Sign-Magnitude Encoding for Efficient VLSI Realization of Decimal Multiplication

Nandyalamounika1 V.RAMESH[2]
nandyalamounika448@gmail.com[1] ramesh719@gmail.com[2]
[1]PG SCHOLAR, VLSI, Vaagdevi Institute of Technology & Science, Peddisettipalli, Proddatur, Kadapa, Andharpradesh.

[2]Assistant Professor, Department of ECE, Vaagdevi Institute of Technology & Science, Peddisettipalli, Proddatur, Kadapa, Andharpradesh.

Abstract: **Decimal multiplication of X and Y is a complicated operation, where intermediate partial products (IPPs) are commonly selected from a set of pre computed radix-10 X multiples where, X is the multiplicand. Two's complement signed-digit (TCSD) encoding is often used to represent IPPs, where dynamic negation (via one xor per bit of X multiples) is required for negative digits of multiplier Y[-5,-1]. In this paper, despite generation of 17 IPPs, for 16-digit operands, we are implementing 17 to 16 PP reduction circuit that enhance the speed of the multiplier. By implementing sign-magnitude signed-digit (SMSD) encoding we save 75% of negating xors via representing pre computed multiples. At the first level of 16 to 8 PP reduction SMSD+SMSD to TCSD adder with two SMSD input numbers, whose sum is represented with TCSD encoding. Thereafter, multilevel TCSD 2:1 reduction leads to two TCSD accumulated partial products, which collectively undergo a special early initiated conversion scheme. The final sum gets converted to the binary-coded decimal format. As such, a VLSI implementation of 16 × 16-digit parallel decimal multiplier is synthesized, where evaluations show some performance improvement over previous relevant designs.**

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

I.INTRODUCTION

Decimal computer arithmetic is preferred in decimal data processing environments such as scientific, commercial, financial, internet-based applications in monetary, web-based, and human interactive applications. Ever growing needs for processing power, required by applications with intensive decimal arithmetic, cannot be met by conventional slow software simulated decimal arithmetic units. However, their hardware counterparts as an integral part of recently commercialized general purpose processors are gaining importance. Binary-coded decimal (BCD) encoding of decimal digits has conventionally dominated decimal arithmetic algorithms, whether realized by hardware or in software.

The research for hardware realization of decimal arithmetic is not matured yet and there are rooms for improvements in hardware algorithms and designs. For example, the state-of-the-art BCD multipliers, for computing X Y, use iterative multiplication algorithms, where the partial products (i.e. the product of one BCD digit of the multiplier Y times the multi-BCD-digit multiplicand X) are generated one at a time and added to the previously accumulated result. Each partial product may be directly generated as one BCD number in [0, 9] X, or may be composed of few easy multiples of the multiplicand (e.g. 7X ¼ 4X þ 2X þ X). The latter approach tends to increase the depth (measured by the maximum number of equally weighted BCD digits) of partial product tree per each BCD digit of multiplier, which in general leads to slower partial product accumulation. But, by using possibly fast and low-cost BCD digit by BCD-digit multipliers, the former approach may lead to less costly BCD multipliers.

Erle et al. have enumerated three reasons for using decimal digit-by-digit multipliers for partial product generation, which leads to less number of cycles, less wiring and no need for registers to store multiples of the multiplicand. With the rapid advances in VLSI technology, semi(fully)-parallel BCD multipliers will soon be attractive, where more than one (all) partial product(s) are generated at once and accumulated in parallel. An integral building block of a

BCD multiplier, whether realizing a sequential, semi- or fully parallel multiplication algorithm, can be the BCD-digit multiplier. Alternative approaches are based on either slow accumulation of easy multiples, or costly retrieval of product of BCD digits from look-up tables.

II. BACKGROUND

Dynamic negation of pre computed 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 and n× (n+1) 1s for digit wise two's complementation in 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. One way to save these costs, as we do in Section III, is to generate the SD pre computed 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.

III.EXISTING SYSTEM

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. Let P = X × Y represent an n × n decimal multiplication, where multiplicand X, multiplier Y , and product P are normal radix-10 numbers with digits in [0, 9]. Such digits are commonly represented via binary-coded decimal (BCD) encoding. However, intermediate partial products (IPPs) are represented via a diversity of often redundant decimal digit sets.

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).

