A Parallel Decimal Multiplier Using Hybrid Binary Coded Decimal (BCD) Codes

Xiaoping Cui, Weiqiang Liu and Dong Wenwen
College of Electronic and Information Engineering
Nanjing University of Aeronautics and Astronautics
Nanjing, 211106, China
Email: {wnhcxp, liuweiqiang}@nuaa.edu.cn

Fabrizio Lombardi
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, US
Email: lombardi@ece.neu.edu

***Abstract*—A parallel decimal multiplier is proposed in this paper to improve performance by mainly 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 BCD-4221/5211 code; hence this design is referred to as hybrid. The signed-digit radix-10 recoding with the digit set {-5, 5} and the redundant BCD excess-3 (XS-3) representations are used for partial product (PP) generation. In this paper, a new decimal partial product reduction (PPR) tree is proposed; it consists of a binary PPR tree block, a nonfixed size BCD-4221 counter correction block and a BCD-4221/5211 decimal PPR tree block. Analysis and comparison using the logical effort model and 45 nm technology show that the proposed decimal multiplier is faster compared with previous designs found in the technical literature.**

Keywords—Parallel Decimal Multiplication, Overloaded BCD Representation, Redundant Excess-3 Code, Redundant Arithmetic.

I. INTRODUCTION

Decimal arithmetic is in demand for many commercial applications; for example in financial computation, binary arithmetic can introduce conversion and rounding errors [1]. Furthermore, the decimal specification has been added to the revised IEEE 754-2008 standard [2-3], thus attracting a significant research interest in decimal arithmetic architectures and implementations [4-7]. Decimal adders and multipliers are one of the most important primitives for decimal arithmetic and recently they have been studied quite extensively.

The first parallel decimal multiplier is proposed in [8]. The most representative high-performance BCD multipliers are presented in [7-13]. Non-redundant and redundant decimal formats are used in a decimal multiplier design; two different sign-digit (SD) encodings (radix-5 and radix-10) are proposed for fast partial product (PP) generation. The SD radix-10 recoding is used to recode the signed digits in the digit set $\{-5, 5\}$. (d+1) PP rows are generated in the radix-10 recoding and 2d PP rows are generated in the radix-5 recoding. Double-BCD unsigned partial products are proposed in [8-11]. The number of PPs is equivalent to the SD radix-5 scheme; furthermore, this scheme has the disadvantage of a large area overhead compared with the radix-10 recoding [6-8].

The decimal carry-save algorithm based on BCD-4221 and BCD-5211 codes are used in [12-13] for PP reduction (PPR). The PPR tree based on BCD-4221 and BCD-5211 codes is very regular; it includes 4-bit binary carry-save adder (CSA) and a decimal $\times$ 2 operation. These 4-bit decimal codes have been used to significantly increase the performance of decimal multi-operand addition and multiplication [12-15]. The BCD-4221 and BCD-5211 (BCD-4221/5221) are self-complementing [16]; therefore, the 9's complement of a 4-bit digit can be easily obtained by bit-inversion. The disadvantage of BCD-4221/5211 codes is that decimal multiples (such as 3X) cannot be obtained in a carry-free manner by using non-redundant radix-10 digit [7].

Redundant decimal formats have been used to speed up BCD multiplication. Decimal carry-free addition is achieved through SD representations using a redundant digit set $\{-a, \cdots, 0, \cdots a\}$, where $5 \leq a \leq 9$ [6-7], [17-18]. The overloaded decimal digit set (ODDS) code is a type of redundant decimal number representation {0, 15}; it was proposed for high-speed multi-operand decimal adders and multipliers [7], [10], [19]. The ODDS representation can generate both simple and complex decimal multiples in a carry-free manner. The PPR can be performed with binary arithmetic by using ODDS code; there is no need to correct invalid six 4-bit digits compared with BCD-8421 code [7]. Therefore, the ODDS code is usually used for implementing fast and efficient decimal arithmetic circuits.

