High Performance Parallel Decimal Multipliers using Hybrid BCD Codes

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

Abstract—A parallel decimal multiplier with improved performance is proposed in this paper by exploiting the properties of three different binary coded decimal (BCD) codes, namely the redundant BCD excess-3 code (XS-3), the overloaded decimal digit set (ODDS) code and the BCD-4221/5211 code. The signed-digit radix-10 recoding is used to recode the BCD multiplier to the digit set [-5, 5] from [0, 9]. The redundant BCD XS-3 code is adopted to generate the multiplicand multiples in a carry-free manner. The XS-3 coded partial products (PPs) are converted to ODDS PPs to fit binary partial product reduction (PPR). In this paper, a regular decimal PPR tree using ODDS and BCD-4221/5211 codes is proposed; it consists of a binary PPR tree block, a non-fixed size BCD-4221 counter block and a BCD-4221/5211 PPR tree block. The decimal carry-save algorithm based on BCD-4221/5211 is used in the PPR tree to obtain high performance multipliers. Moreover, an improved PPG circuit and an improved parallel prefix/carry-select decimal adder are proposed to further improve the performance of the proposed multipliers. Analysis and comparison using the 45 nm technology show that the proposed decimal multipliers are faster and require less hardware area than previous designs found in the technical literature.

Index Terms—Parallel Decimal Multiplication, Overloaded BCD Representation, Redundant Excess-3 Code, Redundant Arithmetic

1 Introduction

As binary arithmetic can introduce conversion and rounding errors, decimal arithmetic is in increased demand for many commercial applications, such as financial computations, currency conversion, insurance and accounting [1-2]. The decimal floating-point arithmetic for Decimal64 (16-digit) and Decimal128 (34-digit) numbers have been defined and added to the revised IEEE 754-2008 standard [3-4], thus attracting a significant research interest in decimal arithmetic architectures and implementations [5-16]. Hardware decimal computation is also implemented in IBM microprocessors [17-18]. Decimal adders and multipliers are the most important primitives for decimal arithmetic and recently, they have been extensively studied.

A parallel decimal multiplication consists of three main stages: the decimal partial product generation (PPG), the decimal partial product reduction (PPR) and the final carry propagate decimal addition. The sign-digit (SD) encoding [7-8], [19-20], the non-redundant and redundant BCD digits encoding are usually used in decimal multipliers design. The non-redundant decimal formats (such as BCD-8421, BCD-4221, BCD-5211) are used in [9-10], [12-15] for PPG and PPR. A variety of redundant decimal formats (such as the redundant BCD XS-3 digit, ODDS digit) have also been proposed to improve the performance of BCD multiplication [7-8], [11], [21]. Due to its self-complementing property [22-23], the BCD-4221 and redundant BCD excess-3 (XS-3) code [-3, 12] has been used in the PPG circuit [8], [13-14]. The overloaded decimal digit set (ODDS [0, 15]) is used in the PPG stage because it can be added using a regular binary PPR tree.

The multiplicand X and the multiplier Y are encoded by BCD (each decimal digit $X_i \epsilon [0,9]$ and $Y_i \epsilon [0,9]$ are represented in a 4-bit binary digit system). In a PPG stage, the SD radix-10 encoding, SD radix-5 encoding and double-BCD encoding have been proposed for the multiplier in [7-9], [12-14]. The BCD number is recoded to the digit set [-5, 5] from [1, 9] in SD radix-10 recoding. The BCD-4221/5211 recoding is used in the PPG stage for generating the multiplicand multiples (1X, 2X, 3X, 4X, and 5X) [13-14]. The SD radix-10 encoding and the BCD-4221/5211 recoding have been proposed in [13-14], [25] to improve the performance of a PPG circuit. The disadvantage of the BCD-4221/5211 codes in a PPG is that the 3X multiples cannot be obtained in a carry-free manner by using the non-redundant radix-10 digit set [8].

Both the ODDS and the redundant XS-3 representations have been used in the PPG stage to further improve the performance of parallel decimal multipliers [8], [15]. An advantage of the XS-3 representation over the BCD-4221/5211 representation is that all relevant multiples (such as 1X, 2X, 3X, 4X, and 5X) can be implemented in a carry-free manner. The XS-3 PPs can also be represented as ODDS PPs by adding a pre-computed correction term. The PPR stage is defined as the ODDS PPR tree when its input signals are the ODDS digits.

- *X. Cui, W. Dong and W. Liu are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China. E-mail: {wnhcxp, wwdong, liuweiqiang}@nuaa.edu.cn.*
- *E.E. Swartzlander, Jr. is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, Texas, 78712, USA. E-mail: eswartzla@aol.com.*
- *F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA. E-mail: lombardi@ece.neu.edu.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

A Decimal carry-save adder (CSA) is usually used in a decimal multiplier for the PPR stage [5-6], [9], [21], [24-25]. A CSA based on BCD-4221/5211 codes has been used in [13-15]; a 4-bit binary CSA and a decimal × 2 operation are used. The 4-bit decimal CSA is very regular and has been used to improve the performance of decimal multipliers [13-15], [25-26].

The ODDS PPR tree is usually used for implementing fast and efficient decimal multipliers. The ODDS digit PPs can be compressed with a binary PPR tree that does not require to correct the invalid six 4-bit digits [8], [11], [15]. However, all decimal carries (also defined as digit carries) generated by the binary CSA tree are required to be corrected by +6 operations. In [8], the results of the binary compression are corrected by a digit carry counter block, the ODDS PPR correction block and a special 3:2 compressor block. Finally, the ODDS PPs are compressed until two decimal PP rows remain. The results are accumulated by a decimal carry propagate addition.

This paper is an extension of our previous paper [15]. In [15], all parts in the PPR tree are regular binary CSAs and regular decimal CSAs based on BCD-4221/5211 codes. The main differences and contributions compared with [15] are summarized as follows:

- an improved PPG circuit based on redundant BCD coding is proposed for partial product generation;
- an improved parallel prefix/carry-select decimal adder is proposed for the final adder to further improve the performance of the multiplier.
- the design of a 16×16-digit decimal multiplier based on a modified PPG circuit, a modified decimal adder and the previous 17:2 ODDS PPR tree of [15] is presented in more detail and with updated figures.
- a detailed design of a new 34×34-digit decimal multiplier based on the improved PPG circuit, the improved decimal adder and the new 35:2 ODDS PPR tree is also presented.

A 34×34-digit multiplier was not presented in [14] and [15], however it is implemented in this paper by following the design method of [14] and [15]. The improved design for the 16×16-digit multiplier and the new design for the 34×34-digit multiplier achieve a smaller delay compared with existing decimal multipliers.

The paper is organized as follows. Section 2 reviews the BCD representation, the SD Radix-10 recoding and the decimal radix-10 multiplication based on non-redundant BCD-4221/5211. The redundant XS-3 code, the ODDS code and SD radix-10 multiplication using a redundant BCD representation are reviewed in Section 3. The PPG and the PPR tree of previous works are also reviewed in Section 3. The improved PPG circuit and the decimal adder are presented in Section 4; also, the proposed PPR tree for the 16×16-digit multiplier and a new PPR tree for the 34×34-digit multiplier are discussed. In Section 5, complexity and delay are evaluated and compared with other decimal implementations found in the technical literature. Finally, the conclusion is provided in Section 6.

2 Radix-10 Multiplication Using Non-Redundant BCD-4221/5211 Codes

In this section, the BCD representation and the decimal radix-10 multiplication based on non-redundant BCD-4221/5211 are reviewed. The SD radix-10 recoding, the decimal multiplicand multiples and the decimal CSA based on BCD-4221/5211 in [14] are discussed.

The decimal digit consists of the weighted 4-bit codes, the non weighted 4-bit codes and the codes involving 5 or more bits. [23] has discussed the differences between BCD codes in more detail. A d-digit decimal integer operand with weighted 4-bit codes is expressed as:

$$Z = \sum_{i=0}^{i=d-1} Z_i \times 10^i \quad (1)$$

$$Z_i = \sum_{j=0}^{3} z_{i,j} \times r_j \quad (2)$$

where $Z_i \epsilon [0,9]$ is the ith decimal digit, $z_{i,j}$ is the jth bit of the ith digit, and $r_j (0 \le j \le 3)$ is the weight of the jth bit. The most commonly used decimal weighted 4-bit codes include BCD-8421, BCD-5421, BCD-4221, and BCD-5211. Codes with the self-complementing property are the BCD-4221, XS-3, 642(-3), 753(-6), [23]. In the self-complementing decimal codes, the 9's complement of each decimal digit is obtained by inverting the bits in the coded representation of that digit.

The multiplicand $X (X = X_{d-1}X_{d-2} \dots X_1X_0)$ and the multiplier $Y (Y = Y_{d-1}Y_{d-2} \dots Y_1Y_0)$ are assumed to be unsigned BCD decimal integers of d digits each. The product $P = X \times Y$ is in a non-redundant BCD format with $2d$ digits. The decimal radix-10 recoding is used in [7], [13-14] for designing a high performance parallel decimal multiplier; it leads to a simple generation of the partial products.

