RESEARCH ARTICLE OPEN ACCESS

DESIGN AND IMPLEMENTATION OF FAST DECIMAL MULTIPLIER USING SMSD ENCODING TECHNIQUE

S.Sirisha
PG Scholar
Department of Electronics and Communication Engineering
AITS, Kadapa, India

T.Vijaya nirmala
H.O.D.&Asst.Prof
Department of Electronics and Communication Engineering
AITS, Kadapa, India

ABSTRACT:

Decimal Xx Y propagation is a complex procedure, where intermediate partial products (IPPs) are commonly chosen from some precomputed radix-10 X multiples. Some works require only [0, 5] x X via recoding digits of Y to one-hot rendering of signed digits in [-5, 5]. This kind of reduces the choice reason at the expense of one extra IPP. Two's complement signed-digit (TCSD) encoding is often used to represent IPPs, where dynamic negation (via one xor per little bit of X multiples) is required for the recoded digits of Y in [-5, -1]. In this paper, despite technology of 17 IPPs, for 16-digit operands, we control to get started on the partial product reduction (PPR) with sixteen IPPs that enhance the VLSI regularity. Moreover, we save 75% of stopping xors via representing precomputed multiples by sign-magnitude signed-digit (SMSD) encoding. For the first-level PPR, we develop an effective adder, with two SMSD input numbers,in whose sum is represented with TCSD encoding. After that, multilevel TCSD 2:1 reduction causes two TCSD accumulated partial products, which collectively undergo a special early initiated conversion{plan to access the last binary-coded decimal product. Because such, a VLSI execution of 4x4-digit parallel digit multiplier is synthesized, where evaluations show some performance improvement over previous relevant designs.

keywords: **Radix-10 multiplier, redundant representation, sign-magnitude signed digits (SMSDs), VLSI design.**

I. INTRODUCTION

Fast radix-10 multiplication, in particular, can be achieved via parallel partial product generation (PPG) and partial product reduction (PPR), which is, however, highly area consuming in VLSI implementations. Therefore, it is desired to lower the silicon cost, while keeping the high speed of parallel realization. The choice of alternative IPP representations is influential on the PPG, which is of particular importance in decimal multiplication from two points of view: one is fast and low cost generation of IPPs and the other is its impact on representation of IPPs, which is influential on PPR efficiency. Straightforward PPG via BCD digit-by-digit multiplication [8], [9] is slow, expensive, and leads to n double-BCD IPPs for n×n multiplication (i.e., 2n BCD numbers to be added). However, the work of [10] recodes both the multiplier and multiplicand to signmagnitude signed digit (SMSD) representation and uses a more efficient 3-b by 3-b PPG. Nevertheless, following a long standing practice [11], most PPG schemes use precomputed multiples of multiplicand X

(or X multiples). Precomputation of the complete set0, 1,... 9} × X, as normal BCD numbers, and the subsequent selection are also slow and costly. A common remedial technique is to use a smaller less costly set that can be achieved via fast carry-free manipulation (e.g., 0, 1, 2, 4, 5} × X) at the cost of doubling the count of BCD numbers to be added in PPR; that is, n double-BCD.

The recoding of multiplier's digits, in some relevant works [4], [6], [7], leads to a carry bit besides the n recoded digits of the multiplier, which will generate an extra partial product. This is particularly problematic for parallel multiplication with n = 16 (i.e., number of significand's decimal digits according to IEEE standard size of single precision radix-10 floating-point numbers [12]), where the 17 generated partial products require five PPR levels instead of four (i.e., log2 16). Furthermore, they dynamically negate positive multiples based on the sign of multiplier's recoded digits. This technique reduces the area and delay of logic that selects the X multiples at the cost of conditionally negating the selected multiples, which requires at least 4n2 XOR gates for n×n multiplication