Both the ODDS and redundant BCD excess-3 (XS-3) representations are used for improving the performance of parallel decimal multipliers [7]. Due to its self-complementing property, the XS-3 code [-3, 12] is used in [7] for PPG. XS-3 PPs can also be represented as ODDS PPs by adding pre-computed correction term; the ODDS PPs [0, 15] are added by a PPR tree until two decimal PP rows (coded as BCD excess-6 and BCD-8421) remain. The PPR tree in [7] consists of a binary CSA tree, a binary sum correction block and a decimal 3:2 compressor block. The final BCD excess-6 and BCD-8421 PP rows are added by a carry propagate addition. An advantage of the XS-3 representation over BCD-4221/5211 representation is that all relevant multiples (such as 1X, 2X, 3X, 4X, and 5X) can be implemented in a carry-free way [7].

In this paper, a decimal multiplier based on hybrid BCD codes is proposed; it uses the same coding method as used in [7] for PPG. However, the PP reduction tree is different; the proposed PPR tree consists of a binary PPR tree block, a nonfixed size BCD-4221 counter correction block and a BCD-

4221/5211 decimal PPR tree block. During the PPR stage, ODDS PPs are compressed to two PP rows by a binary CSA tree. The two ODDS PP rows can be converted directly to BCD-4221 (including both 4×BCD-4221 and BCD-4221); the BCD-4221 sum correction block is used to count the carries generated between two decimal digit columns. All BCD-4221 PP rows are then compressed to two BCD-4221 PP rows by a decimal PPR tree. Finally, two BCD-4221 PP rows are converted into BCD-8421 PP rows and accumulated by a final decimal carry propagate addition. This design is referred to as hybrid, because as outlined above, it utilizes multiple BCD codes.

The proposed decimal multiplier is different from [7] in the following two aspects: (1) Nonfixed size BCD-4221 counters are used in the sum correction block to balance the paths in the PPR tree and reduce the critical path delay. (2) Decimal 3:2 compressors based on 4-bit binary CSAs and decimal digit doublers are used in the decimal PPR tree block.

The paper is organized as follows. Section II reviews the BCD and redundant BCD representations that are used in this work for decimal partial product generation and reduction. The proposed partial product reduction tree for 64-bit (16×16-digit) decimal multiplier is described in Section III. In Section IV, complexity and delay are evaluated and compared with other representative decimal implementations found in the technical literature. Finally, conclusion is provided in Section V.

II. Review of BCD Representations and Decimal Multiplication

Digital multiplication mainly consists of three stages: the generation of partial products, the fast reduction (addition) of partial products and the final carry propagate addition. Both non-redundant and redundant BCD digits are usually used in high-performance decimal multipliers [7], [11-12], [13].

A. *SD Radix-10 Recoding and Decimal Multiplier based on BCD-4221/5221*

A d-digit decimal integer operand can be expressed as follows:

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

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

where $Z_i \epsilon [0,9]$ is the i^{th} decimal digit, $z_{i,j}$ is the j^{th} bit of the i^{th} digit, and $r_j \geq 1$ is the weight of the j^{th} bit. The most common used 4-bit decimal weighted codes include BCD-8421, BCD-4221, and BCD-5211.

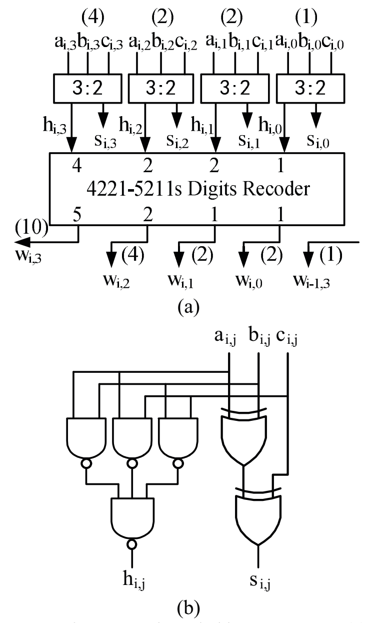
The multiplicand $X (X = X_{d-1}X_{d-2} \dots X_1X_0)$ and multiplier $Y (Y = Y_{d-1}Y_{d-2} \dots Y_1Y_0)$ are assumed to be unsigned BCD decimal integer of d digits each. The product $P = X \times Y$ is in a non redundant BCD format with *2d* digits. The decimal radix-10 recoding is used in [7] [13-14] for a high performance parallel decimal multiplier; it leads to a simple generation of the partial products. The decimal carry-save algorithm based on BCD-4221/5211 is used for partial product reduction [12-13].