The SD radix-10 recoding transforms a 4-bit BCD multiplier digit Y_i ($Y_i \in \{0, 1, \dots, 8, 9\}$) into an SD radix-10 digit Yb_i ($Yb_i \in \{-5, -4, \dots, 4, 5\}$). The value of the recoded digit Yb_i depends on the decimal value of Y_i and the sign Ys_{i-1} ($Ys_{i-1} = 1$, if $Y_{i-1} \ge 5$). Each digit Yb_i is represented by five multiple select signals $\{Y1_i, Y2_i, Y3_i, Y4_i, Y5_i\}$ and a sign bit Ys_i. Multiple select signals are used as a selection control signals for the 5:1 multiplexers (MUXs) to select the positive multiplicand multiples $\{X, 2X, 3X, 4X, 5X\}$. Due to self-complementing property of the BCD-4221 code, when $Ys_i = 1$, a negative multiplicand multiple $\{-X, -2X, -3X, -4X, -5X\}$ is generated by bit inversion of the positive BCD-4221 coded multiple and adding 1 at the least significant bits (LSB). The bit inversion operand is performed by using the XOR gates controlled by Ys_i (when $Ys_i = 0$, no inversion; otherwise inversion takes place). Therefore, p ($p = d + 1$) PP rows are generated in the SD radix-10 decimal multiplier; each partial product $PP[i]$ is at most (d+3)-digits in length [13-14].

In [13-14], the positive multiplicand multiples $\{X, 2X, 3X, 4X, 5X\}$ are coded in BCD-4221. $2X$ is generated by the following three steps: converting BCD-8421 to BCD-5421, BCD-5421 is left shifted 1-bit to BCD-8421 and converting BCD-8421 to BCD-4221. $4X$ is generated by $2X$×2. $5X$ is obtained by a 3-bit left shifted BCD-4221 to BCD-5211 and then converting BCD-5211 to BCD-4221.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

The decimal carry-save algorithm based on BCD-4221/5211 is used for partial product reduction [13-14]. Each column of the p PP rows is compressed to two digits by a decimal digit p:2 CSA tree; decimal carries are passed between adjacent digit columns. The decimal digit (4-bit) 3:2 CSA is made of four full adders and a BCD-4221 to BCD-5211 converter. The decimal digit 3:2 CSA is used to generate the decimal digit p:2 CSA tree to reduce p PP rows to two decimal digits H_i and S_i.

According to [13-14], the addition for the three ith decimal digits can be expressed as:

$$A_i + B_i + C_i = \sum_{j=0}^{3}(a_{i,j} + b_{i,j} + c_{i,j})r_j$$

$$= \sum_{j=0}^{3} s_{i,j}r_j + 2 \times \sum_{j=0}^{3} h_{i,j}r_j = S_i + 2 \times H_i \quad (3)$$

where, the weight of $r_3r_2r_1r_0$ is 4221 or 5211, $s_{i,j}$ and $h_{i,j}$ are the sum and carry bits of a full adder (also defined as a 3:2 CSA or a 3:2 compressor). $S_i \in [0,9]$ and $H_i \in [0,9]$ in Eq. (3) are the decimal sum and carry digits at position i. Fig. 1 shows an example of a decimal 3:2 CSA based on BCD-4221. The binary carries $H_i \in [0,9]$ ($h_{i,3}$, $h_{i,2}$, $h_{i,1}$, $h_{i,0}$) generated by the 4-bit 3:2 CSA are required for multiplication by 2 (i.e., × 2). The $2 \times H_i$ can be achieved by converting the BCD-4221 to BCD-5211s (expressed by W_i) and left shifting W_i by 1-bit. The resultant digit is coded as BCD-4221s. The definition of BCD-4221s/5211s and the difference between BCD-4221s/5221s and BCD-4221/5221 is reported in [13].

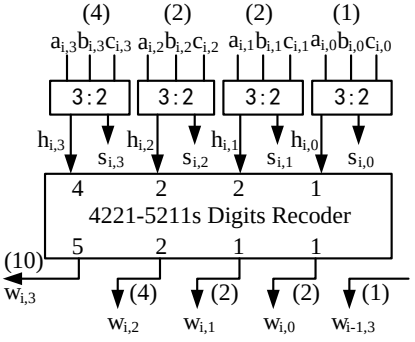

Fig. 1. Decimal 3:2 CSA for operands coded in BCD-4221[13].

The bits of W_i are expressed as $w_{i,j}$ in which the most significant bit $w_{i,3}$ represents the decimal carry out (with a weight of 10) to the next higher decimal digit, while $w_{i-1,3}$ is a decimal carry input from the next lower decimal digit. Thus, the $2 \times H_i$ at the ith digit is given by:

$$(2 \times H)_i = w_{i,2}\,4 + w_{i,1}\,2 + w_{i,0}\,2 + w_{i-1,3}\,1 \quad (4)$$

In [13-14], the adder setup is used to generate $2 \times H$ and also convert the BCD-4221 PPs and S to the BCD-8421 PPs (defined as A) and BCD excess-6 PPs (defined as B), respectively. The final two PPs (A and B) are accumulated by a decimal carry propagate adder, which is actually a $2d$-digit hybrid parallel prefix/carry-select adder [13-14], [16], [27-29].

3 Radix-10 Multiplication Using Redundant BCD Codes

Both BCD-4221 and BCD-5211 decimal encodings are used for partial product generation and reduction [13-14]. As mentioned previously, a disadvantage of the BCD-4221/5211 codes is that they use a non-redundant radix-10 digit set [0, 9], in which $3X$ cannot be obtained in a carry-free manner. The decimal SD representations rely on a redundant digit set $\{-a, \cdots, 0, \cdots a\}$ $(5 \le a \le 9)$ to allow a decimal carry-free addition [7-8], [11], [19-20].

In this section, we present a general overview of the SD radix-10 multiplication using redundant BCD codes [8]. The multiplier of [8] consists of three main stages: the radix-10 PPG stage based on redundant XS-3 and ODDS digits, the ODDS PPR tree and the BCD adder.

3.1 Redundant BCD Representations and the SD Radix-10 Decimal Multiplier Architecture

The redundant BCD representation is given by:

$$Z = -s_z \times 10^i + \sum_{i=0}^{d-1} Z_i \times 10^i \quad (5)$$

where d is the number of decimal digits, s_z is the sign bit, and $Z_i \in [l-e, m-e]$ is the ith digit, with $0 \le l \le e, 9 + e \le m \le 15$ in which e is the excess of the representation and usually e=0 (non-excess), 3 or 6. The redundancy index ρ is defined as $\rho = m - l + 1 - 10$ [11].

The different representations of Z_i are given as follows:

(1) BCD: $Z_i \in [0,9], e = 0,\ l = 0,\ m = 9,\ \rho = 0$;
(2) BCD excess-3: $Z_i \in [0,9], e = 3, l = 3, m = 12,\ \rho = 0$;
(3) BCD excess-6: $Z_i \in [0,9], e = 6, l = 6, m = 15,\ \rho = 0$;
(4) ODDS: $Z_i \in [0,15], e = 0, l = 0, m = 15,\ \rho = 6$;
(5) XS-3: $Z_i \in [-3,12], e = 3, l = 0, m = 15,\ \rho = 6$.

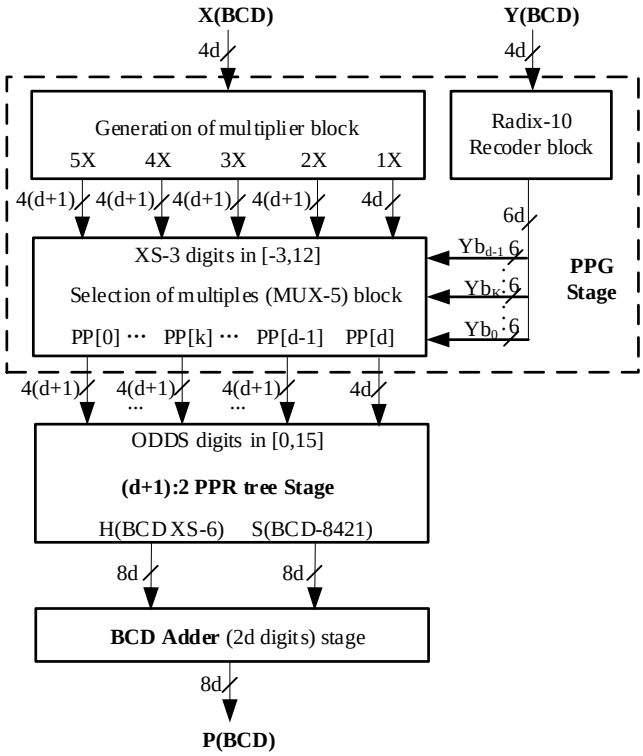

Fig. 2. The radix-10 decimal multiplier architecture using redundant BCD codes [8].

