553.45	4199	52	9.58×10^{6}	3.29×10^{6}	1.24×10^{6}
	28	6370	4051.39	49180	60	0.11×10^{9}	1.73×10^{8}	6.97×10^{8}
R4ARBM4	4	3.26×10^{-4}	2.4×10^{-4}	1.01×10^{-4}	8	2.41×10^{-7}	8.93×10^{-8}	7.58×10^{-8}
	8	9.27×10^{-3}	8.93×10^{-3}	1.58×10^{-2}	16	9.75×10^{-5}	4.88×10^{-5}	4.31×10^{-5}
	12	1.71×10^{-1}	1.84×10^{-1}	8.53×10^{-1}	24	3.52×10^{-2}	1.72×10^{-2}	1.23×10^{-2}
	14	6.99×10^{-1}	7.82×10^{-1}	4.95	28	5.43×10^{-1}	2.92×10^{-1}	2.68×10^{-1}
	16	3.18	3.5	22.17	32	13	5.09	6.5
	18	10.9	11.89	92.12	36	2.22×10^{2}	9.06×10^{1}	96.0
	20	40.3	38.88	375.24	44	4.91×10^{4}	1.81×10^{4}	2.35×10^{4}
	24	525	458.66	5074.79	52	9.70×10^{6}	3.28×10^{6}	3.00×10^{6}
	28	7400	4662.46	50513.3	60	1.10×10^{9}	1.53×10^{8}	6.76×10^{8}

All of the four approximate RB multipliers have similar NMED and MRED. As p increases, more errors are introduced due to the increasing number of approximate Booth encoders and compressors used in the design. The errors of the approximate RB multipliers increase logarithmically with a linear increase of p for both 16-bit and 32-bit designs. These results confirm that the best designs are very close. Their hardware features are studied in the next section.

The NMEDs of the approximate RB multipliers using both exact RN converter and approximate RN converter are shown in Fig. 10. As the results of R4ARBM3 and R4ARBM4 are close to that R4ARBM2, for clarity only the NMEDs of R4ARBM1 and R4ARBM2 are shown. All approximate RB multipliers using the proposed ARNC have smaller NMEDs than using exact RB-NB converter (ERNC). Thus, these results confirm that the proposed ARNC can compensate errors from the approximate Booth encoder and the ARBCs.

4.4 Hardware Evaluation of Approximate RB Multipliers

Hardware evaluation by simulation is pursued for the proposed approximate multipliers under the same conditions as in Section 4.1. The power, area and delay of 16-bit and 32-bit exact and approximate RB multipliers are analyzed in Table 11.

The area of the approximate RB multipliers decreases almost linearly with an increase of p for both 16-bit and 32-bit designs. This occurs because when more approximate units (including Booth encoders, compressors and the RB-NB converter) are used, more area is saved due to the simplified logic. As ARBC1 is more complex than ARBC2,

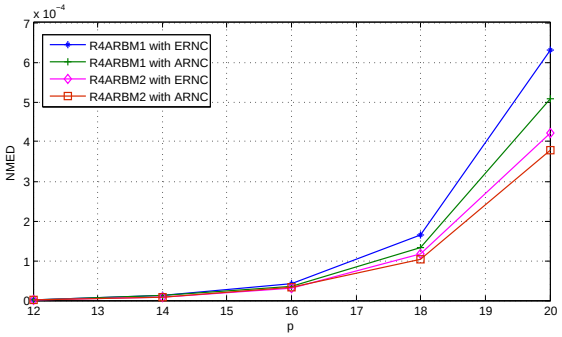


Fig. 10. NMED comparison for 16-bit approximate RB multipliers with both exact and approximate RB-NB converters.

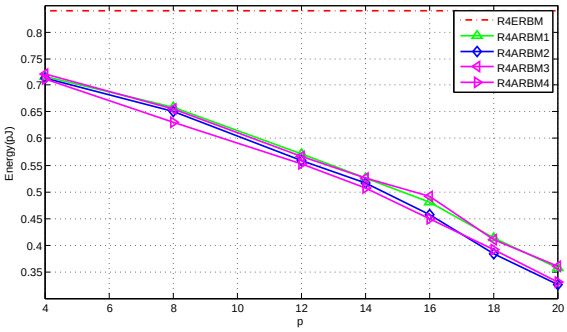


Fig. 11. Energy comparison of four 16-bit approximate RB multipliers.

the areas of R4ARBM1 and R4ARBM3 are slightly larger at a larger value of p (due to ARBC1).

The delay and power also decrease as expected; these two metrics are considered by using the energy. As shown in Fig. 11, the energy of the four designs decrease with an increase of p. Both R4ARBM2 and R4ARBM4 have lower energy, while R4ARBM4 is the best. The energy of R4ARBM1 is slightly larger than for other three designs. The delay of the proposed approximate multipliers is shown in Table 12; as expected, the delay of the compressor is the longest.

4.5 Comparison with Approximate Booth Multipliers

The proposed designs at large values of p have also large errors, but small energy; moreover, the proposed 16-bit designs with p<18 show a good trade-off between energy and NMED. Both proposed 16-bit designs (with p=12, 14 and 16) are compared with previous approximate 16-bit Booth multipliers that have been proposed in [22] (R4ABM04), [25] (R4ABM11), [26] (R4ABM12), [19] (R8ABM2-C15), [20] (RAD256) and [21] (R4ABM1 and R4ABM2). Only signed multipliers are compared for fair comparison. R4ABM04, R4ABM11 and R4ABM12 are radix-4 approximate Booth multipliers with truncation, respectively. R8ABM2-C15 is a radix-8 approximate Booth multiplier with 15-bit truncation and compensation circuits. RAD256 is a radix-256 approximate Booth multiplier. The designs of R4ABM1 and R4ABM2 are also compared at p=12, 14 and 16.