The SD radix-10 recoding transforms a BCD 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 signal ys_{i-1} ($ys_{i-1} = 1$, if $Y_{i-1} \geq 5$). 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 MUXs to select the positive multiplicand multiples $\{X, 2X, 3X, 4X, 5X\}$. Due to the redundancy of the BCD-4221 decimal codes, when $ys_i = 1$, a negative multiplicand multiple $\{-X, -2X, -3X, -4X, -5X\}$ is selected by bit inversion of the positive BCD-4221 coded multiple. The bit inversion operand is accomplished by using the XOR gates controlled by $\mathrm{ys_i}$. 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 [7], [12-13].

Each column of 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 CSA uses a conventional 4-bit binary 3:2 CSA; this can be expressed as [13]:

$$A_i + B_i + C_i = \sum_{j=0}^{3} (a_{i,j} + b_{i,j} + c_{i,j}) r_j$$
$$= \sum_{j=0}^{3} s_{i,j} r_j + 2 \times \sum_{j=0}^{3} h_{i,j} r_j = S_i + 2 \times H_i \qquad (3)$$

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). $S_i \epsilon [0,9]$ and $H_i \epsilon [0,9]$ in Eq. (3) are the decimal sum and carry digits at position *i*. Fig. 1 shows an example of a decimal 3:2 CSA for operands coded in BCD-4221. Every decimal digit $H_i \epsilon [0,9]$ ($h_{i,3}$, $h_{i,2}$, $h_{i,1}$, $h_{i,0}$) is required for multiplication by 2 (*i.e.*, $\times 2$). The double of H ($2 \times H_i$) can be performed 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. Refer to [13] for the definition of BCD-4221s/5211s and the difference between BCD-4221s/5221s and BCD-4221/5221.


(4)
(2)
(2)
(1)
$a_{i,3}b_{i,3}c_{i,3}$ $a_{i,2}b_{i,2}c_{i,2}$ $a_{i,1}b_{i,1}c_{i,1}$ $a_{i,0}b_{i,0}c_{i,0}$
3:2
$h_{i,3}$ $s_{i,3}$ $h_{i,2}$ $s_{i,2}$ $h_{i,1}$ $s_{i,1}$ $h_{i,0}$ $s_{i,0}$
4 2 2 1
4221-5211s Digits Recoder
5 2 1 1
(10) $w_{i,3}$
(4) $w_{i,2}$
(2) $w_{i,1}$
(2) $w_{i,0}$
(1) $w_{i-1,3}$
(a)
$a_{i,j}$ $b_{i,j}$ $c_{i,j}$
$h_{i,j}$
$s_{i,j}$
(b)
Fig. 1 Decimal 3:2 CSA for operands coded in BCD-4221: (a) Decimal digit

3:2 CSA, and (b) Full adder (after [13]).

The bits of W_i are expressed by $w_{i,j}$ in which the most significant output bit $w_{i,3}$ represents the decimal carry out (with a weight of 10) to the next digit position, while $w_{i-1,3}$ is a decimal carry input from the adjacent low decimal digit. Thus, the digits of $2H$ are given by:

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

The decimal digit (4-bit) 3:2 CSA is made of a 4-bit full adder and a BCD-4221 to BCD-5211s converter. The decimal digit 3:2 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.

The adder setup is used to generate the double of H and convert the BCD-4221 PPs to the BCD-8421 and BCD excess 6 representations. The final two PPs are accumulated by a decimal carry propagate addition; the final BCD adder is effectively a 2*d*-digit hybrid parallel prefix/carry-select adder [12-13].