The SD radix-10 decimal multiplier using redundant BCD codes [8] is shown in Fig. 2. The SD radix-10 recoding and XS-3 code are used for a fast PPG stage, in which positive multiplicand multiples $\{0X, 1X, 2X, 3X, 4X, 5X\}$ are pre-computed in a carry-free

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

manner; negative multiples can be obtained by bit inversion of the corresponding positive ones and adding +1 to the LSB.

The XS-3 PPs can be represented as ODDS PPs by adding pre-computed correction terms (*i.e.*, adding all -3 constants). The (d+1) PP rows based on ODDS representation can be compressed up to the two 2d-digit decimal PPs by using a (d+1):2 ODDS PPR tree. These two PP rows are accumulated by a final decimal carry propagate addition. The detailed analysis about PPG circuit and decimal adder are presented in Section 3.2 and 3.4, respectively.

TABLE 1
THE XS-3 RECODING TABLE OF MULTIPLICAND MULTIPLES [8].

	1X			2X			3X			4X			5X		
Xi	Xi+3	Ti	Di	(Xi×2)+3	Ti	Di	(Xi×3)+3	Ti	Di	(Xi×4)+3	Ti	Di	(Xi×5)+3	Ti	Di
0	3	0	3	3	0	3	3	0	3	3	0	3	3	0	3
1	4	0	4	5	0	5	6	0	6	7	0	7	8	0	8
2	5	0	5	7	0	7	9	0	9	11	1	1	13	1	3
3	6	0	6	9	0	9	12	0	12	15	1	5	18	1	8
4	7	0	7	11	1	1	15	1	5	19	1	9	23	2	3
5	8	0	8	13	1	3	18	1	8	23	2	3	28	2	8
6	9	0	9	15	1	5	21	2	1	27	2	7	33	3	3
7	10	0	10	17	1	7	24	2	4	31	2	11	38	3	8
8	11	0	11	19	1	9	27	2	7	35	3	5	43	4	3
9	12	0	12	21	1	11	30	2	10	39	3	9	48	4	8

3.2 Decimal Partial Product Generation based on ODDS Representation

The partial product generation stage consists of the SD radix-10 recoder, the generation of the multiplicand multiples and the selection of the multiples three blocks.

The XS-3 recoding [-3, 12] is used [8] to generate the multiplicand multiples NX (N=1, 2, 3, 4, 5) in a carry-free manner. The XS-3 recoding table of the multiplicand multiples is given in Table 1. The XS-3 digits NX_i are given by:

$$N \times X_i + 3 = D_i + 10 \times T_i \tag{6}$$

where T_{i} is the decimal carry ($0 \le T_{\mathrm{i}} \le T_{\max}$). $T_{\max}$ is the maximum possible value for the decimal carry (such as $T_{\max} = 4$ for 5X). D_{i} is a decimal digit with $-3 \le D_{\mathrm{i}} \le 12 - T_{\max}$. The ith decimal XS-3 digits NX_i are given by:

$$[NX_i] = D_i + T_{i-1} \qquad ([NX_i] \epsilon [0,15]) \tag{7}$$

The negative operation for $[NX_i]$ can be achieved by inverting each bit of $[NX_i]$ and adding '1' to the LSBs; the XS-3 decimal digit can be represented as an ODDS decimal digit by adding all the pre-computed correction terms -3.

The decimal PP array generated for the 16×16-digit radix-10 multiplier using redundant XS-3 codes is shown in Fig.3. The SD radix-10 encoding of the multiplier produces d SD radix-10 digits $Yb_k \in [-5,5]$ (k=0,1,...d-1). As mentioned in Section 2, each digit Yb_k is represented by $\{Y1_k, Y2_k, Y3_k, Y4_k, Y5_k, Ys_k\}$ in which $Y1_K, Y2_K, Y3_K, Y4_K, Y5_K, Ys_K$ are multiple select signals for the appropriate multiple $\{0X, 1X, 2X, 3X, 4X, 5X\}$ and Ys_i is the sign bit. The signals $Y1_k, Y2_k, Y3_k, Y4_k, Y5_k$ and Ys_k are as follows [8]:

$$Ys_K = y_k[3] + y_k[2] \cdot (y_k[1] + y_k[0]) \tag{8}$$

$$Y5_k = y_k[2] \cdot \overline{y_k[1]} \cdot (y_k[0]) \oplus Ys_{K-1}) \tag{9}$$

$$Y4_k = Ys_{K-1} \cdot y_k[0] \cdot (y_k[2]) \oplus y_k[1]) + \overline{Ys_{K-1}} \cdot y_k[2]) \cdot \overline{y_k[0]}) \tag{10}$$

$$Y3_k = y_k[1] \cdot (y_k[0]) \oplus Ys_{K-1}) \tag{11}$$

$$Y2_k = \overline{Ys_{K-1}} \cdot \overline{y_k[0]}(y_k[3]) + \overline{y_k[2]} \cdot y_k[1]) + Ys_{K-1} \cdot \overline{y_k[3]} \cdot y_k[0] \cdot \overline{y_k[2]) \oplus y_k[1])} \tag{12}$$

$$Y1_k = \overline{y_k[2] + y_k[1]} \cdot (y_k[0]) \oplus Ys_{K-1}) \tag{13}$$

The 5:1 multiplexers select the multiplicand multiple and XOR gates are used to invert the multiplicand multiple if $Ys_k = 1$. The d+1 PP rows (*PP[0],..., PP[d])* are generated by the SD radix-10 recoding.

The $PP_d[k]$ is the most significant digit (MSD) of the kth $PP[k]$ and is given by:

$$PP_d[k] = -Ys_K + (-1)^{-Ys_K} \cdot T_{d-1} \tag{14}$$

Where $0 \le T_{d-1} \le 4$, $0 \le PP_d[k] \le 4$ with $Ys_K = 0$ and $-5 \le PP_d[k] \le -1$ with $Ys_K = 1$. To ensure $PP_d[k]$ is always positive, $\lceil PP_d[k] \rceil = PP_d[k] + 8$ is considered [8]. Therefore, $8 \le \lceil PP_d[k] \rceil \le 12$ with $Ys_K = 0$ and $3 \le [PP_d[k]] \le 7$ with $Ys_K = 1$. $\lceil PP_d[k] \rceil$ is given as follows:

$$[PP_d[k]] = \begin{cases} 8 + T_{d-1} & (Ys_K = 0) \\ 7 - T_{d-1} & (Ys_K = 1) \end{cases} \tag{15}$$

All of the -8 terms and -3 terms of the different partial products are pre-computed as constants. The pre-computed results are given for d=16 and d=34 as follows:

$f_c(16) =$ 0740740740740741703703703703 7037

$f_c(34) =$ 074074074074 ... 07417037037 ... 037

As shown in Fig. 3, Digits labeled as O represent the redundant ODDS digits ($PP_i[k] \in [0,15]$, $0 \le i \le d-1$ and $0 \le k \le d-1$). Digits labeled as S_k represent the MSD of each PP (i.e., $PP_d[k]$). The H_k is composed of Ys_K and the corresponding digit of the correction term $f_c(16)$ (*i.e.*, $H_k = Ys_K + \{0,3,7\}$ with $H_k \in [0,8]$). Digits labeled as F in Fig. 3, represent the last PP row (*i.e.*, $PP[d]$), where $F_i = PP_i[d] = f_c(16)$, if $Ys_K = 0$ and $F_i = PP_i[d] = X_i + f_c(16)$ if $Ys_K = 1$.

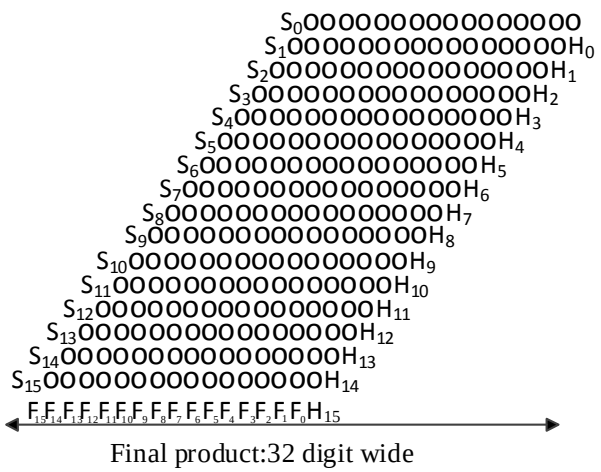

Fig. 3. The decimal partial product array generated for a 16x16-digit radix-10 multiplier using redundant BCD codes ([8]).

3.3 ODDS Partial Product Reduction Tree

When the decimal partial product array is based on BCD-8421 and the ODDS digits, the binary CSA tree can be used in a decimal PPR tree to design high speed and low complexity decimal multipliers. However, the results of the binary compression must be corrected. A digit carry counter block is used in the PPR tree to correct the digit carries per decimal column [8-9], [11-12], [15]. The BCD-8421 counter has been proposed in [9], [12]. A 3 full adder latency stage is required in the 9:4 counter and a 2 full adder latency stage is required in the 6:3 counter [12]. The binary ODDS counter in [8], [11] is utilized in the ODDS PPR tree.