In these cases, all designs are also described in Verilog as combinational multipliers and synthesized by the Synopsys Design Compiler using the NanGate 45 nm Open Cell Library. The power consumption, critical path delay, area, energy, NMED and the NMED and energy product (NEP) are reported in Table 13; as a combined metric, the NEP is used to measure the overall performance of the approximate designs.

TABLE 11
Designs (45nm Technology) of the Proposed 16-Bit and 32-Bit ARBMs at Different Approximation Factors

Designs	16-bit				32-bit			
	p	Power (μW)	Area (μm^2)	Delay (ps)	p	Power (μW)	Area (μm^2)	Delay (ps)
R4ERBM	0	800.4	3206.3	1050	0	3.18×10^3	12342.7	1370
R4ARBM1	4	724.3	2868.3	990	8	2.73×10^3	10519.5	1230
	8	672.8	2689.8	980	16	2.66×10^3	9945.2	1200
	12	607.3	2429.6	940	24	2.36×10^3	8920.6	1230
	14	590.9	2348.5	890	28	2.27×10^3	8565.7	1180
	16	551.9	2223	870	32	2.13×10^3	8145.5	1130
	18	486.2	1947.4	850	36	1.94×10^3	7331.8	1110
	20	464.2	1875.8	770	44	1.63×10^3	6361.7	1110
	24	373.9	1537.7	690	52	1.41×10^3	5596	1000
	28	322.1	1338	610	60	1.27×10^3	5092.3	840
R4ARBM2	4	728.1	2890.1	980	8	2.71×10^3	10448.5	1240
	8	677.9	2689	960	16	2.60×10^3	9813	1240
	12	588.4	2357.6	950	24	2.27×10^3	8625	1240
	14	580.1	2294.2	890	28	2.17×10^3	8252.4	1200
	16	518.9	2101.1	880	32	2.03×10^3	7772.3	1140
	18	466.9	1864.1	820	36	1.80×10^3	6875.8	1120
	20	418.5	1723.7	780	44	1.44×10^3	5775.1	1090
	24	330.7	1389.3	680	52	1.18×10^3	4914.3	1020
	28	271.1	1172.3	590	60	1.01×10^3	4335.3	810
R4ARBM3	4	743.1	2928.1	970	8	2.72×10^3	10448.2	1230
	8	676.4	2679.4	970	16	2.63×10^3	9883	1220
	12	602	2404.9	940	24	2.31×10^3	8823	1240
	14	592	2346.4	890	28	2.23×10^3	8450.6	1190
	16	559.4	2214.4	880	32	2.15×10^3	8147.6	1130
	18	494	1956.2	830	36	1.93×10^3	7290	1100
	20	461.6	1837.3	780	44	1.61×10^3	6272.8	1090
	24	375	1525	690	52	1.43×10^3	5551.2	990
	28	332.5	1337.2	620	60	1.28×10^3	5053.7	840
R4ARBM4	4	733.8	2901	970	8	2.72×10^3	10403.3	1250
	8	670	2686.3	940	16	2.60×10^3	9794.1	1230
	12	587.7	2362.1	940	24	2.27×10^3	8687.8	1220
	14	570.2	2277.8	890	28	2.15×10^3	8187.2	1200
	16	511.4	2091.8	880	32	1.99×10^3	7683.4	1120
	18	467	1876.9	840	36	1.82×10^3	6922.9	1090
	20	417.8	1724.2	790	44	1.43×10^3	5748.3	1110
	24	323.5	1366.4	680	52	1.18×10^3	4883	990
	28	281	1179.4	590	60	1.03×10^3	4328.6	780

As shown in Table 13, among all approximate Booth multipliers, the proposed four approximate RB multipliers with p=12 are all better than previous approximate Booth multipliers by considering both error and energy. The proposed R4ARBMs are more accurate than all other designs, among which R4ARBM2 with p=12 is the most accurate design. The proposed designs R4ARBM3 and R4ARBM4 with p=16 also have the smallest delay. Although R8ABM2-C15 has the smallest power, area and energy, it introduces a significantly larger error and incurs in a large delay. The NEPs for all approximate Booth multipliers are shown in Fig. 12. When considering both energy and NMED, R4ABMs and R4ARBMs with p<16 have smaller NEPs than the truncated multipliers (R4ABM04, R4ABM11 and R4ABM12) and the high radix multipliers (R8ABM2-C15 and RAD256). When considering the MRED (NMED) RAD256 has better (worse) performance than R4ABM. Among high radix multipliers, the radix-8 multiplier has a smaller NEP than the radix-256

TABLE 12
Timing Analysis (45nm Technology) of Proposed 16-Bit and 32-Bit ARBMs at Different Approximation Factors (unit: *ps*)

Designs	16-bit				32-bit			
	p	MBE	Com-pressors	RB-NB Converter	p	MBE	Com-pressors	RB-NB Converter
R4ERBM	0	210	410	430	0	310	490	440
R4ARBM1	4	200	420	370	8	350	450	430
	8	200	410	370	16	320	470	410
	12	190	430	320	24	360	450	420
	14	190	430	270	28	300	500	380
	16	190	360	320	32	240	500	390
	18	200	340	310	36	220	510	380
	20	180	360	230	44	230	510	370
	24	160	340	190	52	220	500	280
	28	160	290	160	60	150	350	340
R4ARBM2	4	210	410	360	8	350	460	430
	8	210	380	400	16	300	510	430
	12	200	410	340	24	350	480	410
	14	200	350	340	28	290	490	420
	16	200	340	340	32	220	540	380
	18	170	370	280	36	240	500	380
	20	190	340	250	44	230	510	350
	24	160	280	240	52	200	550	270
	28	170	250	170	60	150	330	330
R4ARBM3	4	260	340	370	8	380	430	420
	8	210	400	360	16	290	510	420
	12	200	400	330	24	360	470	410
	14	200	350	340	28	300	490	400
	16	190	350	340	32	240	500	390
	18	190	320	300	36	230	510	360
	20	170	350	260	44	210	540	340
	24	120	330	240	52	210	510	270
	28	160	230	230	60	180	330	320
R4ARBM4	4	190	420	360	8	370	450	430
	8	190	410	340	16	290	500	440
	12	210	390	340	24	320	480	400
	14	200	350	340	28	300	490	410
	16	190	350	340	32	230	510	360
	18	190	330	320	36	230	490	370
	20	180	360	250	44	230	510	370
	24	160	290	230	52	200	520	270
	28	140	230	220	60	150	310	320