II. LITERATURE SURVEY

In [4], [6], and [7], dynamic negation of precomputed X multiples reduces their selection cost at the penalty of one XOR gate per each bit of the selected positive multiple. This negation cost is replicated n times for parallel n×n multiplication. Moreover, the n inserted 1s for 10's complementation in [6] and n × (n +1) 1s for digitwise two's complementation in [7] have a negative impact on area and power saving. The same is true for the correction constant, and more complex recoding due to zero handling, for [0, 15] partial products in [4]. One way to save these costs, as we do in Section III, is to generate the SD precomputed X-multiples with sign magnitude format, so as to reduce the XOR gates to one per digit (roughly 75% savings in the number of negating XOR gates) and remove the aforementioned negative impacts. However, besides slowing down the PPG to some extent (e.g., in comparison with radix-5 implementation of [6]), new problems are introduced in PPR, which are explained and solved in the next section, where we also reduce the depth of IPP matrix to n = 16, effectively prior to termination of PPG.

The column acronyms MR, ME, DN, #OP, PPDS, and PPDE stand for multiplier recoding, multiple encoding, dynamic negation of IPPs, number of operands to be added (i.e., number of originally generated nonredundant decimal numbers, or SD partial products), partial product digit set, and partial product digit encoding, respectively. In Svoboda [13] refers to the encoding of a digit d ∈ [0, 6]([−6, −0]) by a 5-bit binary number 3d(31 − 3d). Some other works on decimal multiplication with floatingpoint operands [14]–[17], specific designs for Field Programmable Gate Array (FPGA) (see [18], [19]), or digit-by-digit iterative approach (see [9]) are not listed , since they are based on one of the tabulated works, or use embedded FPGA components, which are out of the scope of this paper.

III. SMSD DIGITS

Decimal SDs in [−α, α] (α ≤ 7) are usually encoded with minimal 4-b signed numbers. For example, consider α = 5 in [10] and α = 7 in [23] with sign magnitude and two's complement representations, respectively. The latter is suitable for basic arithmetic operations, except for negation, which is best, performed on sign magnitude format.

In this section, we propose a decimal multiplication scheme with the following characteristics, which are in the same line as those of the designs is

1) [−5, 5] SMSD recoding of multiplier's digits;

2){ 0, 1, 2, 3, 4, 5} × X precomputed multiples;

3) 4-b [−6, 6] SMSD encoding of precomputed multiples;

4) Dynamic negation of multiples with only one XOR per digit (i.e., per 4 b);

5) N (instead of n + 1) operands to be added for n × n multiplication;

6) Unified SMSD + SMSD −→ TCSD adder for all four input sign combinations;

7) [−7, 7] TCSD representation for accumulated partial products;

8) Early start of redundant to BCD conversion;

9) Augmenting last PPR level with final conversion to BCD.

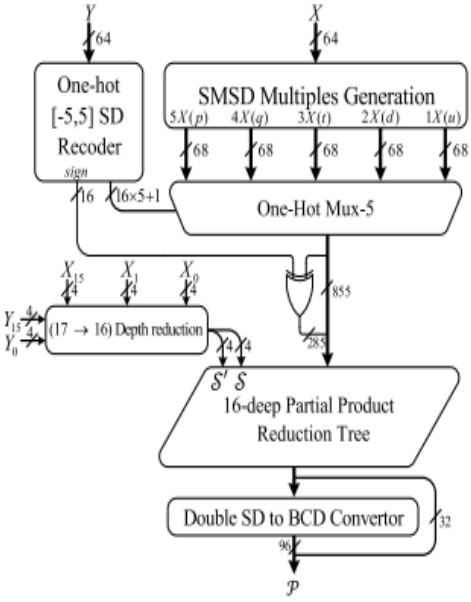


Fig. 1 Block diagram of the proposed multiplier

Fig. 1 depicts the general architecture of the proposed 16×16 multiplication P = X ×Y ; the details of each building block will be explained later. In particular, in the top three blocks, the multiplier's digits are recoded to n one-hot [−5, 5] SMSDs (i.e., one sign and five magnitude bits), augmented with a 10n-weighted carry bit. The multiples [0, 5] × X are precomputed as n [−6, 6] SMSDs and a 10n-weighted [−5, 4] SMSD. Each SMSD contains a sign bit s and 3-b magnitude. The negative multiples [1, 5]× (−X) are achieved via dynamic sign inversion of multiples[1, 5] × X at the cost of only one XOR gate per digit.