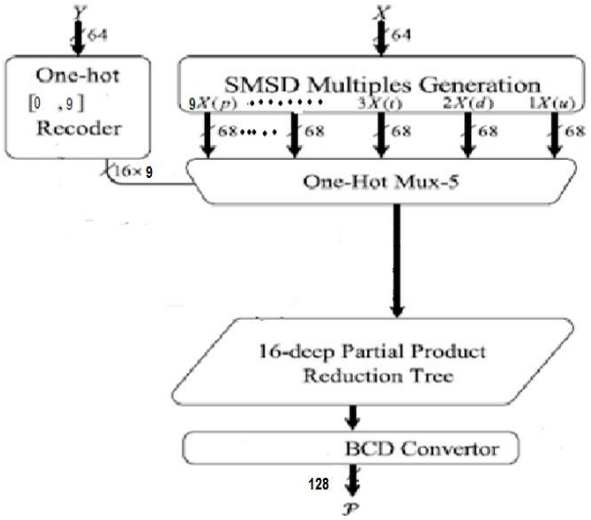


Fig 1: Radix-10 multiplier.

. However, the work of recodes both the multiplier and multiplicand to sign magnitude signed digit (SMSD) representation and uses a more efficient 3-b by 3-b PPG. Nevertheless, following a long standing practice, most PPG schemes use pre computed multiples of multiplicand X (or X multiples). Pre computation 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 IPPs are generated, such as 3X = (2X, X), 7X = (5X, 2X), or 9X = (5X, 4X).

IV.PROPOSED SYSTEM

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.

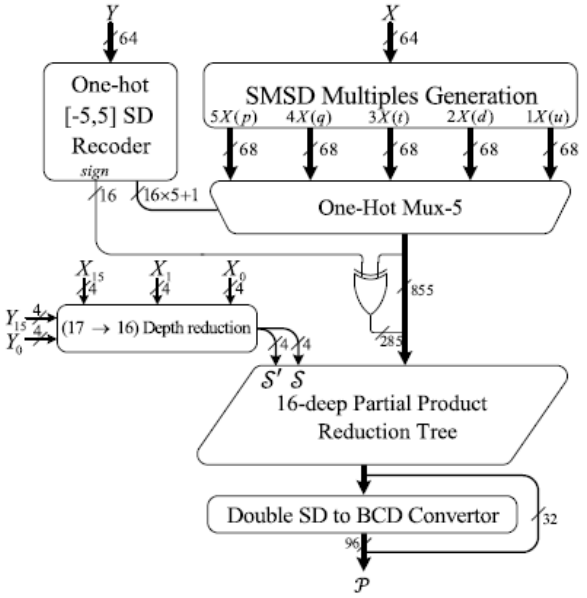


Fig. 2: Block diagram of the proposed multiplier.

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.

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.

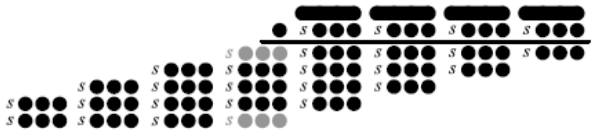


Fig. 3: Normal organization of IPPs.

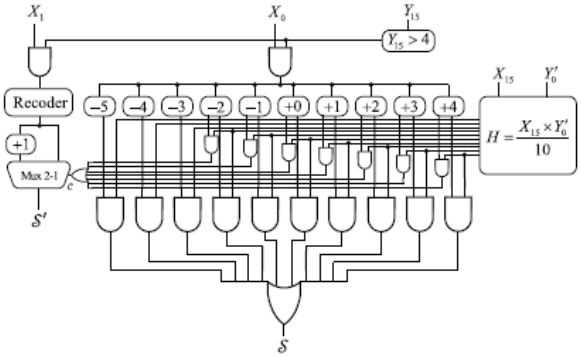


Fig. 4: Required circuit for (17 →16) depth reduction.

Partial Product Reduction

The overall PPR for n = 16 is illustrated by Fig. 5, 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.