TABLE 13
Comparison of 16-Bit Approximate Booth Multipliers

Approximate Booth Multipliers		Power (μW)	Delay (ps)	Area (μm^2)	Energy (pJ)	NMED (10^{-5})	NEP ($pJ \cdot 10^{-5}$)
R4ABM04 [22]		427.3	950	1939	0.406	5.31	2.156
R4ABM11 [25]		404.4	940	1859	0.380	2.18	0.828
R4ABM12 [26]		394.6	950	1808	0.374	2.26	0.845
R8ABM2-C15 [19]		**217.3**	1180	**912**	**0.256**	5.73	1.467
RAD256 [20]		554.6	940	2393	0.523	22.7	11.872
R4ABM1 [21]	(p=12)	535	950	2209	0.508	0.31	0.157
	(p=14)	516.6	950	2169	0.49	0.93	0.456
	(p=16)	479.9	940	2066	0.451	3.1	1.398
R4ABM2 [21]	(p=12)	515.4	940	2077	0.484	0.27	0.131
	(p=14)	479.7	920	2004	0.441	0.62	0.273
	(p=16)	448.7	920	1875	0.412	3.02	1.244
R4ARBM1	(p=12)	607.3	940	2430	0.571	0.17	0.097
	(p=14)	590.9	890	2349	0.526	1.16	0.610
	(p=16)	551.9	870	2223	0.48	3.62	1.738
R4ARBM2	(p=12)	588.4	950	2358	0.559	**0.15**	0.084
	(p=14)	580.1	890	2294	0.516	0.76	0.392
	(p=16)	518.9	880	2101	0.457	3.21	1.467
R4ARBM3	(p=12)	602	940	2405	0.566	0.27	0.153
	(p=14)	592	890	2346	0.527	0.71	0.374
	(p=16)	559.4	**880**	2214	0.492	2.93	1.442
R4ARBM4	(p=12)	587.7	940	2362	0.552	0.17	0.094
	(p=14)	570.2	890	2278	0.507	0.7	0.355
	(p=16)	511.4	**880**	2092	0.45	3.18	1.431

multiplier.

The previously designs of [20], [21], [22], [25], [26], and the naive truncated multipliers without error compensation (R4TBM) are further compared with the four proposed approximate RB multipliers (with p=24, 28, 32) for 32-bit designs. The power, delay, area, energy, NMED and NEP of these 32-bit approximate multipliers are compared in Table 14.

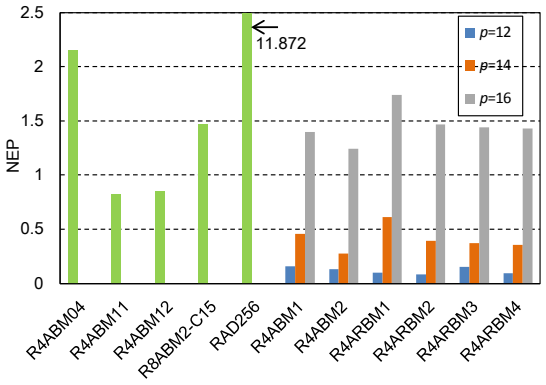


Fig. 12. The comparison of NEP for 16-bit approximate Booth multipliers. Note that the NEP of RAD256 is 11.872.

TABLE 14
Comparison of 32-Bit Approximate Booth Multipliers

Approximate Booth Multipliers		Power (mW)	Delay (ps)	Area (μm^2)	Energy (pJ)	NMED (10^{-11})	NEP ($pJ \cdot 10^{-11}$)
R4TBM	(p=24)	1.79	1440	7346.4	2.582	10.22	26.398
	(p=28)	1.55	1390	6548.9	2.157	16.612	35.838
	(p=32)	1.28	1310	5560.2	**1.670**	121.37	202.717
R4ABM04 [22]		1.45	1360	6146.2	1.972	23.89	47.111
R4ABM11 [25]		1.41	1380	5952.8	1.946	18.86	36.702
R4ABM12 [26]		1.37	1350	5881.3	1.850	12.73	23.551
R8ABM2-C30 [19]		**0.72**	2580	**2979.7**	1.865	8.66×10^{3}	1.615×10^{4}
RAD2^{24} [20]		1.19	1410	4947.9	1.678	4.03×10^{7}	6.762×10^{7}
R4ABM1 [21]	(p=24)	2.24	1520	8901.1	3.407	0.26	0.886
	(p=28)	2.13	1510	8557.4	3.221	5.56	17.909
	(p=32)	2.05	1510	8200.5	3.095	81.16	251.190
R4ABM2 [21]	(p=24)	2.09	1480	8540.1	3.087	0.4	1.235
	(p=28)	2.03	1480	8103.9	3.01	4.04	12.160
	(p=32)	**1.87**	1460	7684.5	2.731	112.26	306.582
R4ARBM1	(p=24)	2.36	1230	8920.6	2.903	0.25	0.726
	(p=28)	2.27	1180	8565.7	2.679	4.62	12.377
	(p=32)	2.13	1130	8145.5	2.407	67.8	163.195
R4ARBM2	(p=24)	2.27	1240	8625	2.815	**0.22**	0.619
	(p=28)	2.17	1200	8252.4	2.604	3.83	9.973
	(p=32)	2.03	1140	7772.3	2.314	55	127.27
R4ARBM3	(p=24)	2.31	1240	8823	2.864	0.54	1.547
	(p=28)	2.23	1190	8450.6	2.654	8.86	23.514
	(p=32)	2.15	1130	8147.6	2.43	182	442.26
R4ARBM4	(p=24)	2.27	1220	8687.8	2.769	0.35	0.969
	(p=28)	2.15	1200	8187.2	2.58	5.43	14.009
	(p=32)	1.99	**1120**	**7683.4**	**2.229**	130	289.77