B. Redundant BCD Representations and Decimal Partial Product Generation and Reduction

Both BCD-4221 and BCD-5211 decimal encodings are used for partial product reduction [12-13]. Both codes are self-complementing in PPG; therefore, the 9's complement of a digit (as required for negation) can be simply obtained by bit-inversion. Hence, they are also used to improve the performance of decimal multi-operand operations, such as addition and multiplication. As mentioned previously, a disadvantage of the BCD-4221 and BCD-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 decimal carry-free addition [6-7], [17-18].

The redundant BCD representation is given by:

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

where d is the number of decimal digits, s_z is the sign bit, and $Z_i \in [l-e, m-e]$ is the i^{th} 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 $\rho = m - l + 1 - 10$ [10].

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: $Z_i \in [-3,12], e = 3, l = 0, m = 15,\ \rho = 6$.

The SD radix-10 decimal multiplier using redundant BCD codes [7] is shown in Fig. 2. The SD radix-10 recoding and XS-3 code is used for fast PPG, in which positive multiplicand multiples $\{0X, 1X, 2X, 3X, 4X, 5X\}$ are pre-computed in a carry-free manner; negative multiples can be obtained by bit inversion of the corresponding positive ones. XS-3 PPs can be represented as ODDS PPs by adding pre-computed correction term (*i.e.,* adding -3 constants). The (*d*+1) PP rows based on ODDS representation can be compressed until two 2*d*-digit decimal PPs by using (*d*+1):2 PPR tree. These two PP rows (coded in both BCD XS-6 and BCD-8421) are accumulated by a final decimal carry propagate addition.

The (*d*+1):2 PPR tree in Fig. 2 consists of three parts:

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

(2) At the same time, a decimal sum correction block counts the carries generated by the binary CSA tree. The number of carries per decimal column is counted and then a multiplication by six (× 6) is performed. The binary counter and the 4221-BCD counter consist of 3:2CSAs [13].

(3) The ODDS partial products in stages (1) and (2) are added by the binary CSA tree and the decimal digit 3:2 compressor to obtain the final double-word length (2*d*) *A* and *B*, where *A* is represented with BCD excess-6 digits and *B* is represented with BCD-8421 digits.

The final non-redundant BCD-8421 product *P=A+B* is achieved by a 2*d*-digit BCD adder [12].

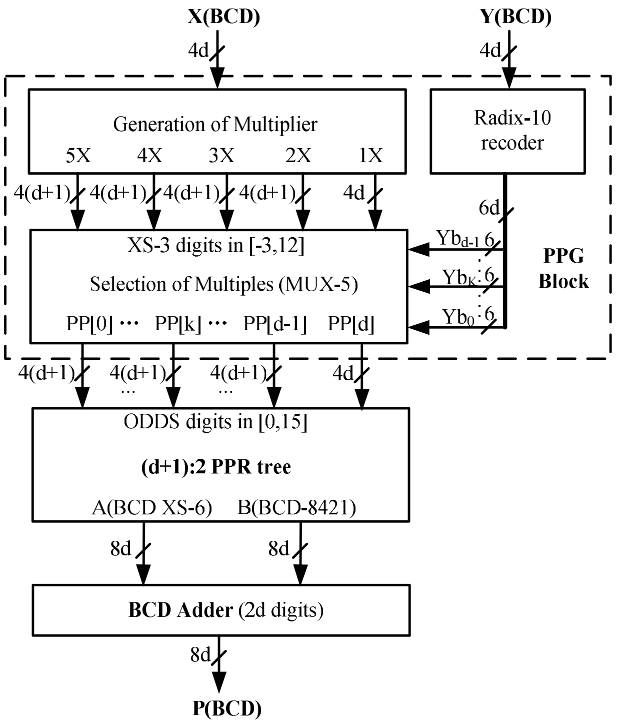


Fig. 2 The radix-10 multiplier using redundant BCD codes (after [7]).