In [8], the (d+1) ODDS PPs are compressed by (d+1):2 ODDS PPR tree which consists of a binary CSA tree block, a digit carries counter block based on ODDS digit, a ODDS PPR correction block and a special decimal 3:2 compressor block. These PPs are compressed firstly by binary CSA tree and the $(d-1)$ digit carries (for the maximum digit column height) are generated. The detailed descriptions for (d+1):2 ODDS PPR tree are as follows:

(1) A regular binary CSA tree block is used to compute the decimal partial product sum. The binary CSA tree accepts an arbitrary number of ODDS or BCD-8421 operands as inputs. The resultant two PP rows (sum and carry) are in ODDS representation.

(2) At the same time, a digit carries counter block counts the digit carries generated by the binary CSA tree. The carries per decimal column are counted by the 4-bit binary ODDS counter and then a multiplication by six $(\times 6)$ is performed. The binary ODDS counter is composed of 3:2 CSAs [13] and a 3-bit carry-propagation adder.

(3) The ODDS PPs in block (1) and (2) are added by the ODDS PPR correction block and a special decimal digit 3:2 compressors to obtain the final double-word length (2d-digit) A and B, where A is represented with BCD excess-6 digits and B is represented with BCD-8421 digits.

The maximum number of digit carries transferred between adjacent columns of the binary 17:2 CSA tree is 15. These carries are labeled as $C_{i+1}[0:14]$ (carry output signals at the ith column) and $C_i[0:14]$ (carry input signals at the ith column). The binary ODDS counter circuit (counter resultant digit $\epsilon[0,15]$) consists of a four stage 3:2 compression and a 3-bit carry-propagation adder [8].

3.4 BCD Decimal Adder

The final non-redundant BCD-8421 sum P=A+B is achieved by a BCD decimal adder [16]. The conditional speculative decimal adder is used in [8], [12-16] and it includes a conditional +6 block which is controlled by a control signal r_{i}, a binary parallel prefix carry network block and 4-bit carry-select decimal adder blocks. A quaternary tree (QT) parallel prefix (PPF) carry network is used for carry-propagate generation [8], [13-16], [27]. A Kogge-Stone (KS) PPF carry network is used in [12] because it is faster than a QT tree [27-29].

If $A_{\mathrm{i}}^{\mathrm{U}}$ is defined as the first 3 most significant bits of a decimal digit A_{i}, $A_{\mathrm{i}}^{\mathrm{U}} + B_{\mathrm{i}}^{\mathrm{U}} \geq 8$ is a necessary (but not sufficient) condition for the generation of a decimal carry at ith 4-bit decimal digit [16].

The control signal r_{i} is the +6 condition signal [16] and it is obtained in terms of the binary carry generate signal $g_i[j]$ ($g_i[j] = a_i[j] \cdot b_i[j]$) and the carry propagate signal $p_i[j]$($p_i[j] = a_i[j] + b_i[j]$) as follows:

$$r_i = p_i[3] + g_i[2] + p_i[2] \cdot g_i[1] \tag{16}$$

$$r_i = \begin{cases} 1, & A_i^U + B_i^U \geq 8 \\ 0, & A_i^U + B_i^U < 8 \end{cases} \tag{17}$$

B_{i}^* is expressed as follows:

$$B_{\mathrm{i}}^* = \begin{cases} B_i + 6, & r_i = 1 \\ B_i, & r_i = 0 \end{cases} \tag{18}$$

The binary hybrid parallel prefix/carry-select adder block [27-28] is used for $A_i + B_i^*$. The decimal carries (c_{4i}, i=1,2,…, 2d-1) are computed using a PPF carry network, while the two conditional BCD digit sums are computed by using 4-bit decimal carry select adders which implement $A_i + B_{\mathrm{i}}^* + 0$ and $A_i + B_{\mathrm{i}}^* + 1$. These two conditional sums are selected by c_{4i}. If the decimal carry output signal from a decimal digit is one, then, the decimal digit adder performs a -6 operation. The final BCD products are generated with 2: 1 multiplexers.

According to [16] and [28], the critical path delay of the 2d-digit decimal adder is determined in the PPF block. Therefore, it is necessary to reduce the delay in the PPF carry network. The detailed analysis is presented in Section 4.4.

4 The Proposed Decimal Multiplier Using Hybrid BCD Codes

The improved 16×16-digit (compared with [15]) decimal multiplier and a new 34×34-digit decimal multiplier using hybrid BCD codes are presented in this Section. This design is referred to as hybrid, because it utilizes binary, BCD-8421, BCD-4221/5211, BCD-5421, ODDS, XS-3 and XS-6 codes [15]. The decimal multiplier is shown in Fig. 4.

The principle for the ODDS PPR tree is based on our previous paper [15]; it consists of a binary PPR tree block, a non-fixed size BCD-4221 counter block and a BCD-4221/5211 PPR tree block. A new 35:2 ODDS PPR tree for the 34×34-digit design is given in Section 4.3.

Also, a improved PPG circuit based on redundant ODDS codes and the XS-3 representations is proposed in this paper. The proposed decimal adder uses a conditional speculative decimal adder based on a KS parallel prefix/carry-select adder structure. As the carry-select adder block in [12-16] uses complex gates to generate carry signals, the PPF carry network in proposed design also uses this structure to balance the delay between the PPF carry network and the decimal carry-select adder block. The proposed modified method in the PPG stage and the BCD adder stage improves the performance of the decimal multiplier.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

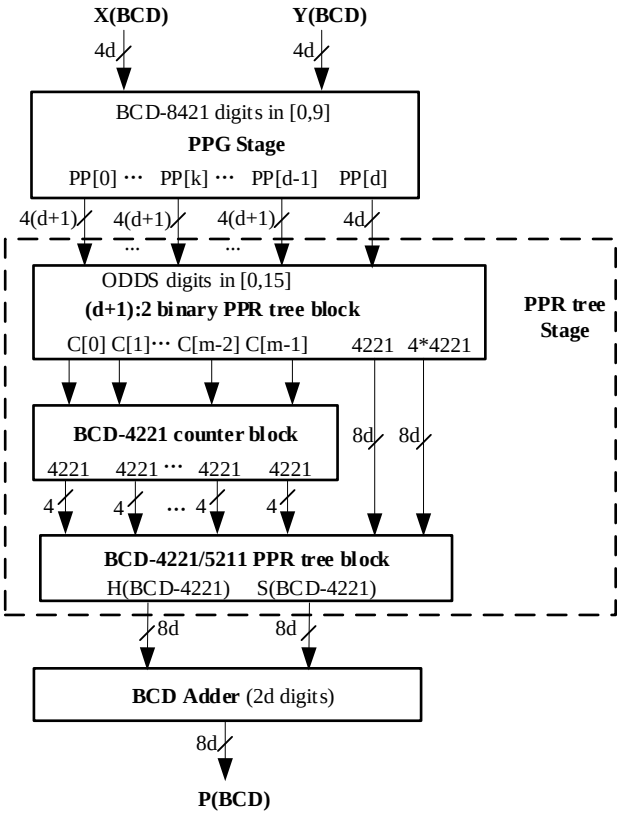

Fig. 4. The proposed decimal multiplier using hybrid BCD codes.

4.1 The Improved Decimal PPG Circuit Based on Radix-10 and XS-3 Recoding

As mentioned above, the partial product generation stage consists of the SD radix-10 recoder, the generation of the multiplicand multiples and the selection of multiples three blocks. The improved PPG circuit is used in both the SD radix-10 recoder block and the multiplicand multiples generation block.

The PPG circuit uses a SD radix-10 recoding to reduce the number of multiplicand multiples with $\{1X, 2X, 3X, 4X, 5X\}$. The $Y1_k$, $Y2_k$, $Y3_k$, $Y4_k$, $Y5_k$ in the proposed design are all implemented by 2:1 MUXs and the control signal in the 2:1 MUXs is Ys_{K-1}. As in Eqs. (9)-(13), the longest delay path ($Y5_k$ in Eq.(9)) consists of a 1-stage 2-input AND gate, 1-stage 3-input AND gate, 2-stage 2-input OR gate, and 1-stage 2-input XOR gates. As per [30], the normalized gate delay is 3.8.