Table 14 shows that RAD2^{24} and R8ABM2-C30 have the lowest energy and power, respectively. However they all have large NMEDs. The truncated multipliers (R4ABM04, R4ABM11 and R4ABM12) all have smaller energy but little higher NMEDs than the proposed R4ARBMs when p<32. The naive truncated multipliers (R4TBM) generally have lower energy dissipation but higher NMEDs compared with R4ABMs and R4ARBMs with the same corresponding p. The proposed R4ARBM2 with p=24 has the smallest NMED. As shown in Fig. 13, the high radix multipliers (R8ABM2-C30 and RAD2^{24}) have significantly higher values of NEP. The truncated multipliers also have larger NEPs than R4ARBMs and R4ABMs when p<32. The proposed R4ARBM1 and R4ARBM2 are better than the two R4ARMs for large input sizes. For 32-bit designs, the best design is R4ARBM2 (p=24), while R4ARBM1 (p=24) is close to it. When considering the NEP metric or when targeting very small error values, the proposed multipliers are very good designs.

In general, for a high dynamic range computation, a large size approximate arithmetic circuit is required. The proposed 32-bit approximate RB multipliers can be applied to many error-tolerant applications beyond machine learning. For example, for high dynamic range (HDR) image processing, the data range is up to 10^8. The application of the proposed 32-bit R4ARBMs into HDR image processing

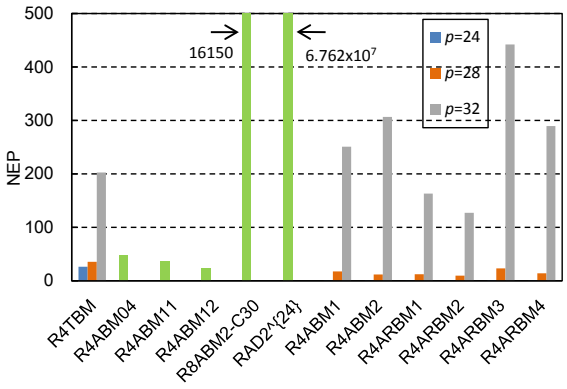


Fig. 13. The comparison of NEP for 32-bit approximate Booth multipliers. Note that the NEP of R8ABM2-C30 and RAD2^{24} is 16150 and 6.762$\times 10^7$, respectively.

TABLE 15
Signal-to-Noise Ratio (SNR_{out}) of the Filter Output Signal Processed by R4ARBM2 with Different p

p	0	4	8	12	14	16	18
SNR_{out}/dB	34.0800	35.5517	34.2425	34.0882	33.0489	23.9773	10.8200

is further illustrated in next section.

5 Application Case Studies

The proposed approximate RB multipliers are applied to FIR filtering, k-mean clustering and HDR image processing in this section. As R4ARBM2 shows the best accuracy, it is firstly used in this section for all four applications and then the results from all R4ABMs and R4ARBMs are compared.

5.1 FIR Filter

R4ARBM2 is applied to a 73-tap low-pass finite impulse response (FIR) filter using a Kaiser Window to further validate the proposed designs. The Filter Design & Analysis Tool in Matlab is used to design the FIR filter. The pass-band and stop-band frequencies of the filter are set to 8 *kHz* and 15 *kHz*, respectively, while the sample frequency is 100 *kHz*. The input signal is given by $s = s_1(n)+s_2(n)+s_3(n)+wgn(n)$, where s_1, s_2 and s_3 are sinusoidal signals with 1 *kHz*, 15 *kHz* and 20 *kHz* frequencies, respectively, and wgn is a white Gaussian noise with -30*dBW* power.

The input signal-to-noise ratio (SNR_{in}) and output signal-to-noise ratios (SNR_{out}) are used to assess the quality of the FIR filter that is designed using approximate Booth multipliers. For all cases, the SNR_{in} of -3.0257*dB* is used for comparison. The SNR_{out} of the FIR filter output signal processed by using R4ARBM2 is provided in Table 15. R4ARBM2 with $p\leq$14 produces good results for this application. This is consistent with the error analysis of Section 4.

The proposed ARBMs are further compared with R4ABMs for p=14. Table 16 shows the power, delay, energy and SNR_{out} when using the corresponding multipliers in the FIR application. The power is measured with the benchmark data. The proposed R4ARBMs show significantly better results than R4ABMs. The result with SNR_{out}=33.05*dB* for R4ARBM2 is the best. However, R4ABM2 with p=14 has lower energy than R4ARBMs.