A. *Recoding of Multiplier's Digits*

Original BCD digits of multiplier require [0, 9]×X precomputed multiples, which include hard multiples {3, 6, 7, 9} × X that unlike {2, 4, 5, 8} × X are not derivable without carry propagation. On the other hand, BCD-to-redundant [−5, 5] SMSD recoding of multiplier's digits with dynamic negation of IPPs reduces the required X multiples to [0, 5] × X that include only one hard multiple (i.e., 3X). However, this recoding produces a carry as the (n + 1)th digit of multiplier, which increases the number of IPPs by 1. This is especially not desirable for n = 16 (i.e., the recommended IEEE754-2008 word size for decimal operands [12]).

The one-hot recoding input/output expressions are given by (1), where Yi = v3v2v1v0, and Yi−1 = w3w2w1w0 represent two consecutive digits of BCD multiplier, ω indicates whether Yi−1 ≥ 5, is the sign of target code, and v1–v5 are one-hot signals corresponding to absolute values of recoded multiplier's digit Y i (i.e., 1–5), whose decimal weight is equal to that of More derivation details can be found in [4], [6], [7], and [10]

B. *Precomputed Multiples*

We need to generate 0, 1, 2, 3, 4, 5}× X, where X is a BCD multiplicand. The only hard multiple 3X can be generated in carry-free manner, if it is represented via a redundant digit set [5], [7]. Therefore, for uniformity sake in PPR, we generate all the required multiples in the same SD number system.

The corresponding logical expressions for SMSD multiples of the multiplicand X (i.e., 1X(u), 2X(d), 3X(t), 4X(q), and 5X(p)) can be derived in terms of bits of BCD digits a (in position i) and b (in position i−1). For example, the logical expression of 3X(t) is given by (3), and the rest can be found in the Appendix. Note that such multiples are represented with at most one

extra digit (i.e., total of 68 bits), since the most significant digit of the generated multiple is at most 4 (due to 5 × 9 = 45), which remains 4 within the BCD-to-SMSD conversion.

C. Partial Product Generation

We reduce the matrix depth to n (e.g., 5 → 4 for n = 4, and 17 → 16 for n = 16), with no delay between the termination of PPG and start of PPR. Here is how it works: we compute sum of the two gray digits independent of (and in parallel with) normal PPG as follows. If $Y_{n-1} \leq 4$, the value of 10n-weighted carry of recoded multiplier is zero, so the bottom gray digit has to be zero. Therefore, no addition is required. For $Y_{n-1} > 4$, let H denote the most significant digit of $X_{n-1} \times Y_0$ where X_{n-1} and Y_0 represent the most significant BCD digit of multiplicand and the least significant recoded digit of multiplier, respectively. We extract H as ten one-hot signals via an eight-input logic (see the rightmost box in Fig. 2).

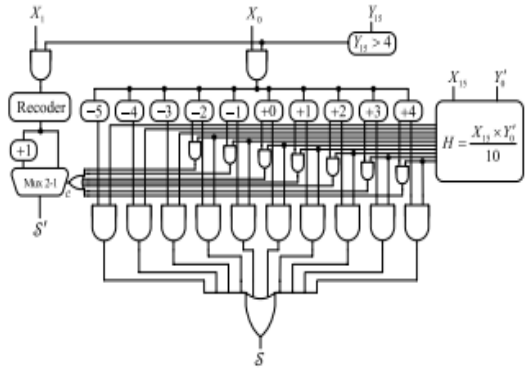


Fig. 2: Required circuit for (17 → 16) depth reduction.

The least significant BCD digit of multiplicand (i.e., X0), as illustrated in the rest of Fig. 2, is added to constants in [−5, 4]. This leads to the desired sum digit S in 10n-weighted position and a carry bit c to be added to the 10n+1-weighted digit next to the bottom gray digit to result. This digit, as shown in the leftmost part of Fig. 2, is obtained by directly recoding the 10-weighted digit of multiplicand (i.e., X1).