III. The Proposed Partial Product Tree

The proposed BCD multiplier uses a PPG circuit (based on both the ODDS and the XS-3 representations) that is similar as the one used in [7]. The PPG circuit uses the SD radix-10 recoding to reduce the number of multiples with $\{0X, 1X, 2X, 3X, 4X, 5X\}$. The PPR tree is however different from [7]. $(d-1)$ carries (for the maximum digit column height) are generated in [7] by a binary CSA tree; a binary BCD-8421 counter is used in the (counter) correction block and the ODDS partial products are then accumulated [7]. The proposed PPR tree consists of a binary PPR tree block (generating $d-2$ carries in the maximum digit column height), a nonfixed size BCD-4221 counter correction block and a BCD-4221 decimal PPR tree block. The functions of the three blocks are as follows:

(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 sum correction block counts the carries generated between the digit columns in the binary CSA tree and a multiplication by six (× 6) is then performed.

(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., H* and *S*). Every decimal bit H_i is required to be multiplied by 2 before adding *H* and *S*.

A. A 17:2 Binary PPR Tree

The maximum column height in the PP array of a 16×16-digit multiplier is d+1=17. The proposed PPR tree in maximum column height for a 16 × 16-digit multiplier is shown in Fig. 3. The input signals of the 17:2 binary CSA tree are coded in a ODDS representation. A 17:2 binary CSA tree has a 4-stage binary compression. The numbers of input PP rows in the 1st, 2nd, 3rd and 4th compression stages are 17, 9, 6, and 4, respectively, given as follows:

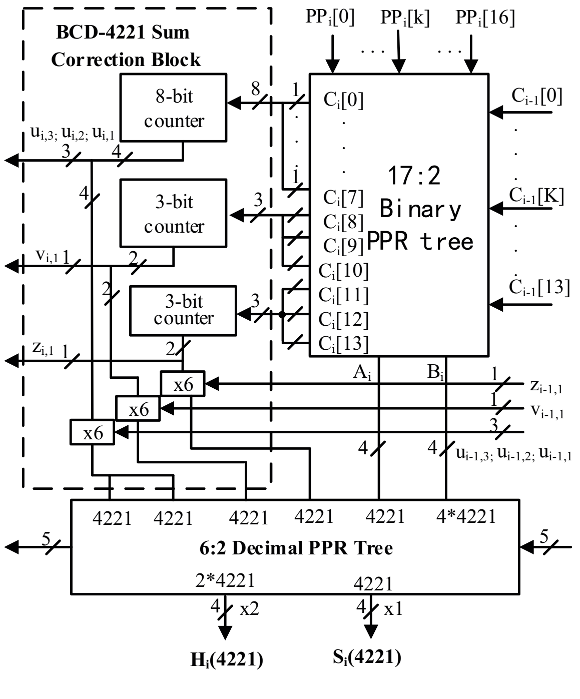


Fig. 3 The proposed PPR tree in the maximum height columns for a 16×16-digit multiplier.

- The first level (4:2 compression) generates 9 PP rows.
- The second level consists of 3:2 compressors, in which 6 PP rows are generated.
- The third level (3:2 compression) generates 4 PP rows.
- The final 2 PP rows are generated by the last level of the 4:2 compression.

The last level (4:2 compression) in the i^{th} position is shown in Fig. 4, it includes four binary 4:2 compressors. In the last compression stage, 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 in BCD-4221 recoding (both 4×BCD-4221 and BCD-4221).

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. 4, only one +6 carry correction term in cout2 is required during the last 4:2 compression stage. The maximum number of carries transferred between adjacent columns of the binary 17:2 CSA tree is 14. These carries are labeled as $C_i[0:13]$ (carry output signals at the i^{th} column) and $C_{i-1}[0:13]$ (carry input signals at the i^{th} column).

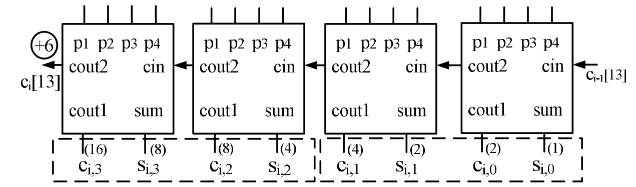


Fig. 4 The 4-bit binary 4:2 compressor in last a compression stage.

The proposed binary 4:2 compressor used in the binary PPR tree is shown in Fig. 5. This compressor is modified from the one given in [20], in which the cout2 signal is generated by NAND gates other than MUXs to reduce its delay.

