Computers and Electrical Engineering 40 (2014) 2126–2138

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

# On high-performance parallel decimal fixed-point multiplier designs[☆]

Ming Zhu[a], Yingtao Jiang[a,*], Mei Yang[a], Tianding Chen[b]

[a] Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, Las Vegas, NV 89154, USA
[b] Department of Information & Electronics Engineering, Zhejiang Gongshang University, Hangzhou, Zhejiang, China

A R T I C L E I N F O

*Article history:*
Received 10 July 2013
Received in revised form 26 August 2014
Accepted 27 August 2014
Available online 26 September 2014

*Keywords:*
Decimal
Multiplication
8421 BCD
5421 BCD
5211 BCD
4221 BCD

A B S T R A C T

High-performance, area and power efficient hardware implementation of decimal multiplication is preferred to slow software simulations in various key scientific and financial applications, where errors caused by converting decimal numbers into their approximate binary representations are unacceptable. This paper presents a parallel architecture for fixed-point 8421-BCD-based decimal multiplication. In essence, it applies a hybrid 8421–5421 recoding scheme to generate partial products, and accumulates them with 8421 carry-lookahead adders organized as a tree structure. In addition, we propose a 4221-BCD-based decimal multiplier that is built upon a novel 4221-BCD full adder; operands of this 4221 multiplier are directly represented in the 4221 BCD. The proposed $16 \times 16$ decimal multipliers are compared with other best-known decimal multiplier designs with a TSMC 90-nm technology, and the evaluation results show that the proposed 8421–5421 multiplier achieves the lowest delay and area, as well as the highest power efficiency, among all the existing hardware-based BCD multipliers.

## 1. Introduction

Decimal-based computer arithmetic has been around since the era of the Electronic Numerical Integrator And Computer (ENIAC) [1]; however, this was quickly replaced by binary arithmetic for two reasons:

(i) Representing the 10 decimal numbers with a four-bit binary-coded-decimal (BCD) is much less efficient than representing 16 binary numbers with the same four bits.
(ii) Decimal arithmetic operations are typically much more complex and slower than their counterpart binary arithmetic operations due to the wider value range of each digit (from 0 to 9) and the representation redundancy.

Even so, support for decimal arithmetic still is appreciated in niche financial and many other key scientific applications, where errors caused by converting a decimal number to its approximate binary representation are often unacceptable; for example, the decimal number of 0.2 cannot be exactly represented in binary. Data in these applications are stored in BCD representations, and computations are processed by using software approaches that are typically 100 to 1,000 times slower than hardware-based binary computations [2].

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Srinivasan Rajavelu.
[*] Corresponding author.
*E-mail addresses:* zhum2@unlv.nevada.edu (M. Zhu), yingtao.jiang@unlv.edu (Y. Jiang), mei.yang@unlv.edu (M. Yang), chentianding@163.com (T. Chen).

http://dx.doi.org/10.1016/j.compeleceng.2014.08.013
0045-7906/© 2014 Elsevier Ltd. All rights reserved.

Thanks to the rapid advancement of VLSI technologies, and partially driven by the release of the IEEE 754 Standard [3], which has newly added specifications governing decimal computations (e.g., the significand of a double-precision decimal floating-point number can represent a decimal value of 16 digits), hardware-based implementation of decimal computations have enjoyed revived interest.

A decimal multiplication generally consists of two major stages: (i) the partial product generation (PPG) stage, where the partial products are computed by multiplying the multiplicand with each multiplier digit and (ii) the partial product accumulation (PPA) stage, where all the partial products are shifted and added together to obtain the final multiplication product.

In this paper, various techniques of hardware-based implementation of decimal multiplication are explored. An 8421–5421-BCD-based multiplier is proposed where both the logic and architecture of the PPG and PPA are simplified. Further, two 4221-BCD multipliers are developed based on a novel 4221-BCD full adder. All three designs are synthesized with a 90-nm technology from Taiwan Semiconductor Manufacturing Company (TSMC), and are compared with the best-known designs in the literature. Evaluation reports confirm that the proposed 8421–5421 multiplier achieves a significant speed-up and hardware overhead reduction, and outperforms all the existing BCD multipliers in terms of delay, circuit area, and power.