As per Eqs. (8)-(13), the signals $Y1_k$, $Y2_k$, $Y3_k$, $Y4_k$, $Y5_k$ and Ys_k are modified as Eqs. (19)-(24). The longest delay path ($Y2_k$ in Eq.(23)) consists of a 1-stage 3-input OR-Inverter gate, a 1-stage 2-input XOR gate and a 1-stage 2:1 MUX (the normalized gate delay is 2.75 [30]). Therefore, the improved radix-10 encoder is faster.

$$Ys_k = \overline{\overline{y_k[3]} \cdot \overline{y_k[2]y_k[1]} \cdot \overline{y_k[2]y_k[0]}} \quad (19)$$

$$Y5_k = \overline{Ys_{k-1}} \cdot \overline{y_k[1] + \overline{y_k[2] \cdot y_k[0]}} +$$
$$Ys_{k-1}(y_k[2] \cdot \overline{y_k[1] + y_k[0]}) \quad (20)$$

$$Y4_k = \overline{Ys_{k-1}} \cdot (y_k[2]) \cdot \overline{y_k[0]}) +$$
$$Ys_{k-1} \cdot (y_k[0] \cdot (y_k[2] \oplus y_k[1]) \quad (21)$$

$$Y3_k = \overline{Ys_{k-1}}(y_k[1] \cdot y_k[0]) + Ys_{k-1}(y_k[1] \cdot \overline{y_k[0]}) \quad (22)$$

$$Y2_k = \overline{Ys_{k-1}} \cdot \overline{y_k[0] + \overline{y_k[3] + \overline{y_k[2]} \cdot y_k[1]}} +$$
$$Ys_{k-1} \cdot \overline{y_k[3] + \overline{y_k[0]} + y_k[2]) \oplus y_k[1]}) \quad (23)$$

$$Y1_k = \overline{Ys_{k-1}} \cdot y_k[0] \cdot \overline{y_k[2]) + y_k[1]}) +$$
$$Ys_{k-1} \cdot \overline{y_k[0]} \cdot \overline{y_k[2]) + y_k[1]}) \quad (24)$$

By considering Table 1, the multiplicand multiples ($\{1X, 2X, 3X, 4X, 5X\}$) are generated by the XS-3 recoding. In the digit recoding for the multiples NX+3, the longest path is in the (4X+3) recoding circuit. After the SD radix-10 recoder is modified, the longest path in the PPG circuit is in (4X+3). The XS-3 recoding table of 4X is shown in Table 2. As per Table 1 and Table 2, the proposed decimal digit D_i and the decimal carry T_i ($T_i[3:2] = 0$) for (4X+3) are given as follows:

$$T_i[0] = X_i[3] + \overline{X_i[2]}X_i[1] + X_i[2]\overline{X_i[1]}\,\overline{X_i[0]} \quad (25)$$

$$T_i[1] = X_i[2](X_i[1] + X_i[0]) + X_i[3] \quad (26)$$

$$D_i[0] = 1 \quad (27)$$

$$D_i[1] = \overline{X_i[3]}\,\overline{X_i[2]}\,\overline{X_i[1]} + X_i[2](X_i[1] + X_i[0]) \quad (28)$$

$$D_i[2] = X_i[3]\overline{X_i[0]} + X_i[2]X_i[1]\overline{X_i[0]} +$$
$$\overline{X_i[3]}\,\overline{X_i[2]}X_i[0] \quad (29)$$

$$D_i[3] = X_i[3]X_i[0] + X_i[2](\overline{X_i[1] \oplus X_i[0]}) \quad (30)$$

TABLE 2
THE XS-3 RECODING TABLE OF 4X

Xi	$X_i[3]$	$X_i[2]$	$X_i[1]$	$X_i[0]$	$T_i[1]$	$T_i[0]$	$X_i[3]$	$X_i[2]$	$X_i[1]$	$X_i[0]$
0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	0	0	1	1	1
2	0	0	1	0	0	1	0	0	0	1
3	0	0	1	1	0	1	0	1	0	1
4	0	1	0	0	0	1	1	0	0	1
5	0	1	0	1	1	0	0	0	1	1
6	0	1	1	0	1	0	0	1	1	1
7	0	1	1	1	1	0	1	0	1	1
8	1	0	0	0	1	1	0	1	0	1
9	1	0	0	1	1	1	1	0	0	1

As presented in Section 3.2, the *i*th decimal XS-3 digit recoding for the multiplicand multiples NX_i is $[NX_i] = D_i + T_{i-1}$. As $D_i[0]$ is always 1 (refer to Table 2), $(D_i + T_{i-1})$ in the (4X+3) recoding can be simplified as Eqs. (31)-(34) by the previous 4-bits carry propagate addition.

$$PP_i[0] = \overline{T_{i-1}[0]} \quad (31)$$

$$PP_i[1] = D_i[1] \oplus T_{i-1}[0] \oplus T_{i-1}[1] \quad (32)$$

$$PP_i[2] = D_i[2] \oplus (T_{i-1}[0]T_{i-1}[1] + D_i[1]T_{i-1}[0] +$$
$$D_i[1]T_{i-1}[1]) \quad (33)$$

$$PP_i[3] = D_i[3] \oplus \{(T_{i-1}[0]T_{i-1}[1] +$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

$$D_i[1]T_{i-1}[0] + D_i[1]T_{i-1}[1]) \cdot D_i[2]\} \quad (34)$$

4.2 The ODDS PPR Tree for a 16×16-digit Multiplier

The functions of the three blocks in the PP reduction tree are given by the following steps:

(1) A regular (d+1):2 binary CSA tree is used to compress the d+1 ODDS PP rows to two PP rows (*i.e.*, sum and carry). The carry and sum signals can be directly represented in BCD-4221 recoding (both 4×BCD-4221 and BCD-4221);

(2) A BCD-4221 counter block counts the digit carries generated between the digit columns in the binary CSA tree and a multiplication by six (× 6) is then performed (note that no extra hardware is needed by the × 6 operation).

(3) The BCD-4221 partial products in steps (1) and (2) are added by a decimal digit 3:2 compressor tree to obtain the final 2d-digit PPs (i.e., carry H and sum S). Every decimal digit H_i must be multiplied by 2 before adding H and S.

The maximum column height in the PP array of a 16×16-digit multiplier is d+1=17. The PPR tree in the maximum column height is given in Fig. 5. The input signals of the PPR tree are coded in an ODDS representation. A 17:2 PPR tree block consists of a 17:2 binary PPR tree, a BCD-4221 counter block and a 6:2 BCD-4221/5211 PPR tree. A 17:2 binary CSA tree has four binary compression levels (16:8, 9:6, 6:4 and 4:2 compression).

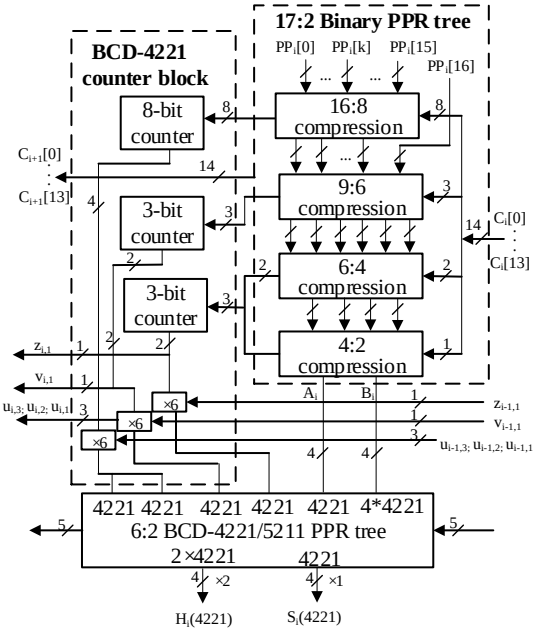

Fig. 5. The ODDS PPR tree in the maximum height columns for a 16×16-digit multiplier.

4.2.1 17:2 Binary PPR Tree

As shown in Fig. 5, the four compression levels are given in the 17:2 binary PPR tree. The numbers of input PP rows in the 1st, 2nd, 3rd and 4th compression levels are 17, 9, 6 and 4, respectively. The 9 PP rows are generated by the first 16:8 compression level (made of 4:2 compressors). The second 9:6 compression level is composed of 3:2 compressors and the 6 PP rows are generated. The third 6:4 compression level generates 4 PP rows. The final 2 PP rows are generated by the last 4:2 compression level.

The last 4:2 compression level in the ith position is shown in Fig. 6; it includes four binary 4:2 compressors. In the last compression level, the two PP rows with weights of 16, 8, 4, 2 for the carry output signal and weights of 8, 4, 2, 1 for the sum signal are generated. The carry and sum signals are represented with 4×BCD-4221 (weights of 16, 8, 8, 4) and BCD-4221 (weights of 4, 2, 2, 1).

The binary PPR tree is used for the decimal PPR stage; however, the +6 correction terms are required when every decimal carry occurs. As shown in Fig. 6, only one +6 carry correction term in cout2 is required during the last 4:2 compression level. The maximum number of carries transferred between adjacent digit columns of the binary 17:2 CSA tree is 14. These carries are labeled as $C_{i+1}[0{:}13]$ (carry output signals at the ith column) and $C_i[0{:}13]$ (carry input signals at the ith column). All numbers of digit carry transferred between adjacent digit columns of the binary CSA tree is represented by variable m (see fig.4).

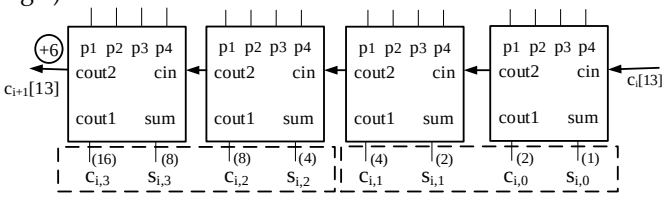

Fig. 6. The 4-bit binary 4:2 compressors in last a compression level.