TABLE 16
SNR_{out} of the Filter Output Signal Processed by Different Approximate 16-Bit Multipliers with p=14

p=14	R4ERBM	R4ABM1 [21]	R4ABM2 [21]	R4ARBM1	R4ARBM2	R4ARBM3	R4ARBM4
Power (μW)	680.2	410.6	390.7	427.5	418.8	429.8	410.0
Delay (ns)	1.03	0.95	0.92	0.89	0.89	0.89	0.89
Energy (%)	100	55.7	51.3	54.3	53.2	54.6	52.1
SNR_{out} (dB)	34.08	24.18	21.86	32.30	33.05	31.75	31.30

TABLE 17
F-measure Values of Clustering by K-mean Using R4ARBM2 with Different Approximate Factors. (NS: No. of Samples, NC: No. of Clusters)

Datasets	NS	NC	F-measure values						
			p=0	p=12	p=16	p=20	p=24	p=28	p=32
Iris	150	3	0.8068	0.8128	0.8126	0.7983	0.5044	0.4962	0.4215
Glass	214	7	0.4776	0.4703	0.4743	0.4793	0.4438	0.328	0.3712
Hayes-roth	132	4	0.4564	0.4493	0.5499	0.5104	0.5071	0.5174	0.5092
Balance-scale	625	3	0.4765	0.4681	0.4833	0.4462	0.5646	0.595	0.6028
Customers	440	3	0.5239	0.5237	0.526	0.5274	0.5474	0.4644	0.4541

5.2 K-Mean Clustering

K-mean clustering is a method for cluster analysis in data mining. It partitions n observations into K clusters with the nearest mean [43]. The proposed 16-bit R4ARBM2 is applied to calculate the squared deviation between points belonging to different clusters. The F-measure value [44] is used as the metric to evaluate the clustering results. It considers both the precision and the recall of the test. So, the F-measure score can be interpreted as a weighted average of the precision and recall. The best value of the F-measure score is 1 and its worst value is 0. Each F-measure value is the average of 50 experiments for each data set. In this work, several University of California Irvine (UCI) benchmark datasets [45] are selected to test the K-mean clustering using R4ARBM2.

The F-measure results are listed in Table 17. When $p\leq$24, the clustering results are similar as those processed with exact multipliers. For some approximate factors, the R4ARBM2 provides better results. Table 18 shows the comparison between the proposed R4ARBMs and R4ABMs with p=24 where the power is also measured with benchmark data. For the data sets of Iris, Glass, Hayes-roth, all R4ABMs produce similar results. For the data sets of Balance-scale and Customers, R4ABMs produce very accurate results. However, they also have higher energy compared with the proposed R4ARBMs.

Note that the K-means algorithm is sensitive to initial centroids. Hence, approximate multipliers could even achieve better accuracy than the accurate algorithm because the acceptable error introduced by approximate computing avoids overfitting the initial centroids.

5.3 High Dynamic Range (HDR) Image Processing

The proposed 32-bit approximate RB multipliers are applied to high dynamic range (HDR) OpenEXR images. OpenEXR is a HDR image file format developed by Industrial Light & Magic [46], which is widely used in computer imaging applications, including motion pictures and graphics.

TABLE 18
F-measure Values, Power, Delay and Energy Using Different Booth Multipliers with p=24. (The Same NS and NC as in Table 17)

Datasets	R4ERBM	R4ABM1 [21]	R4ABM2 [21]	R4ARBM1	R4ARBM2	R4ARBM3	R4ARBM4
Iris	0.8068	0.5050	0.5050	0.4971	0.5044	0.6424	0.4999
Power(μW)	378.2101	167.1571	151.1017	151.5664	135.1198	181.5521	157.5443
Delay(ns)	1.03	0.96	0.85	0.69	0.68	0.69	0.68
Energy(%)	100	41.2	33.0	26.8	23.6	32.2	27.5
Glass	0.4776	0.4211	0.4211	0.4127	0.4438	0.4225	0.4224
Power(μW)	341.7247	155.7819	134.6661	142.9088	125.7333	162.2852	140.1560
Delay(ns)	1.03	0.96	0.85	0.69	0.68	0.69	0.68
Energy(%)	100	42.5	32.5	28.0	24.3	31.8	27.1
Hayes-roth	0.4564	0.5241	0.5241	0.5252	0.5071	0.5038	0.5133
Power(μW)	551.0314	223.1198	259.0917	204.1002	188.5126	302.8586	261.7941
Delay(ns)	1.03	0.96	0.85	0.69	0.68	0.69	0.68
Energy(%)	100	37.7	38.8	24.8	22.6	36.8	31.4
Balance-scale	0.4765	0.6031	0.6031	0.5113	0.5646	0.5786	0.6022
Power(μW)	372.8424	166.7223	152.4608	150.6228	133.6219	178.6651	154.3847
Delay(ns)	1.03	0.96	0.85	0.69	0.68	0.69	0.68
Energy(%)	100	41.7	33.7	27.1	23.7	32.1	27.3
Customers	0.5239	0.717	0.717	0.5563	0.5474	0.533	0.525
Power(μW)	747.4084	346.5967	277.7754	308.3633	269.6311	308.1013	264.4449
Delay(ns)	1.03	0.96	0.85	0.69	0.68	0.69	0.68
Energy(%)	100	43.2	30.7	27.7	23.8	27.6	23.4

TABLE 19
P_det and Q_MOS Using R4ARBM2 with Different p

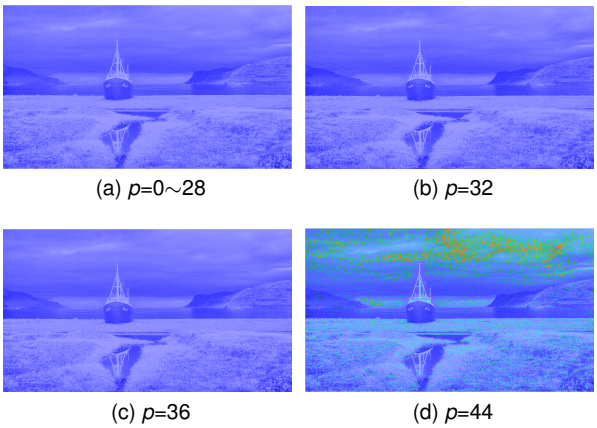
p=14	0 ∼ 28	32	36	40
P_det	0	6.04578×10^{-8}	0.00147516	1
Q_MOS	99.9999	99.9999	99.9999	99.9938

It supports 32-bit integer pixels, 16-bit floating-point, and 32-bit floating-point. The HDR visible difference predictor (HDR-VDP) [47] is a visual metric to evaluate approximate multipliers targeting HDR image processing applications; it compares a pair of images (reference and test images) and predicts the probability that the difference is visible to an average observer. HDR-VDP works within the complete range of luminance that the human eye can see and so it produces subjective comparison results.