The remainder of the paper is organized as follows. Section 2 reviews the previous work. The proposed 8421 and 4221 BCD multipliers are detailed in Sections 3 and 4, respectively. Performance evaluations of various BCD multiplier designs are reported and analyzed in Section 5. Finally, the conclusion is summarized in Section 6.

## 2. Prior work

In this section, various design techniques for PPGs and PPAs in decimal multipliers are reviewed. First, however, the terms and acronyms that shall appear throughout this paper must be defined.

### 2.1. Definitions

(1) A binary number $B$ is formally expressed as $(B)_2$, and a decimal number $C$ as $(C)_{10}$ or just $C$.
(2) *bit:* Each bit has a value of either 0 or 1.
(3) *digit:* Each arabic number is a decimal digit, with a value ranging from 0 to 9. Unless explicitly stated otherwise, an $m$-by-$n$ multiplication means an $m$-digit decimal multiplicand multiplies with an $n$-digit decimal multiplier.
(4) *BCD:* In this paper, four sets of BCD codes are used, namely, 8421, 5421, 4221, and 5211. Table 1 tabulates all the valid representations of a digit for the four BCD schemes. Unless explicitly stated otherwise, 8421 means 8421-BCD representation, and so forth. An $n$-digit BCD $A$ is given as: $A_n = A_n[4n-1:0] = a_{n-1}\ldots a_0$, where $n$ is the digit length and the total bit length of $A_n$ is $4n$. The $i$-th digit $a_i = a_i[3:0]$, $i = 0,\ldots,n-1$ has a weight of $10^i$.
(5) "*Radix*-4", "*Radix*-5", and "*Radix*-10": Across the literature, these terms are used in different contexts with various meanings [4,5]. This paper follows the definition given in [5]: "*Radix-n*" refers to the decoding scheme that is used, instead of the number representation system. Details of various decoding schemes are discussed in Section 2.3.

### 2.2. Introduction of decimal multiplication algorithms

An $n$-by-$n$ BCD multiplication, $A_n \times B_n$, generally consists of two major stages: PPG and PPA. This process can be mathematically described as:

$$P_{2n} = A_n \times B_n = A_n\left(\sum_{i=0}^{n-1}(b_i(10)^i)\right) = \sum_{i=0}^{n-1}\left((A_nb_i)(10)^i\right) = \sum_{i=0}^{n-1}\left(PPi_{n+1}(10)^i\right). \quad (1)$$

$PPi_{n+1} = A_n \times b_i$ is the $i$-th generated partial product, given by the $n$-digit multiplicand $A_n$ and $b_i$, the $i$-th digit of the multiplier, $B_n$. These partial products are shifted and added together to obtain the final multiplication product. This multiplication process is depicted in Fig. 1.

**Table 1**
BCD representations [6].

| Decimal value | 8421-BCD | 5421-BCD | 4221-BCD | 5211-BCD |
|---|---|---|---|---|
| 0 | 0000 | 0000 | 0000 | 0000 |
| 1 | 0001 | 0001 | 0001 | 0001 \| 0010 |
| 2 | 0010 | 0010 | 0010 \| 0100 | 0011 \| 0100 |
| 3 | 0011 | 0011 | 0011 \| 0101 | 0101 \| 0110 |
| 4 | 0100 | 0100 | 0110 \| 1000 | 0111 |
| 5 | 0101 | 1000 | 1001 \| 0111 | 1000 |
| 6 | 0110 | 1001 | 1100 \| 1010 | 1010 \| 1101 |
| 7 | 0111 | 1010 | 1101 \| 1011 | 1100 \| 1011 |
| 8 | 1000 | 1011 | 1110 | 1110 \| 1101 |
| 9 | 1001 | 1100 | 1111 | 1111 |