The proposed binary 4:2 compressor used in the binary PPR tree is shown in Fig. 7. This compressor is modified from the scheme given in [31] so, the cout2 signal is generated by the 2-stage NAND gates (rather than the XOR gate and the MUXs) to reduce the delay [30].

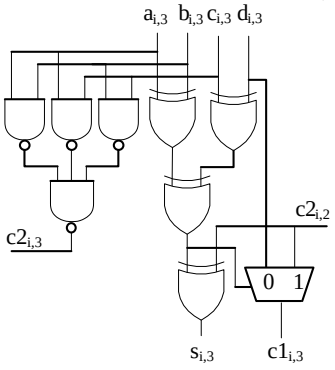

Fig. 7. The proposed binary 4:2 compressor (with weigh of 8).

4.2.2 BCD-4221 Counter Correction and 6:2 BCD-4221/5211 PPR Tree

Fig. 8 shows the architecture for a non-fixed size BCD-4221 counter block and a BCD-4221 PPR tree block in the maximum height column (that determines the critical path delay of a 17:2 ODDS PPR tree). The 14-bit digit carry output signals (*i.e.*,$C_i[0{:}13]$) generate 4 BCD-4221 decimal correction terms. As shown in Fig. 4, Fig. 5 and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

Fig. 8, the 8-bit, 3-bit and 3-bit BCD-4221 decimal counters are used to count the 14-bit digit output carries for the following reasons:

(1) The BCD-4221 counter is more regular than a binary ODDS counter, that consists of only 3:2 CSAs or half adders (HA);

(2) The non-fixed size BCD-4221 counter splits the 14-bit counter into the 8-bit and the two 3-bit counters which can balance the paths in the PPR tree and reduce the critical path delay in the decimal 6:2 PPR tree, where first generated PPs are compressed firstly (for the correction terms and the results of the binary compression).

(3) 3-bit counters are used to generate a BCD-4221 decimal correction digit by using only a 3:2 compressor. 8-bit counters are used to generate two BCD-4221 decimal correction digits by one HA and four full adders (3:2 CSA).

As shown in Fig. 9, an 8-bit 4221 counter generates two 4-bit BCD-4221 digits U_i ($U_i \epsilon [0,9]$). The input signals in the counter are generated by the first 16:8 binary compression level. The U_i and $6 \times U_i$ in the ith column are given by:

$$U_i = 4 \times u_{i,3} + 2 \times u_{i,2} + 2 \times u_{i,1} + 1 \times u_{i,0} \quad (35)$$

$$6 \times U_i = 20 \times u_{i,3} + 10 \times u_{i,2} + 10 \times u_{i,1} + 4 \times u_{i,3} + 2 \times u_{i,2} + 2 \times u_{i,1} + 4 \times u_{i,0} + 2 \times u_{i,0} \quad (36)$$

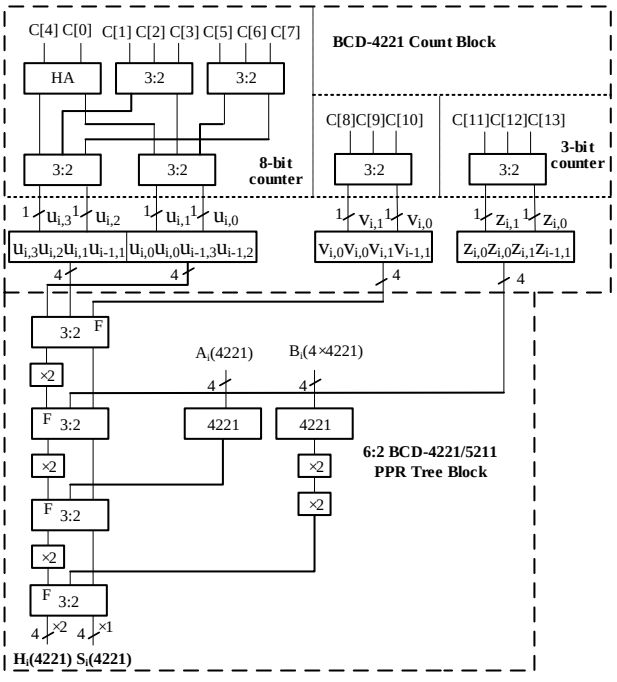

Fig. 8. The correction block in the maximum height column (F denotes the fast input in a 3:2 CSA).

As per Eq. (36), the digits with weight of 10 and 20 in $6 \times U_i$ are provided to the $(i+1)$th column with weight of 1 and 2; the digits with weight of 10 and 20 in $6 \times U_{i-1}$ are provided to the ith digit column with weight of 1 and 2 too. As shown in Fig. 8, the 8 carry output signals generate two BCD-4221 correction terms. A full adder is used for a 3-bit counter to generate a 2-bit digit V_i (*i.e.*, the sum bit $v_{i,1}$ and the carry bit $v_{i,0}$) given by:

$$V_i = 2 \times v_{i,1} + v_{i,0} \quad (37)$$

$$6 \times V_i = 10 \times v_{i,1} + 4 \times v_{i,0} + 2 \times v_{i,0} + 2 \times v_{i,1} \quad (38)$$

As per Eq.(38), the digit with weight of 10 in $6 \times U_{i-1}$ is provided to the ith column with weight of 1. Therefore, 3 carry output signals generate one BCD-4221 correction term; similarly, another 3-bit counter (with inputs $C_i[11:13]$ and outputs $z_{i,1}$ and $z_{i,0}$) also generates one BCD-4221 correction term. The 14-bit digit carry output signals generate four BCD-4221 correction terms. The counter block and the binary 17:2 reduction tree blocks generate five BCD-4221 terms and one 4×BCD-4221 for the maximum height columns. These six decimal digits are transferred to the 6:2 BCD-4221/5211 PPR tree block to generate the final two PP rows. As shown in Fig.8 for the 6:2 BCD-4221/5211 PPR tree block, the slowest outputs are connected to the fast inputs of the next 3:2 CSA to balance the critical path delay.

4.3 New ODDS PPR Tree for the 34×34-digit Multiplier

A new 34×34-digit decimal multiplier is proposed in this section. The improved decimal PPG circuit based on radix-10 recoding and XS-3 recoding is used in the design and 35 ODDS PP rows are generated. A binary 35:2 PPR tree block, a non-fixed size BCD-4221 counter block and a BCD-4221 PPR tree block are used in the 35:2 ODDS PPR tree stage for the 34×34-digit multiplier. The maximum column height in the partial product array of a 34×34-digit multiplier is 35. Finally, the 68-digit improved decimal adder is used in last stage.

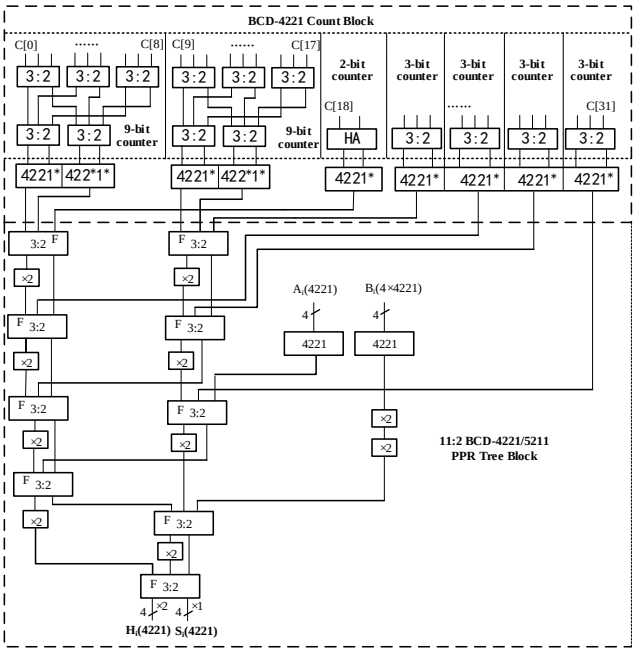

Fig. 9. The correction block in the maximum height column (F denotes the fast input in a 3:2 CSA and * denotes the carry signals from the (i-1)th digit column).

A binary 35:2 CSA tree requires compression using 5-levels (4:2, 3:2, 3:2, 4:2 and 4:2); the numbers of PP rows in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

the compression level are 35, 18, 12, 8, 4 and 2, respectively, they are generated as follows:

- The 18 PP rows are generated after the 4:2 compression in the first 35:18 compression level. The 17 digit carries are generated in the maximum height column.
- The 12 PP rows are generated after the 3:2 compression in the second 18:12 compression level. The 6 digit carries are generated in the maximum height column.
- The 8 PP rows are generated by the third 12:8 compression level, which is made of 3:2 compressors. The 4 digit carries are generated in the maximum height column.
- The 4 PP rows are generated by the third 8:4 compression level, which is made of 4:2 compressors. The 4 digit carries are generated in the maximum height column.
- The final 2 PP rows are generated by the last level of the 4:2 compression. The 1 digit carries are generated in the maximum height column.

