INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 2 JULY 2015

AN IMPLEMENTATION OF N X N PARALLEL DECIMAL MULTIPLIER USING CSA

[1] **J. NAGARAJU,** PG Scholar in Embedded Systems & VLSI Design,
[2]**G. SATHYAPRABHA** Assoc.Professor, ECE Department,
[1]nagaraju0465@gmail.com,
[2]sathyaprabhamtech@gmail.com ,
[1,2] SLC's Institute of Engineering and Technology, Hayathnagar Rangareddy,Telangana

Abstract: This paper introduces two novel architectures for parallel decimal multipliers. Our multipliers are based on a new algorithm for decimal carry–save multioperand addition that uses a novel BCD–4221 recoding for decimal digits. It significantly improves the area and latency of the partial product reduction tree with respect to previous proposals. Decimal floating-point multiplication is important in many commercial applications including banking, tax calculation, currency conversion, and other financial areas. The novelty of the design is that it is the first parallel decimal floating-point multiplier offering low latency and high throughput. This design is based on a previously published parallel fixed-point decimal multiplier which uses alternate decimal digit encodings to reduce area and delay. The corresponding hardware algorithms are normally composed of three steps: partial product generation (PPG), partial product reduction (PPR), and final carry propagating addition. In order to improve the speed of parallel decimal multiplication, we presenta new PPG method, fine tune the PPR method of one of the full solutions and the final addition scheme of the other; thus assembling a new full solution. Logical Effort analysis and 0.13 μm synthesis show at least 13% speed advantage, but at a cost of at most 36% additional area consumption.

Keywords: Decimal computer arithmetic, parallel decimal multiplication, partial product generation and reduction,
Decimal carry-save addition.

I. INTRODUCTION

Hardware implementations of decimal arithmetic units have recently gained importance because they provide higher accuracy in financial, commercial, scientific, and internet based applications. One reason is the need for precise floating-point representation of many decimal values (e.g. 0.5) that do not have an exact binary representation. The revision of the IEEE 754 Standard for Floating-Point Arithmetic (IEEE 754-2008) incorporates specifications for DFP arithmetic that can be implemented in software, hardware, or in a combination of both. The computer arithmetic literature includes articles on decimal arithmetic hardware such as BCD adders and multioper and BCD adders, sequential BCD multipliers and dividers. However, some single arithmetic operations (e.g., division or square root), and also function evaluation circuits (e.g., radix-10 exponentiation or logarithm), are often implemented by a sequence of simpler operations including several multiplications. Therefore, hardware implementation of such operations and functions would call for high-speed parallel decimal multiplication. Such multiplication schemes are generally implemented as a sequence of three steps :partial product generation (PPG), partial product reduction (PPR), and the final carry - propagating addition. Parallel binary multipliers are used extensively in most of the binary floating point units for high performance. However, decimal multiplication is more difficult to implement due to the complexity in the generation of multiplicand multiples and the inefficiency of representing decimal values in system based on binary signals. These issues complicate the generation and reduction of partial products. Several different parallel decimal multiplier architectures are proposed in [3], which use new techniques for partial product generation and reduction. Furthermore, some of these architectures were extended to support binary multiplication. Some concepts of [3] were applied in [6] to design decimal 4:2 compressor trees. All of the previous designs are combinational fixed point architectures. A pipelined IEEE 754-2008 compliant DFP multiplier based on architecture from [3] was presented in [7].