Iceland.exr is used in this paper, and its data range is $0 \sim 10^7$. Two images are multiplied on a pixel-by-pixel basis to blend them into a single output image. The overall visibility, i.e., P_det, is defined as the probability that the differences between the images are visible for an average observer; the quality, i.e., Q_MOS, is defined as the degradation with respect to the reference image, expressed as a mean-opinion-score. P_det has a range of 0 to 1 and Q_MOS has a range of 0 to 100. A higher value of P_det means that it is more likely that a difference can be observed; a higher value of Q_MOS means that the image has a better quality. Therefore, Q_MOS is more relevant when evaluating the quality of a processed image. In this simulation, it is assumed that the diagonal display size is 12 inches, the resolution is 3200 by 1799, the viewing distance is 0.5 meters, and the color encoding is a sRGB display.

Table 19 shows the overall visibility (*i.e.*, P_det) and quality (*i.e.*, Q_MOS) of R4ARBM2 at different p values for Iceland.exr. The results for the overall visibility of R4ARBM2 with p<28 are all smaller than 10^{-15} so close to 0. These results show that the differences between the exact and approximate results are very small such that an observer cannot detect them. When p is larger than 40, P_det is 1, and the difference is easy to detect. The Q_MOS results using R4ARBM2 are all 99.999, showing that the quality of the processed images are very good.

The HDR-VDP detection maps of the images processed by R4ARBM2 at different p values are shown in Fig. 14. The default map is represented by a multiple color (blue, cyan, green, yellow and red) picture. Red denotes a high probability and blue denotes a low probability. As shown in Fig. 14, all differences for R4ARBM2 when p<36 are not visible to the average human observer.



(a) p=0∼28 (b) p=32 (c) p=36 (d) p=44

Fig. 14. HDR-VDP detection maps from R4ARBM2 with different p.

The P_dets of the processed image are almost zero when p<28. Furthermore, the proposed R4ARBMs are compared with the truncated multipliers (R4ABM04 [22], R4ABM11 [25] and R4ABM12 [26]), R4ABMs and other R4ARBMs at p=28. Table 20 shows the power, delay, energy, P_det and Q_MOS when using the respective multipliers in the HDR image processing application, where the power is also measured with benchmark data. The energy is compared with an exact RB multiplier.

The truncated multipliers (R4ABM04 [22], R4ABM11 [25] and R4ABM12 [26]) have lower energy dissipation but larger P_det compared with R4ABMs and R4ARBMs. All approximate multipliers have the same Q_MOS. R4ABM2 and R4ARBM2 generate smaller P_det so consistent with the NMED results. Although both R4ABM2 and R4ARBM2 generate the smallest P_det, the proposed R4ARBM2 has lower energy dissipation than R4ABM2.

6 Conclusions

This paper has studied the design of approximate redundant binary multipliers and the following conclusion can be drawn:

- The proposed approximate Booth encoders on conventional exact Booth encoders(R4AMBE6) and new exact Booth encoders (R4ANMBE6) have moderate error and energy consumption; also they achieve a very good tradeoff between error and performance.
- The two proposed RB 4:2 compressors (ARBC-1 and ARBC-2) reduce the energy consumption by over 47% and 64%, respectively, compared with ERBC.
- Four approximate RB multipliers (R4ARBM1, R4ARBM2, R4ARBM3 and R4ARBM4) have been designed based on approximate Booth encoders, approximate RB 4:2 compressors, (exact and approximate) regular partial product arrays, and approximate RB-NB converters. Error analysis and simulation show that the approximate RB multipliers are very good designs when considering the NEP as metric.

TABLE 20
P_det and Q_MOS of the Proposed Images Using Different 32-Bit Approximate Booth Multipliers With p = 28

p=28	R4ERBM	R4ABM04 [22]	R4ABM11 [25]	R4ABM12 [26]	R4ABM1 [21]	R4ABM2 [21]	R4ARBM1	R4ARBM2	R4ARBM3	R4ARBM4
Power (mW)	2.38	0.92	0.89	0.87	1.51	1.50	1.59	1.49	1.57	1.48
Delay (ns)	1.37	1.36	1.38	1.35	1.51	1.51	1.18	1.20	1.19	1.20
Energy (%)	100	38.4	37.7	36.0	70.0	69.6	57.6	54.9	57.1	54.3
P_det (10^{-8})	0	7.73973	0.0449954	0.14592	$2.22045*10^{-8}$	$4.44089*10^{-8}$	$2.55351*10^{-7}$	$4.44089*10^{-8}$	$6.4726*10^{-6}$	$4.55191*10^{-7}$
Q_MOS	100	99.9999	99.9999	99.9999	99.9999	99.9999	99.9999	99.9999	99.9999	99.9999

- Three case studies of error-resilient applications together with benchmark data have also been presented to show the validity of the proposed designs.

The proposed approximate redundant binary multipliers are efficient for error-tolerant applications with high accuracy.