Fig. 9 shows the implementation of the BCD-4221 count block and the 11:2 BCD-4221/5211 PPR block for the maximum height column. The input signals in the binary 35:2 CSA tree are in an ODDS representation; the output signals are in a BCD-4221 representation for both 4×4221 and 4221. The maximum number of digit carries transferred between adjacent columns of the binary 35:2 CSA tree is 32; these carries are labeled as $C_{i+1}[0]$, $C_{i+1}[1] \cdots C_{i+1}[31]$(output carries) and $C_i[0]$, $C_i[1] \cdots C_i[31]$ (input carries).

The two 9-bit, one 2-bit and four 3-bit BCD-4221 decimal counters are used to count the 32 digit carries and generate nine BCD-4221 decimal correction terms. The non-fixed size BCD-4221 counter block is used to balance the paths in the PPR tree and reduce the critical path delay in the decimal 11:2 PPR tree; the 9-bit 4221 counter (Fig. 9) produces the 4-bit 4221 decimal digit W_i. The computation of $6 \times W_i$ is given by Eq. (36). The 9 carries generate two 4221 correction terms; the 2-bit counter (Fig. 9) produces the 2-bit digit L_i. It uses half adder to generate s_i and c_i.

$$L_i = 2 \times c_i + s_i \tag{39}$$

$$6 \times L_i = 10 \times c_i + 4 \times s_i + 2 \times s_i + 2 \times c_i \tag{40}$$

Therefore, the two output carries generate one 4221 correction term. The 2-bit counter is used in to generate a fast correction term (compression with the correction term obtained in the first 35:18 binary compression level). As shown in Fig.9, the 32 digit carries are counted and generate nine 4221-BCD correction terms. The counter block and the 35:2 binary PPR tree blocks generate ten BCD-4221 PPs and one 4×4221 PP for maximum height column. These eleven decimal PP rows are used by an 11:2 BCD-4221 PPR block to generate the final two PP rows.

4.4 Improved Decimal Adder

In the compression stage, the d+1 PP rows are reduced to the two $2d$-digit operands S and H coded in BCD-4221. Prior to the binary addition, the BCD-4221 to BCD-8421 conversion and the conditional +6 blocks are required. S is recoded from BCD-4221 to BCD-8421 and BCD excess-6 code (that is S+6). S and S+6 are transferred to a 2:1 MUX that is controlled by the +6 condition signal r_i; the result is given by B*. At the same time, the $2 \times H$ is performed by a BCD-4221 to BCD-5421 digit conversion and a 1-bit wired left shift to obtain the operand $2 \times H$ coded in BCD-8421 [8]. The conditional speculative decimal adder based on a binary KS hybrid parallel prefix/carry-select structure is used in the proposed decimal adder.

The delay in the critical path for a 4-bit decimal carry-select adder consists of a 1-stage OR gate, two 1-stage AND-OR-Inverter gates, and a 1-stage XOR gate according to [16]. However, a 128 bit binary adder for a 16×16-digit multiplier consists of a 1-stage OR gate, 7 stages of AND-OR gates; a 272 bit binary adder for a 34×34-digit multiplier consists of a 1-stage OR gate, 9 stages of AND-OR gate [28]. So, the critical path delay occurs in the PPF block.

The proposed 128-bit and 272-bit decimal adders use the parallel prefix approach based on a KS structure. The structure for the 2d digit BCD adder is given in Fig. 10. The carry generation signal g and the carry propagate signal p are replaced by the corresponding inverse functions $\overline{g}$ and $\overline{p}$. The OR-AND Inverter gate and the OR Inverter gate are used to replace the prefix operator (*i.e.*, block carry generation signal and the block carry propagate signal) in the odd PPF levels. The AND-OR Inverter gate and the AND Inverter gate are used to replace the prefix operator in the even PPF levels. The new design of the conditional speculative decimal adder can reduce the delay in PPF level and then obtain improvements in performance of a decimal adder.

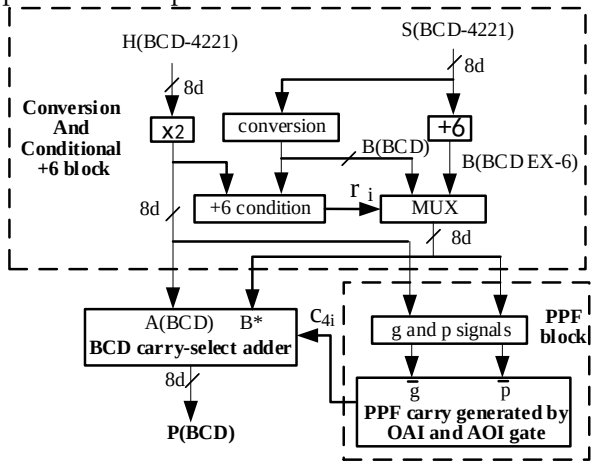

Fig. 10. 2d digit BCD adder.

5 Evaluation and Comparison

The 16×16-digit and 34×34-digit multipliers are evaluated in this section using the decimal PPR tree based on the hybrid BCD codes, the improved PPG circuit based on redundant BCD coding and the improved parallel prefix/carry-select decimal adder; the results are compared with both a non-redundant decimal multiplier [14] and the redundant decimal multipliers [8], [15] that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2706262, IEEE Transactions on Computers

represent some of the best designs found in the technical literature. Furthermore, since a 34×34-digit multiplier is not given in [14] and [15], it is implemented in this work by following the design method of [14] and [15].

The decimal multiplier designs in [14-15] and the proposed design are described at the gate level in Verilog HDL and verified by Synopsys VCS using randomly generated input patterns. The designs are synthesized by using the Synopsys Design Compiler and the NanGate 45nm Open Cell Library. In the simulation of each design, a supply voltage of 1.25V and room temperature are assumed. Three standard inverters of 4X strength are used for the output load and a standard NAND of 4X strength is used for the input drivers. The option for logic structuring is turned off to prevent the tool from changing the structure of the unit cells.

A binary PPR tree block, a non-fixed size BCD-4221 counter block and a BCD-4221/5211 PPR tree block are employed in the proposed decimal PPR tree for the 16×16-digit and the 34×34-digit multipliers. A improved PPG circuit and a improved decimal adder are used for high-speed parallel decimal multiplication to improve the design of [15]. To reduce the delay in the decimal adder, efficient AND-OR-Inverter and OR-AND-Inverter gates are used in the PPF block.

As per the design method in [14], the radix-10 PPG circuit, the 35:2 PPR tree made of the 17:2 and the 18:2 CSAs in [14] and a conditional speculative decimal adder based on a QT tree are used in the 34×34-digit decimal multiplier design for [14]. The synthesis results based on the Synopsis Design Complier for the proposed design and the previous designs is given Table 3.

TABLE 3
DESIGNS OF 16×16-DIGIT AND 34×34-DIGIT PARALLEL DECIMAL MULTIPLIERS (USING NANGATE 45NM OPEN CELL LIBRARY)

	Scheme	Delay (ns)	Ratio	Area (mm^2)	Ratio
16×16-digit	NRDM10 [14]	3.86	1.203	0.0479	1.120
	RDM16 [15]	3.39	1.056	0.0430	1.018
	Improved	3.21	1.000	0.0422	1.000
34×34-digit	NRDM10 [14]*	4.73	1.258	0.1732	1.095
	RDM16[15] **	4.01	1.067	0.1597	1.008
	Proposed	3.76	1.000	0.1588	1.000

* The data is obtained from a design implemented according to the design method in [14].

** The data is obtained from a design implemented according to the PPG and decimal adder design method in [15] and new 35:2 ODDS PPR tree in this paper.

As the detailed description of some modules of the PPR circuit are not provided in [8] (only the dot-diagrams are shown), then accurate synthesis data is not given in this work. As per [8], its redundant radix-10 16×16-digit decimal multiplier reduces the delay by 10.75% and the area by 11.07% compared with NRDM10 [14]. Table 3 summarizes the delay and area of NRDM10, RDM16 [15] and the proposed 16×16-digit decimal multiplier (with hybrid BCD codes, a modified PPG circuit and a decimal adder). The proposed design reduces the delay and area by 16.83% and 11.90%, respectively, compared with NRDM10; the proposed design reduces the delay and area by 5.31% and 1.81%, respectively, compared with our previous design RDM16.

For the 34×34-digit decimal multiplier design, the redundant radix-10 design of [8] reduces the delay by 10.75% and the area by 11.1% compared with NRDM10. The proposed design reduces the delay and area by 20.51% and 8.31% compared with NRDM10.The proposed design reduces the delay by 6.23% compared with RDM16 and occupies less area than our previous design in [15].

6 CONCLUSION

A parallel decimal multiplier based on hybrid BCD codes, a modified PPG circuit and a decimal adder are proposed in this paper for improved performance. In the hybrid BCD coded PPR tree, a BCD-4221 counter block is used to count the digit carries generated between the digit columns in a binary CSA tree. A decimal PPR tree based on BCD-4221 decimal digit 3:2 compressors is used to add the result of the 6× carries count and the result of the binary CSA PPR tree to obtain the final two PP rows. Analysis and comparison using the Synopsys Design Compiler with NanGate 45nm Open Cell Library have confirmed that the proposed decimal multiplier is faster and smaller than previous state-of-the-art designs. The proposed high-speed decimal multiplier can be used for both high performance fixed-point and floating-point decimal multiplications.