MULTIPLIERS

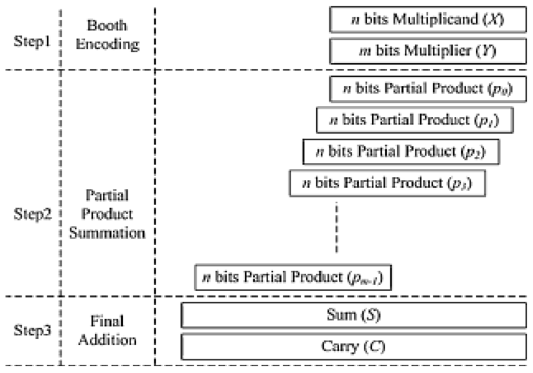
Step1
Booth Encoding
n bits Multiplicand (X)
m bits Multiplier (Y)
n bits Partial Product (p0)
n bits Partial Product (p1)
n bits Partial Product (p2)
n bits Partial Product (p3)
Step2
Partial Product Summation
n bits Partial Product (pm-1)
Step3
Final Addition
Sum (S)
Carry (C)

Fig. 1. Basic arithmetic steps of multiplication

Multiplication is implemented by accumulation of partial products, each of which is conceptually produced via multiplying the whole multiplicand by a weighted digit of the multiplier. Decimal multiplication, as in binary, may be accomplished sequentially, in parallel, or by a semi-parallel approach [22]. Following the recent trend [18 and 20], we focus on fully parallel BCD multiplication, which is generally a 3-step process:

a) Partial product generation (PPG): A partial product $W_I = Y^i$ x X, as in Fig. 1, is generated via multiplying the multiplicand X by a multiplier digit Y^i, where the positional weight of W_I and Y^i is equal to the ith power of the radix (e.g., 10i). One could think of BCD multipliers producing all the partial products in parallel by a matrix of BCD digit multipliers [22], or through selection of pre computed multiples [23, 11, and 18]. In the case of [20], 4-2-2-1 decimal encoding is used to represent the partial products.

b) Partial product reduction (PPR): This is the major step in parallel multiplication, which is a special case of the more general problem of multi-operand decimal addition [9 and 10]. A parallel solution for this problem uses a reduction tree that leads to a redundant intermediate representation of product (e.g., two equally-weighted BCD numbers [20]).

c) Final product computation: The redundant product computed in Step b) is converted to the final BCD product (e.g., via a BCD adder [8])

III. DECIMAL PARALLEL MULTIPLIER

In this section, we present a general overview of the Radix-10 architecture for d-digit (4d-bit) BCD decimal fixed-point parallel multiplication. This design is based on the techniques for partial product generation and reduction detailed in Sections 4 and 5, respectively. The code (4221) and (5211) is used instead of BCD to represent the partial product is the main feature of this architecture. This improves the reduction of decimal partial product with respect to other proposals, in terms of latency and area is expected.

SD Radix-10 Architecture

The architecture of the d-digit SD radix -10 multiplier is shown in Fig.1. The multiplier consists of the following stages : Generation of decimal partial products coded in (4221) (generation of multiplicand multiples and SD radix-10 encoding of the multiplier), reduction of partial products, and a final BCD carry-propagate addition.

The generation of the d+ 1 partial products is performed by an encoding of the multiplier into d SD radix-10 digits and an additional leading bit as described in Section 4.1. Each SD radix-10 digit controls a level of 5:1 muxes, which selects a positive multiplicand multiple (0, X, 2X, 3X, 4X, 5X) coded in (4221). The generation of these multiples is detailed in Section 4.3. To obtain each partial product, a level of XOR gates inverts the output bits of the 5:1 muxes when the sign of the corresponding SD radix-10 digit is negative.

The final product is a 2d-digit BCD word given by P=2H +S. Before being added, S and H need to be processed. S is recoded from (4221) to BCD excess-6(BCD value plus 6, which requires practically the same logical complexity as a recoding to BCD). The H×2 multiplication is performed in parallel with the recoding of S. This ×2 block uses a (4221) to (5421) digit recoder (see Section 4.4) and a 1-bit wired left shift to obtain the operand 2H coded in BCD.

For the final BCD carry-propagate addition, we use a quaternary tree (Q-T) adder based on conditional speculative decimal addition. It has low latency and requires less hardware than other alternatives.