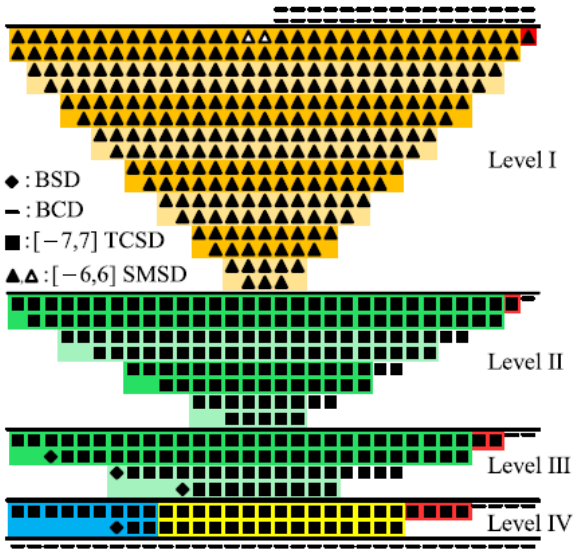


Fig. 5: Overall view of 16 × 16 digit multiplier.

Special 4-in-1 SMSD Adder:

A digit slice of the aforementioned SMSD+SMSD-to-TCSD adder for four different cases corresponding to all possible combinations of the input signs is depicted by Fig. 6(a)–(d) in dot-notation representation. The black and white dots represent posibits and negabits. (A posibit is a normal bit whose arithmetic value equals its logical status, and the arithmetic value of a nega bit with logical status x equals x − 1 [24].)

The sum of two [−6, 6] SMSD digits (e.g., P = sp p2 p1 p0 and Q = sqq2q1q0), and a signed carry in (e.g., Cin) is produced as one [−7, 7] TCSD digit (e.g., S = s3s2s1s0), and a signed carry out (e.g., Cout). This is a two-stage process. In the stage I, the sign bits are applied to the magnitudes, such that a negative sign changes the polarity of magnitude posibits to negabits and inverts their logical states. Subsequently, in the same stage, the bit collection U is decomposed, and the bit collection V is recoded. In the second stage, however, as will be explained shortly, only one 4-b adder takes care of all the four cases, which explains the rationale for designation of the adder.

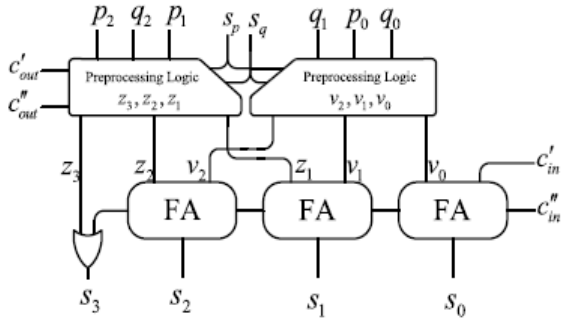


Fig. 6: Digit slice of the 4-in-1 SMSD+SMSD → TCSD adder.

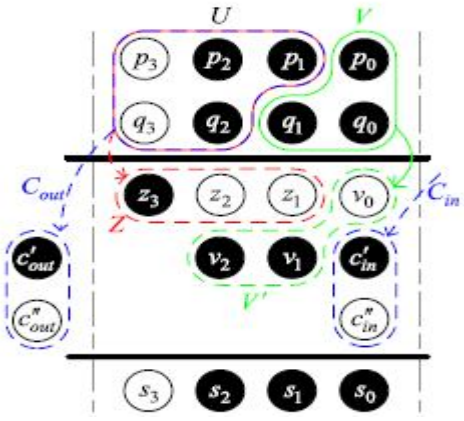


Fig. 7: TCSD adder.

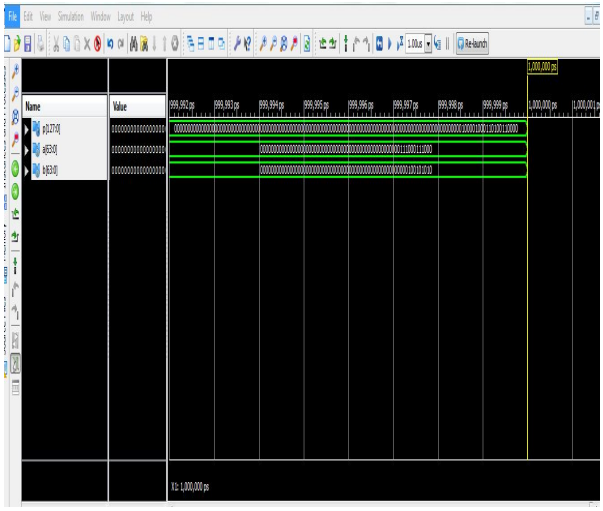
To speed up the latter two steps (i.e., 2:1 reduction and TCSD-to-BCD conversion), the actual BCD product generation of Fig. 5 uses a more efficient method which is described below. The final 2:1 reduction level that is required for positions 8 to 22 and the subsequent TCSD-to-BCD conversion can be actually augmented as a TCSD + TCSD addition with BCD result.