ACKNOWLEDGMENT

This work is supported by grants from National Natural Science Foundation of China (No. 61401197), Natural Science Foundation of Jiangsu Province (No. BK20151477) and Foundation of Graduate Innovation Center in NUAA (KFJJ20160407). Part of this work is supported by the ITC Endowment at Northeastern University.

REFERENCES

[1] M. F. Cowlishaw, "Decimal Floating-Point: Algorism for Computers," in *Proc. 16th IEEE Symp. Computer Arithmetic*, pp. 104-111, Jul. 2003.

[2] M. A. Erle, B. J. Hickmann and M. J. Schulte, "Decimal Floating-Point Multiplication," *IEEE Trans. Computers*, vol. 58, pp. 902-916, July. 2009.

[3] M. F. Cowlishaw, E. M. Schwarz, R. M. Smith, and C. F. Webb, "A Decimal Floating-Point Specification," in *Proc. 15th IEEE Symp. Computer Arithmetic*, pp. 147-154, June. 2001.

[4] IEEE Std 754(TM)-2008, *IEEE Standard for Floating-Point Arithmetic*, IEEE, Aug. 2008.

[5] L. Dadda, "Multioperand Parallel Decimal Adder: A Mixed Binary and BCD Approach," *IEEE Trans. Computers*, vol. 56, pp. 1320-1328, Oct. 2007.

[6] M. A. Erle and M. J. Schulte, "Decimal Multiplication Via Carry-Save Addition," in *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures, and Processors*, pp. 348-358, Jun. 2003.

[7] M. A. Erle, E. M. Schwarz and M. J. Schulte, "Decimal Multiplication with Efficient Partial Product Generation," in *Proc. 17th IEEE Symp. Computer Arithmetic*, pp. 21-28, Jun. 2005.

[8] A. Vazquez, E. Antelo, and J. Bruguera, "Fast Radix-10 Multiplication Using Redundant BCD codes," *IEEE Trans. Computers*, vol. 63, pp. 1902-1914, Aug. 2014.

[9] T. Lang and A. Nannarelli, "A Radix-10 Combinational Multiplier," in *Proc. 40th Asilomar Conference on Signals, Systems, and Computers*, pp. 313-317, Oct. 2006.

[10] L. Dadda and A. Nannarelli, "A Variant of a Radix-10 Combinational Multiplier," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pp. 3370-3373, May. 2008.

[11] S. Gorgin and G. Jaberipur, "A Fully Redundant Decimal Adder and Its Application in Parallel Decimal Multipliers," *Microelectronics Journal*, vol. 40, no. 10, pp. 1471-1481, Oct. 2009.

[12] G. Jaberipur, and A. Kaivani, "Improving the Speed of Parallel Decimal Multiplication," *IEEE Trans. Computers*, vol. 58, pp. 1539-1552, Nov. 2009.

[13] A. Vazquez, E. Antelo and P. Montuschi, "A New Family of High-Performance Parallel Decimal Multipliers," in *Proc. 18th IEEE Symp. Computer Arithmetic*, pp. 195-204, Jun. 2007.

[14] A. Vazquez, E. Antelo and P. Montuschi, "Improved Design of High-Performance Parallel Decimal Multipliers," *IEEE Trans. Computers*, vol. 59, pp. 679-693, May 2010.

[15] X. Cui, W. Liu, W. Dong and F. Lombardi, "A Parallel Decimal Multiplier Using Hybrid Binary Coded Decimal (BCD) Codes," in *Proc. 23rd IEEE Symp. Computer Arithmetic*, pp. 150-155, Jul. 2016.

[16] A. Vazquez and E. Antelo, "Conditional Speculative Decimal Addition," in *Proc. Seventh Conf. Real Numbers and Computers (RNC 7)*, pp. 47-57, Jul. 2006.

[17] F. Busaba, T. Slegel, S. Carlough, C. Krygowski, and J. Rell, "The Design of the Fixed Point Unit for the z990 Microprocessor," in *Proc. 14th ACM Great Lakes Symp. VLSI 2004*, pp. 364-367, Apr. 2004.

[18] S. Carlough, S. Mueller, A. Collura, and M. Kroener, "The IBM zEnterprise-196 Decimal Floating Point Accelerator," in *Proc. 20th IEEE Symp. Computer Arithmetic*, pp. 139–146, Jul. 2011.

[19] B. Shirazi, D. Y. Y. Yun, and C. N. Zhang, "RBCD: Redundant Binary Coded Decimal Adder," *IEE Proceedings-Computers and Digital Techniques*, vol. 136, pp. 156-160, Mar. 1989.

[20] A. Svoboda, "Decimal Adder with Signed Digit Arithmetic," *IEEE Trans. Computers*, vol. C-18, pp. 212-215, Mar. 1969.

[21] R. D. Kenney, M. J. Schulte, and M. A. Erle, High-Frequency Decimal Multiplier, *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 26–29, Oct. 2004.

[22] E. L. Braun, *Digital Computer Design: Logic, Circuitry, and Synthesis*, Academic Press, 1963.

[23] R. K. Richards, *Arithmetic Operations in Digital Computers*, New Jersey: D. Van Nostrand Company, Inc., 1955.

[24] R. D. Kenney and M. J. Schulte, "High-Speed Multioperand Decimal Adders," *IEEE Trans. Computers*, vol. 54, pp. 953-963, Aug. 2005.

[25] A. Vazquez and E. Antelo, "Multi-Operand Decimal Addition by Efficient Reuse of a Binary Carry-Save Adder Tree," in *Proc. 44th Asilomar Conference on Signals, Systems and Computers*, pp. 1685-1689, Nov. 2010.

[26] I. D. Castellanos and J. E. Stine. "Compressor Trees for Decimal Partial Product Reduction," in *Proc. 18th ACM Great Lakes Symp. VLSI*, pp. 107-110, Jan. 2008.

[27] S. K. Mathew, M. Anders, R. K. Krishnamurthy, and S. Borkar, "A 4Ghz 130 nm Address Generation Unit with 32-Bit Sparse-Tree Adder Core," *IEEE J. Solid-State Circuits*, vol. 38, pp. 689-695, May 2003.

[28] G. Dimitrakopoulos and D. Nikolos, "High-Speed Parallel-Prefix VLSI Ling Adders," *IEEE Trans. Computers*, vol. 54, pp. 225-231, Aug. 2005.

[29] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Trans. Computers*, Vol. C-22, pp. 786-793, Aug. 1973.

[30] W. Yeh and C. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. Computers*, vol. 49, pp. 692-701,2000.

[31] D. Radhakrishnan and A. Preethy, "Low Power CMOS Pass Logic 4-2 Compressor for High-Speed Multiplication," in *Proc. IEEE Midwest Symp. Circuits Syst.*, vol. 3, pp. 1296-1298, 2000.

Xiaoping Cui received the B.Sc. degree in Radio Technology from Anhui University, Hefei, China, in 1983 and the M. Eng. degree in Information Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China in 1992. She was an engineer in Nanjing Panda Electronics Company Limited during 1992-1998. She has been with NUAA Since 1998, where she is now an Associate Professor. Her research interests include computer arithmetic, digital system design and computer architecture.

Wenwen Dong received the B.Sc. degree in Microelectronics from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2014 and M. Eng. dgree in Integrated Circuit Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China in 2017. She is currently with Yangtze Memory Technologies Co. Ltd., Shanghai, China. Her research interest includes computer arithmetic and embedded system.

Weiqiang Liu 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 Queen's 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 was a Research Fellow in the Institute of Electronics, Communications and Information Technology (ECIT) at QUB from Aug. 2012 to Nov. 2013. He has published one research book by Artech House and over 40 leading journal and conference papers. His paper was finalist in the Best Paper Contest of IEEE ISCAS 2011 and he is the co-author of a Best Paper Candidate of ACM GLSVLSI 2015. He serves as an Associate Editor of IEEE Transactions on Computers (TC), the leader of The Multimedia Team at TC Editorial Board, and the Guest Editors of two special issues of IEEE Transactions on Emerging Topics in Computing. He has been a technical program committee member for several international conferences including ARITH, ASAP, ISCAS, and ICONIP. He is a IEEE Circuits & Systems for Communications (CASCOM) Technical Committee member. His research interests include computer arithmetic, emerging technologies in computing systems, and cryptographic hardware.

Earl E. Swartzlander Jr. 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 86 refereed journal papers, 41 book chapters, and 310 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 (2014-2016), the IEEE Awards Board (2015-2016), and the IEEE Awards Policy and Portfolio Review Committee (2017). He has chaired a number of conferences. He is a life fellow of the IEEE and 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 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. During 2007-2010 Dr. Lombardi was the Editor-In-Chief of the IEEE Transactions on Computers. Currently, he is the Editor-in-Chief of the IEEE Transactions on Nanotechnology and the inaugural Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computing. He currently serves as an elected Member of the Board of Governors 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.