Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

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

Improved designs of digit-by-digit decimal multiplier

Syed Ershad Ahmed[a,*], Santosh Varma[a], M.B. Srinivas[b]

[a] *Department of Electrical Engineering, BITS Pilani, Hyderabad Campus, India*
[b] *School of Engineering and Technology, BML Munjal University, Gurgaon, India*

ARTICLE INFO

Keywords:
Decimal arithmetic
Digit-by-digit multiplier
Binary-to-BCD converter
Partial product generation
Partial product reduction
Multi-operand addition

ABSTRACT

Decimal multiplication is a ubiquitous operation which is inherently complex in terms of partial product generation and accumulation. In this paper, the authors propose a generalized design approach and architectural framework for ‘digit-by-digit’ multiplication. Decimal partial products are generated in parallel using fast and area efficient BCD digit multipliers and their reduction is achieved using hybrid multi-operand binary-to-decimal converters. In contrast to most of the previous implementations, which propose changes either in partial product generation or reduction, this work proposes modifications at both partial product generation and reduction stages resulting in an improved performance. A comprehensive analysis of synthesis results (consistent with IEEE-compliant 16-digit decimal multiplier architecture) indicates an improvement in delay of 8–29% and a reduced area-delay product of 4–38% compared to similar work published previously.

1. Introduction

Decimal arithmetic is preferred in applications such as financial, scientific and commercial owing to their higher precision compared to binary arithmetic [1]. However, these computations are generally sluggish (slow) and tend to occupy more silicon area. This has led to efforts in improving decimal architectures to enable high performance and compact arithmetic circuits [2,3]. Like in binary arithmetic, one of the most vital and common operations in decimal arithmetic, is multiplication. While a large body of literature on decimal arithmetic covers serial multiplication [4,5], parallel (‘word-by-digit’) [6–9] and (‘digit-by-digit’) [10,11] multiplication has also been reported. Decimal (BCD) ‘digit-by-digit’ multipliers are appropriate for pipelined computations and result in improved regularity of the circuits. This regularity, in conjunction with shorter interconnects, results in improvement in the multiplier performance [12]. Thus, this work focuses on efficient design and implementation of ‘digit-by-digit’ multipliers [10]. Though, more recent literature shows other efficient implementations, they are not relevant to present work since their focus is not on ‘digit-by-digit’ multiplication.

The partial products in ‘digit-by-digit’ multiplication scheme are generated using BCD digit-multiplier (BDM) and their reduction is accomplished using a carry-free binary adders, multi-operand binary-to-decimal (BD) converters and decimal adder. Since BDM is an important component in partial product generation, we focus on new and improved designs for BDM cells in this paper. Besides, novel designs of multi-operand BD converters are proposed to convert the column binary sum to decimal in partial product reduction. Further, a hybrid multi-operand BD converter algorithm is proposed and analyzed for its performance. It is expected that these improvisations would result in significant savings in terms of area and latency.

Throughout this paper, upper and lower case letters are used to signify decimal digits and binary bits respectively, where a digit represents a 4-bit BCD number. The symbols ‘. ’ and ‘ + ’ are used to denote AND and OR gates while the symbols ‘$\oplus$’ and ‘$\odot$’ denote XOR and XNOR operations respectively. Further, the term binary coded decimal (BCD) is used interchangeably with decimal.

Rest of the paper is organized as follows: Preliminary information related to the existing decimal multiplication algorithms and previous work on partial product generation and reduction are provided in Section 2. An outline of the proposed partial product generation and reduction schemes in 16*16 ‘digit-by-digit’ multiplier is provided and discussed in Section 3 and Section 4. In addition, design of hybrid multi-operand BD converters is described in Section 4.2. A detailed performance analysis of 16*16 ‘digit-by-digit’ multiplier is carried out and compared in Section 5 while conclusions are drawn in Section 6.

2. Decimal multiplication

Decimal multiplication architectures typically have the following stages: (i) partial product generation (ii) partial product reduction and (iii) final product computation. The ‘digit-by-digit’ multiplication architecture is presented in Section 2.1 while the various existing partial product generation and reduction schemes adapted in these designs are discussed below.