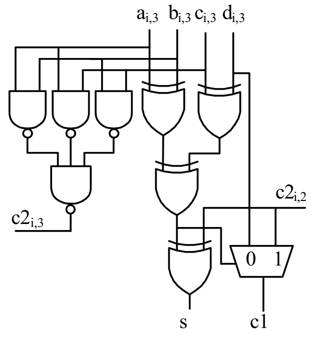


Fig. 5 The proposed binary 4:2 compressor(with weigh of 8).

B. BCD-4221 Counter Correction and 6:2 Decimal PPR Tree

Fig. 6 shows the architecture for a nonfixed size BCD-4221 counter correction block and a BCD-4221 reduction tree block in the maximum height columns that determine the critical path delay of the proposed PPR tree. The 14-bit binary carry output signals (i.e. $C_i[0:13]$) generate 4 BCD-4221 decimal correction terms. As shown in Fig. 3 and Fig. 6, the 8-bit, 3-bit and 3-bit BCD-4221 decimal counters are used to count the 14-bit binary output carries due to the following reasons and objectives:

(1) The BCD-4221 counter is faster than a binary counter, that has only two 3:2 CSA delay;

(2) To balance the paths in the PPR tree and reduce the critical path delay in the decimal 6:2 PPR tree;

(3) 3-bit counters are used to generate a BCD-4221 decimal correction digit by using only a 3:2 compressor.

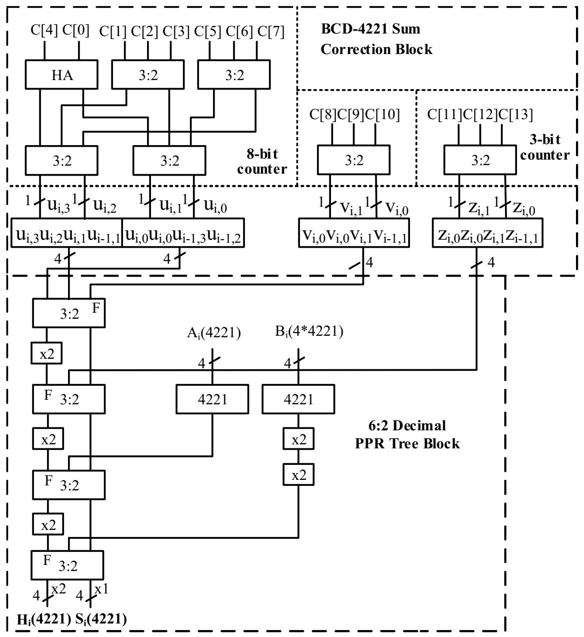

C[4] C[0] C[1] C[2] C[3] C[5] C[6] C[7]
BCD-4221 Sum Correction Block
HA
3:2
C[8]C[9]C[10]
C[11]C[12]C[13]
8-bit counter
3-bit counter
$u_{i,3}$ $u_{i,2}$ $u_{i,1}$ $u_{i,0}$
$v_{i,1}$ $v_{i,0}$
$z_{i,1}$ $z_{i,0}$
$u_{i,3}u_{i,2}u_{i,1}u_{i-1,1}$
$u_{i,0}u_{i,0}u_{i-1,3}u_{i-1,2}$
$v_{i,0}v_{i,0}v_{i,1}v_{i-1,1}$
$z_{i,0}z_{i,0}z_{i,1}z_{i-1,1}$
1
4
F
$A_i(4221)$
$B_i(4*4221)$
4221
x2
6:2 Decimal PPR Tree Block
x1
$H_i(4221)$ $S_i(4221)$

Fig. 6 The maximum height columns of the PPR tree (F indicates the fast input in a 3:2 CSA).