D. Partial Product Reduction

The overall PPR for n = 16 is illustrated where a bar, triangle, square, and diamond represent a BCD, [−6, 6] SMSD, [−7, 7] TCSD, and binary signed digit (BSD), respectively. The choice of SMSD representation for the firstlevel IPPs, while facilitating the PPG, bears no extra complexity for PPR, since all reduction levels use TCSD adders, except for the first one that requires a special SMSD+SMSD-to-TCSD adder. However, as will be shown at the end of Section III-D1, this adder is not more complex than a simple TCSD adder

The red-shaded SMSD in level II of Fig. 3 is directly converted to BCD. Similar direct conversions are in order for the red-shaded digits (TCSDs, however) in the subsequent levels III and IV.

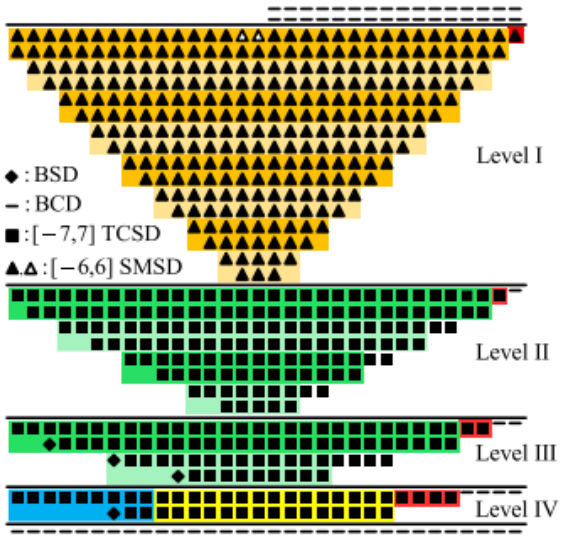


Fig. 3: Overall view of 16 × 16 digit multiplier.

We aim to take advantage of [−5, 5] SMSD recoding of multiplier and dynamic negation of X multiples, while reducing the number of XOR gates via generating [−6, 6] SMSD pre computed X multiples (i.e., just one XOR gate per 4-b digit). Other contributions of this paper are highlighted below

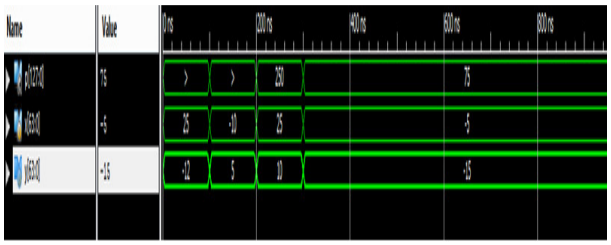
1) Starting the PPR With 16 Partial Products: An especial on the fly augmentation of two middle SMSD digits leads to reducing the depth of partial product matrix by 1, such that the PPR starts with

16 operands right at the end of PPG, with no delay penalty for the latter.

2) Special 4-in-1 SMSD Adder With TCSD Sum: To avoid the challenging addition of SMSD IPPs, we design a novel carry-free adder that represents the sum of two [−6, 6] SMSD operands in [−7, 7] two's complement signed-digit (TCSD) format, where one unified adder is utilized for all the four possible sign combinations.
3) Improved TCSD Addition: The rest of the reduction process uses special TCSD adders that are actually an improved version of the fast TCSD adder. Such 2:1 reduction promotes the VLSI regularity of the PPR circuit, especially for n = 16 (i.e., the recommended operand size of IEEE 754-2008).
4) Augmenting the Final Redundant to Non redundant Conversion with the Last PPR Level: The last PPR level would normally lead to TCSD product, which should be converted to BCD. However, to gain more speed and reduce costs, we device a special hybrid decimal adder with two TCSD inputs and a BCD output.