* Corresponding author.
E-mail address: syed@hyderabad.bits-pilani.ac.in (S.E. Ahmed).

https://doi.org/10.1016/j.vlsi.2017.12.001
Received 24 July 2017; Received in revised form 25 October 2017; Accepted 3 December 2017
0167-9260/ © 2017 Elsevier B.V. All rights reserved.

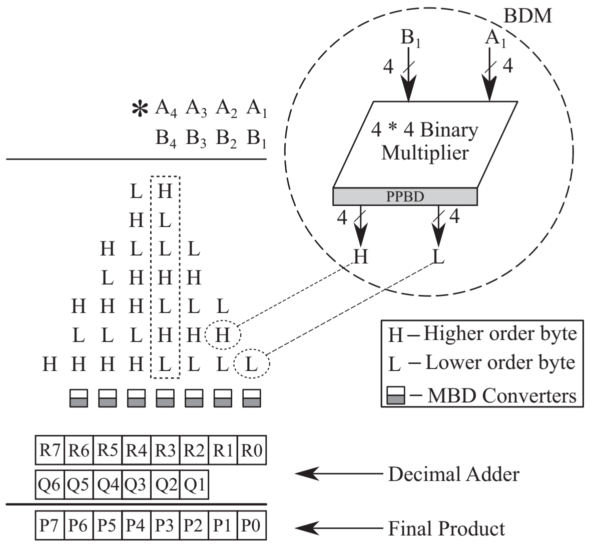

Fig. 1. Example of 4*4 'digit-by-digit' multiplication using BDMs.

2.1. 'Digit-by-digit' multiplier

A step by step implementation of 4*4 'digit-by-digit' multiplication [10] is illustrated in Fig. 1. Multiplication of each digit of the multiplicand with the digit of multiplier is performed using the BDM.

For example, multiplication of A_1 and B_1 is highlighted in the dotted circle of the figure. The output of the BDM results in most significant digit and least significant digit denoted by H and L respectively. A typical BDM is composed of a 4*4 binary multiplier and a partial product binary-to-decimal (PPBD) converter. Most of the previous works available in the literature have focused on PPBD converters at partial product generation stage which is discussed below.

The individual decimal partial product columns (one such column is highlighted with dotted rectangle in Fig. 1) are compressed in parallel by using a tree of binary carry save adders [13] resulting in a binary number as output of each column. The conversion from binary-to-decimal is carried out using multi-operand BD (MBD) converters resulting in rows of decimal digits $R0 - R7$ and $Q1 - Q6$ which are eventually compressed using a decimal adder to obtain the final product ($P0 - P7$).

2.2. Existing partial product generation schemes

Existing Binary to Decimal (BD) converters involved in partial product generation are described in detail in [10,14–17]. The algorithm proposed in [10] converts a 7-bit binary number to a 2-digit BCD number (D_H and D_L) to support high performance decimal multiplication. This algorithm calculates the contributions for lower BCD digit (D_L) and the higher BCD digit (D_H) from each of the input binary bits. However, this approach appears to be faulty as pointed out by Bhattacharya et.al. [14] who modify this algorithm by adding the contributions in a BCD fashion.

Their approach partitions or splits the binary input into two subparts, that is, three MSBs and four LSBs. It then calculates the contributions to the two BCD digits and adds them in a BCD fashion to get the final result. An improved architecture is presented in [15] that proposes two schemes, 'three-four split' and 'four-three split' binary to BCD converters. The 'three-four split' design has optimized D_L and D_H generator blocks resulting in better performance in terms of delay. The 'four-three split' design partitions the 7-bit binary input into four MSBs and three LSBs. Since the LSBs do not contribute to the higher BCD digit, the LSB contribution generator is removed resulting in area savings. However, this comes at the cost of increased complexity of MSB digit generator. The 'three-four split' design is faster than the 'four- three split' one whereas the 'four-three split' results in a more area efficient design.