As shown in Fig. 6, the 8-bit 4221 counter generates the 4-bit decimal BCD-4221 digit U_i. The U_i and $6 \times U_i$ in the *i*th bit are given as follows:

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

$$6 \times U_i = 20 \times u_{i,3} + 10 \times u_{i,2} + 10 \times u_{i,1} +$$
$$4 \times u_{i,3} + 2 \times u_{i,2} + 2 \times u_{i,1} + 4 \times u_{i,0} + 2 \times u_{i,0} \quad (7)$$

According to Eq.(7), the digit with weight of 20 and 10 in $6 \times U_i$ is transfered to the *(i+1)th* column, while the digit with weight of 20 and 10 in $6 \times U_{i-1}$ is transfered to the *i*th column. As shown in Fig. 6, the 8 carry output signals generate two BCD-4221 correction terms. The 3-bit counter produces a 2-bit digit V_i. The later one uses full adders to generate the sum bits *i.e.,* $v_{i,1}$ and the carry bit $v_{i,0}$ as follows:

$$V_i = 2 \times v_{i,1} + v_{i,0} \quad (8)$$

$$6 \times V_i = 10 \times v_{i,1} + 4 \times v_{i,0} + 2 \times v_{i,0} + 2 \times v_{i,1} \quad (9)$$

The first term in Eq.(9) is transfered to the *(i+1)th* column and the digit with weight of 10 in $6 \times \mathrm{U}_{i-1}$ is transfered to the *i*th column. Therefore, 3 carry output signals generate one BCD-4221 correction term; similarly, another 3-bit counter (with inputs $\mathrm{C}_i[11{:}13]$ and outputs $\mathrm{z}_{i,1}$ and $\mathrm{z}_{i,0}$) also generates one BCD-4221 correction term. The 14-bit binary carry output signals generate four BCD-4221 correction terms. The correction block and 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 decimal PPR tree block to generate the final two PP rows. As shown in Fig. 6 in the 6:2 decimal PPR tree block, the slowest outputs are connected to fast inputs of the next 3:2 CSA to balance the critical path delay.

IV. Evaluation and Comparison

Using the proposed PPR tree, the 16×16-digit multipliers are evaluated in this section; the results are compared with both a non-redundant decimal multiplier [13] and the redundant decimal multipliers [7] as corresponding to some of the best designs found in the technical literature.

The decimal multiplier designs are described at gate level in Verilog HDL and verified by Synopsys VCS using randomly generated input patterns. The synthesis results based on Synopsis Design Complier are given for the proposed design and the previous design. The logic effort (LE) model [21] is used to obtain an estimate of the area and delay for the proposed decimal multipliers.

The LE model is technology independent and appropriate to provide reasonable accuracy for comparing different designs [7], [21]. Although the input and output gate loads have been taken into account, interconnections and gate sizing are not modeled. The delay is measured in unit of FO4 and the complexity is provided by the number of two-input NAND gates.

Table I shows the delay, input capacitance and area of the main components used in the proposed BCD multipliers. Refer to [7] [21] for more details on the LE-based model, The area and delay obtained for the 16×16-digit multipliers are listed in Table II.

TABLE I Area and Delay eqs. of Main Building Blocks in the LE-Based Model [7].

Component	Delay #FO4	Lin (#inv)	Area (NAND)
NAND2	$0.4+0.2L_{out}$	4/3	1
XOR2	$0.9+0.2L_{out}$	2	2.5
FA (sum, carry)	$(2.2, 0.8)+0.2L_{out}$	5	10
Binary 4:2 CSA	$4.0+0.2L_{out}$	5	20
4-bit binary CLA	$4.5+0.2L_{out}$	5	45
BCD (4-bit) CLA	$5.4+0.2L_{out}$	5	60

Table III compares the proposed BCD multipliers with the BCD combinational multipliers [7], [13] in terms of area and delay as obtained from the LE-based model; the proposed 16×16-digit multiplier reduces the delay by up to 13.89% compared with the non-redundant BCD multiplier of [13], while occupying 8.65% less area. Compared with the 16 × 16-digit redundant BCD multiplier [7], the proposed design reduces the delay by 2.23% at an increase in area (6.05%).

TABLE II Area and Delay (LE-Based Model) for the Proposed 16×16-digit Multipliers.

Block	Delay #FO4	Area #NAND2
PPG Stage	10.2	14900
PPR Tree	25.3	14306

Adder Setup	3.2	1050
Decimal Adder	11.5	2400
Total	50.2	32656

TABLE III AREA AND DELAY (LE-BASED MODEL) COMPARISION FOR DIFFERENT BCD MULTIPLIER DESIGNS.