IV. SIMULATION RESULTS

We could not actually copy the analytical evaluation results of all the reference works, since the work of [2] provides only synthesis results. The results of [6] and [22] are in terms of Fanout of 4 (FO4), where their underlying FO4 evaluation assumptions are not apparently the same and thus could not be followed in the evaluation of our design. Therefore, for fair comparisons, we preferred to derive the entries in rows 1–7 of Tables III–V directly from the design description of the corresponding articles, in the same way that we did with our design. These gate-level evaluations are in terms of G (i.e., delay of a two-input simple gate), which is easily verifiable.



V. CONCLUSION

We propose a parallel 16 × 16 radix-10 BCD multiplier, where 17 partial products are generated with [−6, 6] SMSD representation. The least possible delay for the latter is (existing delay) ns, while the proposed design leads the synthesis tool to meet the (proposed delay) ns time constraint. In other words, the advantage is that the proposed design can operate at higher frequency and dissipate less power with no claim in area improvement.

VI. REFERENCES

[1] M. F. Cowlishaw, "Decimal floating-point: Algorism for computers," in Proc. 16th IEEE Symp. Comput. Arithmetic, Jun. 2003, pp. 104–111.

[2] T. Lang and A. Nannarelli, "A radix-10 combinational multiplier," in Proc. 40th Asilomar Conf. Signals, Syst., Comput., Oct./Nov. 2006, pp. 313–317.

[3] R. D. Kenney, M. J. Schulte, and M. A. Erle, "A high-frequency decimal multiplier," in Proc. IEEE Int. Conf. Comput. Design (ICCD), Oct. 2004, pp. 26–29.

[4] A. Vazquez, E. Antelo, and J. D. Bruguera, "Fast radix-10 multiplication using redundant BCD codes," IEEE Trans. Comput., vol. 63, no. 8, pp. 1902–1914, Aug. 2014.

[5] S. Gorgin and G. Jaberipur, "Fully redundant decimal arithmetic," in Proc. 19th IEEE

Symp.Comput. Arithmetic, Jun. 2009, pp. 145–152.

[6] A. Vazquez, E. Antelo, and P. Montuschi, "Improved design of highperformance parallel decimal multipliers," IEEE Trans. Comput., vol. 59, no. 5, pp. 679–693, May 2010.

[7] L. Han and S.-B.Ko, "High-speed parallel decimal multiplication with redundant internal encodings," IEEE Trans. Comput., vol. 62, no. 5, pp. 956–968, May 2013, doi: 10.1109/TC.2012.35.

[8] G. Jaberipur and A. Kaivani, "Binary-coded decimal digit multipliers," IET Comput.Digit.Techn., vol. 1, no. 4, pp. 377–381, 2007.

[9] R. K. James, T. K. Shahana, K. P. Jacob, and S. Sasi, "Decimal multiplication using compact BCD multiplier," in Proc. Int. Conf. Electron. Design, 2008, pp. 1–6.

[10] M. A. Erle, E. M. Schwarz, and M. J. Schulte, "Decimal multiplication with efficient partial product generation," in Proc. 17th IEEE Symp.Comput. Arithmetic, Jun. 2005, pp. 21–28.

[11] R. K. Richards, Arithmetic Operations in Digital Computers. New York, NY, USA: Van Nostrand, 1955.

[12] IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008, IEEE Standards Committee, 2008, doi: 10.1109/IEEESTD. 2008.4610935.

[13] A. Svoboda, "Decimal adder with signed digit arithmetic," IEEE Trans. Comput., vol. C-18, no. 3, pp. 212–215, Mar. 1969.

[14] B. J. Hickmann, A. Krioukov, M. Schulte, and M. Erle, "A parallel IEEE P754 decimal floating-point multiplier," in Proc. IEEE 25th Int. Conf. Comput. Design (ICCD), Oct. 2007, pp. 296–303.

[15] R. Raafat et al., "A decimal fully parallel and pipelined floating point multiplier," in Proc. 42th Asilomar Conf. Signals, Syst., Comput., Oct. 2008, pp. 1800–1804.

[16] M. A. Erle, B. J. Hickmann, and M. J. Schulte, "Decimal floatingpoint multiplication," IEEE Trans. Comput., vol. 58, no. 7, pp. 902–916, Jul. 2009.