Authors in [17] propose a BD converter based on Shift-Add3 algorithm. The resulting architecture however tends to have large area and delay due to the redundant contribution blocks. A recent work [16] introduces new methods for BCD-digit multiplication. The most effective one among these methods results in what is called a new N2 BD converter whose limitation however is its high latency.

In this work, we propose two different partial product BD converters namely, high-speed PPBD converter and low-area PPBD converter, to overcome the shortcomings mentioned above.

2.3. Existing partial product reduction schemes

In a scheme proposed by Dadda [13], partial product reduction in the first stage of 'digit-by-digit' multiplier is achieved using a binary carry save adder structure. The binary result of each partial product column is subsequently converted to decimal (BCD) using a multi-operand BD converter consisting of an iterative connection of Nicoud cells [18], as suggested by Dadda [19]. A typical Nicoud (ND) cell would accept a 4-bit binary input (b_j), multiplies it by two, and then adds it to b_i as depicted in Fig. 2(a). Thus the computation of BCD (decimal) outputs, b_0 (higher digit) and D_0 (lower digit), is carried out using the relation $\{b_0,\ D_0\} = 2\ .\ b_j + b_i$ where the maximum values of b_0 and D_0 are $(1)_{10}$ and $(9)_{10}$ respectively.

An example to convert a binary number to two BCD digits (b_0 and D_0) is illustrated in Fig. 2(b). Since the binary number $(1010011)_2$ to be converted here is larger than $(19)_{10}$, four Nicoud cells are required to realize the converter. As illustrated in Fig. 2(b), the input to the Nicoud cell is restricted to $(1001)_2$. Hence the 3 MSBs of binary input along with '0' prepended is accepted as b_j and the next significant binary input '0' as b_i resulting in the outputs $(1)_2$ and $(0000)_2$. The 4-bit output $(0000)_2$ of cell 1 along with the next significant binary input '0' form input to cell 2, resulting in $(0)_2$ and $(0000)_2$. Similarly, the 4-bit output of each subsequent cell along with residual 1-bit binary input feeds the decimal input of the following cell resulting in higher (P) digit $(1000)_2$ and lower (Q) digit $(0011)_2$. In general, binary number of any operand width can be converted to decimal by a linear arrangement of Nicoud cells.

The limitation of Nicoud cells however is their latency and thus the delay of multi-operand BD converter increases with the size of the

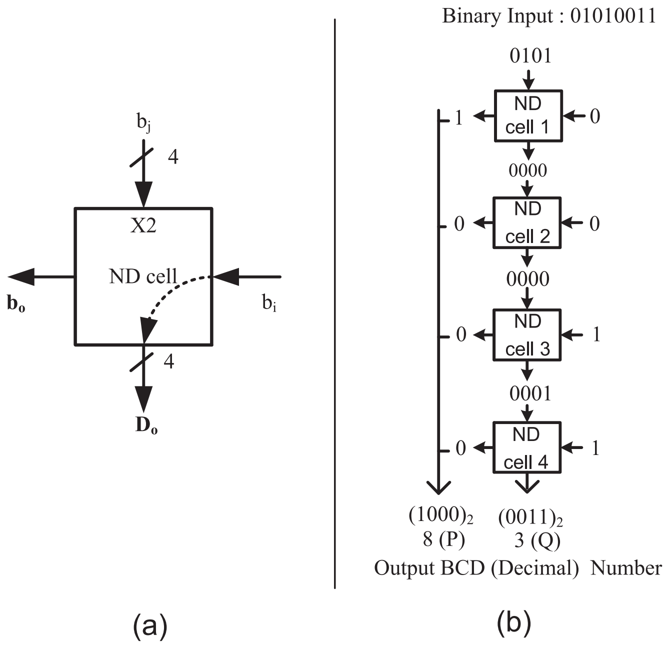

Fig. 2. (a) Compact notation of Nicoud cell (b) Linear array of Nicoud cells to form Dadda multi-operand BD converter.