V. RESULTS

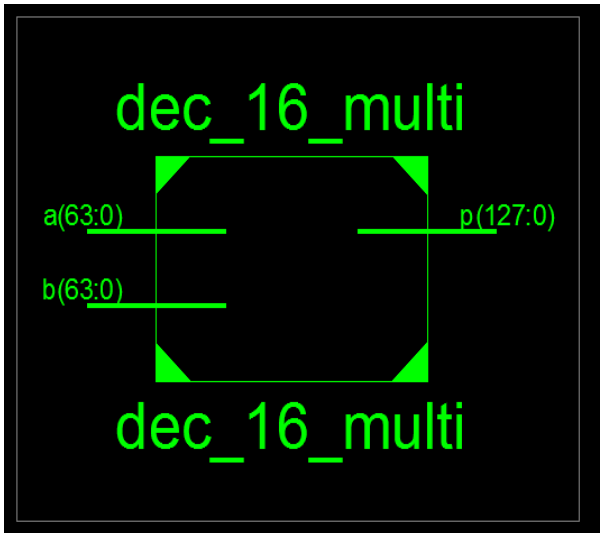
The Verilog HDL Modules have successfully simulated and synthesized using Xilinx ise13.2.

SIMULATION RESULT

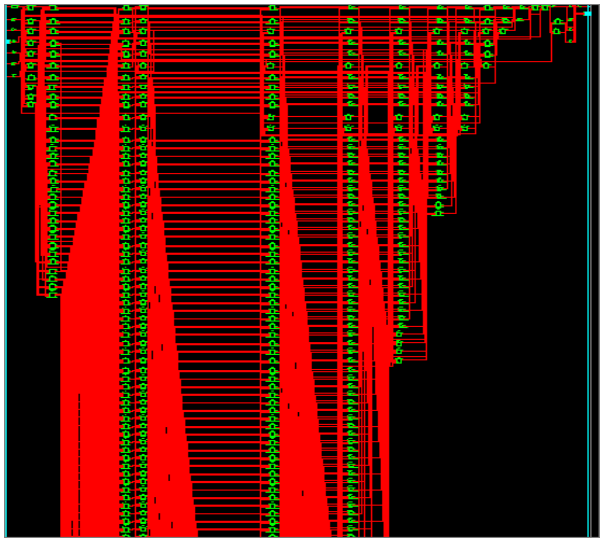
Proposed .



RTL SCHEMATIC:



TECHNOLOGY SCHEMATIC



DESIGN SUMMARY

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	2564	4656	55%
Number of 4 input LUTs	4720	9312	50%
Number of bonded IOBs	256	232	110%
Number of MULT18X18SIOs	16	20	80%

Timing Report:

```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 13854889693 / 128
-------------------------------------------------------------------------
Delay:               23.372ns (Levels of Logic = 104)
  Source:            a<0> (PAD)
  Destination:       p<127> (PAD)

  Data Path: a<0> to p<127>
```

VI. CONCLUSION

We propose a parallel 16 × 16 radix-10 BCD multiplier, where 17 partial products are generated with [−6, 6] SMSD representation. Some innovations of this paper and use of previous techniques, as listed below, have led to marginal 1.5% less area consumption and 10% less power dissipation, at 4.8-ns latency, with respect to the fastest previous work [4]. The least possible delay for the latter is 4.8 ns, while the proposed design leads the synthesis tool to meet the 4.4-ns time constraint (i.e., 9% faster). In other words, the advantage is that the proposed design can operate in 9% higher frequency and dissipate up to 13% less power with no claim in area improvement.

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 floating point multiplication," IEEE Trans. Comput., vol. 58, no. 7, pp. 902–916, Jul. 2009.

[17] C. Tsen, S. González-Navarro, M. Schulte, B. J. Hickmann, and
K. Compton, "A combined decimal and binary floating-point multiplier,"
in *Proc. 20th IEEE Int. Conf. Appl.-Specific Syst., Archit.*
Process. (ASAP), Jul. 2009, pp. 8–15.

[18] C. Minchola and G. Sutter, "A FPGA IEEE-754-2008 decimal64
floating-point multiplier," in *Proc. Int. Conf. Reconfigurable Comput.*
FPGAs (ReConFig), Dec. 2009, pp. 59–64.