ACKNOWLEDGMENTS

This work is supported by grants from the National Natural Science Foundation of China (61871216 and 61401197), the Six Talent Peaks Project in Jiangsu Province (2018XYDXX-009) and the USA National Science Foundation under Grant no. 1812467.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," in Proc. 18th IEEE European Test Symposium, 2013, pp.1-6.
[2] V. Chippa, S. Chakradhar, K. Roy and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing,"in Proc. 50th Annual Design Automation Conference (DAC), 2013, Article 113, 9 pages.
[3] S. Venkataramani, S. Chakradhar, K. Roy and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in Proc. 52nd Annual Design Automation Conference (DAC), 2015, Article 120, 6 pages.
[4] Q. Xu, N. S. Kim and T. Mytkowicz, "Approximate computing: a survey," IEEE Design and Test, vol. 33, no.1, pp. 8-22, 2016.
[5] Y. Wang, H. Li, X. Li, "Real-time meets approximate compureting: An elastic CNN inference accelerator with adaptive trade-off between QoS and QoR," in Proc. 54th Annual Design Automation Conference (DAC), 2017.
[6] H. Jiang, J. Han and F. Lombardi, "A comparative review and evaluation of approximate adders," in Proc. 25th IEEE/ACM Great Lakes Symposium on VLSI, 2015, pp. 343-348.
[7] S.-L. Lu, Speeding up processing with approximation circuits, Computer, vol. 37, no. 3, pp. 67-73, 2004.
[8] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo and Z. H.'Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," IEEE Trans. VLSI Syst., vol. 18, no. 8, pp.1225-1229, 2010.
[9] K. Du, P. Varian and K. Mohanram, "High-performance reliable variable latency carry select addition," in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2012, pp. 1257-1262.
[10] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," IEEE Trans. Circuits Syst.: Part I Regular Papers, vol. 57, no. 4, pp. 850-862, 2010.
[11] M. Imani, D. Peroni, and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," in Proc. 54th Annual Design Automation Conference (DAC), 2017, pp. 18-22.
[12] J. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electronic Computers, vol. EC-11, no. 4, 512-517, 1962.
[13] J. Low and C. Jong, "Unified mitchell-based approximation for efficient logarithmic conversion circuit," IEEE Trans. Computers, vol. 64, no. 6, pp. 1783-1797, 2015.
[14] M. Sullivan and E. E. Swartzlander, Jr., "Truncated error correction for flexible approximate multiplication," in Proc. Conf. Record the 46th Asilomar Conf. Signals, Systems and Computers, 2012, pp. 355-359.
[15] K. Y. Kyaw, W. L. Goh and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in Proc. Electron Devices and Solid-State Circuits, 2010, pp. 1-4.
[16] S. Hashemi, R. Bahar and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in Proc. IEEE/ACM International Conference on Computer Design, 2015, pp. 418-425.
[17] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in Proc. 24th Int. Conf. VLSI Design, 2011, pp. 346-351.
[18] K. Bhardwaj, P. Mane and Jörg Henkel, "Power-and area-efficient approximate wallace tree multiplier for error-resilient systems," in Proc. 15th IEEE Int. Symp. Quality Electronic Design, 2014, pp. 263-269.
[19] H. Jiang, J. Han, F. Qiao, F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high performance operation," IEEE Trans. Computers, vol. 65, pp. 2638-2644, Aug. 2016.
[20] V. Leon, G. Zervakis, D. Soudris and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 3, pp. 421-430, 2018.
[21] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," IEEE Trans. Computers, vol. 66, pp.1435-1441, Aug. 2017.
[22] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low error fixed-width modified Booth multiplier," IEEE Trans. VLSI Systems, vol. 12, no. 5, pp. 522-531, 2004.
[23] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," in Proc. Workshop VLSI Signal Process. VI, 1993, pp. 388-396.
[24] E. J. King and E. E. Swartzlander Jr., "Data dependent truncated scheme for parallel multiplication," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., 1998, pp. 1178-1182.
[25] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified Booth multipliers for lossy applications," IEEE Trans. VLSI Systems, vol. 19, no. 1, pp. 52-60, 2011.
[26] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional-probability estimator for fixed-width Booth multipliers," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 59, no. 3, pp. 594-603, 2012.
[27] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 10, pp. 3105-3117, 2016.
[28] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in Proc. International Conference on Computer Design, 2013, pp. 33-38.
[29] A. Momeni, J. Han, P. Montuschi and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Computers, vol. 64, no. 4, pp. 984-994, 2015.
[30] C. Liu, J. Han and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. DATE, 2014, pp. 1-4.
[31] N. Maheshwari, Z. Yang, J. Han and F. Lombardi, "A design approach for compressor based approximate multipliers," in Proc. Int. Conf. VLSI Design and Embedded Systems, 2015, pp. 209-214.
[32] Z. Yang, J. Han and F. Lombardi, "Approximate compressors for error-resilient multiplier design," in Proc. IEEE Int. Symp. Defect

and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2015, pp. 183-186.
[33] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, "Synthesizing parsimonious inexact circuits through probabilistic design techniques," ACM Trans. Embed. Comput. Syst., vol. 12, no. 2, Article 93, 26 pages, May 2013.
[34] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in Proc. Design Automation Conference (DAC), 2012, pp. 796-801.
[35] 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, pp. 773-783, 1996.
[36] X. Cui, W. Liu, E. E. Swartzlander Jr. and F. Lombardi, "A modified partial product generator for redundant binary multipliers," IEEE Trans. Comput., vol. 65, no. 4, pp. 1165-1171, Apr. 2016.
[37] T. Cao, W. Liu, C. Wang, X. Cui and F. Lombardi, "Design of approximate redundant binary multipliers," in Proc. 12th ACM/IEEE Int. Symp. Nanoscale Architectures (NANOARCH), 2016, pp. 31-36.
[38] A. Booth, "A signed binary multiplication technique," The Quarterly J. Mechanical and Applied Math., vol. 4, pp. 236-240, 1951.
[39] W. Yeh and C. Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, pp. 692-701, Jul. 2000.
[40] Y. Kim, B. Song, J. Grosspietsch, and S. F. Gillig, "A carry-free 54b×54b multiplier using equivalent bit conversion algorithm," IEEE J. Solid-States Circuits, vol. 36, pp. 1538-1545, 2001.
[41] G. Wang and M. Tull, "A new redundant binary number to 2's-complement number converter," in Proc. Region 5 Conference: Annual Technical and Leadership Workshop, pp. 141-143, 2004.
[42] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Computers, vol. 63, pp. 1760-1771, Sep. 2013.
[43] E. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," Biometrics, vol. 21, pp. 768-769, 1965.
[44] L. Rushi, D. Snehlata and M. Latesh, "Class imbalance problem in data mining: review," Int. J. Computer Science and Network, vol. 2, pp. 83-87, 2013.
[45] Kevin Bache and Moshe Lichman, UCI machine learning repository, http: //archive.ics.uci.edu/ml, 2013.
[46] Industrial Light & Magic, http://www.openexr.com, 2014.
[47] R. Mantiuk, K. Kim, A. Rempel and W. Heidrich, "HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions," ACM Transactions on Graphics, vol. 30, no. 4, pp. 76-79, 2011.