Design	Delay #FO4	Area #NAND2
Non-Redundant [13]	58.3	35750
Redundant [7]	51.4	30600
Proposed	50.2	32656

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; standard buffers of a 2X strength are used for both the input driver and the output load. The option for logic structuring is turned off to prevent the tool from changing the structure of the unit cells.

As some parts in PPR circuit structure is not provided in details in [7] (only the dot-diagrams are given), so accurate synthesis data is not available [7]. As per [7], the redundant radix-10 multiplier of [7] reduces the delay by 10.75% and the area by 11.1% compared with a non-redundant radix-10 multiplier [13]. Table IV summarizes the delay and area of the 16×16-digit decimal multiplier [13] and the proposed 16×16-digit decimal multiplier. The delay and area are compared separately. The proposed design reduces the delay by 12.30% compared with a non-redundant radix-10 multiplier [13]; the proposed design also reduces the area by 10.9% compared with a non-redundant radix-10 multiplier [13].

TABLE IV DESIGN RESULTS OF 16×16-DIGIT PARALLEL DECIMAL MULTIPLIERS (USING NANGATE 45NM OPEN CELL LIBRARY)

Scheme	Delay (ns)	Ratio	Area(μm^2)	Ratio
Proposed	3.21	1	43053.5	1
Non-Redundant [13]	3.66	1.14	48326.1	1.12

V. CONCLUSIONS

A parallel decimal multiplier based on a new PPR tree has been proposed in this paper for improved performance. In the proposed PPR tree, a BCD-4221 sum correction block is used to count the 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 LE model show that the proposed decimal multiplier is faster 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) and Natural Science Foundation of Jiangsu Province (No. BK20151477). 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] IEEE Std 754(TM)-2008, IEEE Standard for Floating-Point Arithmetic, IEEE Computer Society, Aug. 2008.

[3] A. Aswal, M.G. Perumal, and G.N.S. Prasanna, “On Basic Financial Decimal Operations on Binary Machines”, *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1084-1096, Aug. 2012.

[4] L. Dadda, “Multioperand Parallel Decimal Adder: A Mixed Binary and BCD Approach”, *IEEE Transactions on Computers*, vol. 56, no. 10, pp. 1320-1328, Oct. 2007.

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

[6] M. A. Erle, E. M. Schwarz and M. J. Schulte, “Decimal Multiplication With Efficient Partial Product Generation”, in *Proc. 17th IEEE Symposium on Computer Arithmetic*, pp. 21-28, Jun. 2005.

[7] A. Vazquez, E. Antelo, and J. Bruguera, “Fast Radix-10 Multiplication Using Redundant BCD codes”, *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 1902-1914, Aug. 2014.

[8] 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.

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

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

[11] G.Jaberipur, and A.Kaivani, “Improving the Speed of Parallel Decimal Multiplication”, *IEEE Transactions on Computers*, vol. 58, no. 11, pp. 1539-1552, Nov. 2009.

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

[13] A. Vazquez, E. Antelo and P. Montuschi, “Improved Design of High-Performance Parallel Decimal Multipliers”, *IEEE Transactions on Computers*, vol. 59, no. 5, pp. 679-693, May 2010.

[14] 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.

[15] 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.

[16] Braun, Edward L. *Digital computer design : logic, circuitry, and synthes is*. Academic Press, 1963.

[17] 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.

[18] A. Svoboda, “Decimal Adder with Signed Digit Arithmetic”, *IEEE Transactions on Computers*, vol. C, pp. 212-215, Mar. 1969.

[19] R. D. Kenney and M. J. Schulte, “High-Speed Multioperand Decimal Adders”, *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 953-963, Aug. 2005.

[20] D. Radhakrishnan, 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.

[21] A. Vazquez and E. Antelo, Area and Delay Evaluation Model for CMOS Circuits, Internal Report, University of Santiago de Compostela, Jun. 2012. Available: http://www.ac.usc.es/node/160