Weiqiang Liu (S'10-M12-SM15) received the B.Sc. degree in Information Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China and the Ph.D. degree in Electronic Engineering from the Queens University Belfast (QUB), Belfast, UK, in 2006 and 2012, respectively. In Dec. 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently an Associate Professor. He has published one research book by Artech House and over 70 leading journal and conference papers. One of his papers was the Most Popular Article of IEEE TC in July 2017 and one was selected as the Feature Paper of IEEE TC in the 2017 December issue. His paper was also the Best Paper Candidates of ISCAS 2011 and GLSVLSI 2015. He serves as an Associate Editor of IEEE Transactions on Computers (TC), the leader of The Multimedia Team at TC Editorial Board, a Steering Committee Member of IEEE Transactions on Multi-Scale Computing Systems (TMSCS), the Guest Editors of IEEE Transactions on Emerging Topics in Computing (TETC) and Elsevier Microelectronics Journal. He has been a program committee member for several conferences including ARITH, ISCAS, ASAP, ISVLSI, NANOARCH and ICONIP. He is a member of both CASCOM and VSA Technical Committee, IEEE CAS Society. His research interests include computer arithmetic, approximate computing, emerging technologies in computing systems and hardware security.

Tian Cao received the BSc degree in Microelectronics from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2015 and the MEng dgree in circuits and system from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China in 2018. He is currently a chip design engineer at MTK. His research interest includes approximate computing, Channel coding for new radio of 5th generation mobile communications.

Peipei Yin received the B.Sc. and M.Sc. degrees from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2010 and 2013, respectively. Since Sep. 2015, she has been pursuing the Ph.D. degree in the College of Electronic and Information Engineering, NUAA, Nanjing, China. Her research interests include computer arithmetic, fault tolerance systems, and low power technologies in approximate computing.

Yuying Zhu received the BSc degree in Electronic Information Science and Technology from Central South University (CSU), Changsha, China, in 2017. Since September 2017, She has been working toward the MEng degree in circuits and system, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. Her research interests include approximate computing and VLSI design.

Chenghua Wang received the B.Sc. and M.Sc. degrees from Southeast University, Nanjing, China, in 1984 and 1987, respectively. In 1987, he joined the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, where he became a full Professor in 2001. He has published 6 books and over 100 technical papers in journals and conference proceedings. He is the recipient of more than ten teaching and research awards at the provincial and ministerial level. His current research interests include testing of integrated circuits, and circuits & systems for communications.

Earl E. Swartzlander Jr. (S'64-M'72-SM'79-F'88-LF'11) received the B.S. degree from Purdue University in 1967, the M.S. degree from the University of Colorado in 1969, and the Ph.D. degree from the University of Southern California in 1972, all in electrical engineering. He is a professor of electrical and computer engineering at the University of Texas at Austin. In this position, he and his students conduct research in computer engineering with emphasis on application-specific processor design, including high-speed computer arithmetic, embedded processor architecture, VLSI technology, and nanotechnology. As of December 2016, he has supervised 46 Ph.D. students. He is the author of two books, editor of 11 books and the author or coauthor of 84 refereed journal papers, 40 book chapters, and 303 conference papers. He was the editor-in-chief of the IEEE Transactions on Computers from 1990 to 1994 and was the founding editor-in-chief of the Journal of VLSI Signal Processing. In addition, he has served as an associate editor for the IEEE Transactions on Computers, the IEEE Transactions on Parallel and Distributed Systems, and the IEEE Journal of Solid-State Circuits. He has been a member of the Board of Governors of the IEEE Computer Society (1987-1991), the IEEE Signal Processing Society (1992-1994), and the IEEE Solid-State Circuits Council/Society (1986-1991). He has been a member of the IEEE History Committee (1996-2004), the IEEE Fellows Committee (2000-2003), the IEEE James H. Mulligan, Jr., Education Medal Committee (2007-2011), the IEEE Awards Planning and Policy Committee (2011-2013), the IEEE Awards Board Awards Review Committee (2014 to present), and the IEEE Awards Board (2015 to present). He has chaired a number of conferences. He is a Life Fellow of the IEEE. He has been honored with the IEEE Third Millennium Medal, the Distinguished Engineering Alumnus Award from the University of Colorado, the Outstanding Electrical Engineer and Distinguished Engineering Alumnus Awards from Purdue University, and the IEEE Computer Society Golden Core Award.

Fabrizio Lombardi (M81-SM02-F'09) graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. from the University of London (1982). He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. In the past Dr. Lombardi was the Editor-In-Chief of the IEEE Transactions on Computers and the inaugural Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computing. Currently, he is the Editor-in-Chief of the IEEE Transactions on Nanotechnology . In 2019, he will serve as the Vice President for Publications of the IEEE Computer Society. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books. He is a Fellow of IEEE.