IV. Decimal Partial Product Generation

Negative numbers are represented in radix-10 complement and are implemented by doing the 9’s complement of the BCD digit and adding a 1 in the least significant position. To reduce the delay of the additions, we perform the addition xyi =xyHi+xyLi carry save and use a radix-10 carry-save representation of the partial products. That is, the addition to produce a partial product results in

x :						1	9	6	3	×
y :						8	1	4	5	=
xy_0 :	$5x$					9	8	1	5	
	0					0	0	0	0	
xy_1 :	$5x$				9	8	1	5		
	$-x$			-	1	9	6	3		
xy_2 :	x			1	9	6	3			
	0			0	0	0	0			
xy_3 :	$10x$	1	9	6	3	0				
	$-2x$	-	3	9	2	6				
		1	5	9	8	8	6	3	5	

Example Of Multiplication(4×4digits)

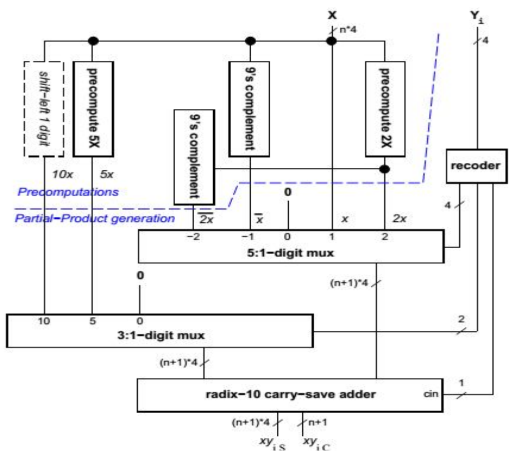

Fig. 2. Partial product generation.

One of the accompanying challenges in designing ICs for portable electrical devices is lowering down the power consumption to prolong the operating time on the basis of given limited energy supply from batteries. Owing to the vigorous development of the wireless infrastructure and the personal electronic devices like video mobile phones, mobile TV sets, PDAs, etc., multimedia and DSP applications have been adopted in wireless environments. Increasing demands of high speed data signal processing motivated the researchers to seek fastest processors. The multiplier and multiplier-and-accumulator (MAC) [1] are the building blocks of the processor and has a great impact on the speed of the processor. MAC is the necessary element of the digital signal and image/audio processing system such as filtering, convolution and inner products hence high speed is crucial to develop for real processing applications. Many researchers have attempted in designing MAC for high computational performance and low power consumption. High throughput MAC is always a key factor to achieve high performance digital signal processing applications for real time signal processing applications. Since the multiplier requires the longest delay among the basic operation in digital system, the critical path is limited by the multiplier. Multiplier basically consists of three operational steps: Booth Encoder, Partial product reduction network (Wallace Tree) and final adder. For high speed multiplication, Modified Booth Algorithm (MBA) [4] is most commonly used, in which partial product is generated from Multiplicand (X) and Multiplier (Y) .Booth multiplication allows for the smaller ,faster multiplication circuits through encoding the signed bits to 2’s complement which is also the standard technique in chip design and provide substantial improvement by reducing the partial products. Although the partial products are further reduced by using higher radix (4, 8, 16, 32) Booth Encoder which increases complexity and improves the performance[1].

Generation of Multiplicand Multiples

All the required decimal multiplicand multiples, except the 3X multiple, are obtained in a few levels of combinational logic using different digit recoders and performing different fixed m-bit left shifts (Lmshift) in the bit-vector representation of operands. Fig. 3 shows the block diagram for the generation of the positive multiplicand multiples {X, 2X, 3X, 4X, 5X} for the SD radix-10 recoding.

All these multiples are coded in (4221). The X BCD multiplicand is easily recoded to (4221) using the logical expressions.

Multiple 2X: Each BCD digit is first recoded to the (5421) decimal coding shown in Table 1 (the mapping is unique). An L1 shift is performed to the recoded multiplicand, obtaining the 2X multiple in BCD.

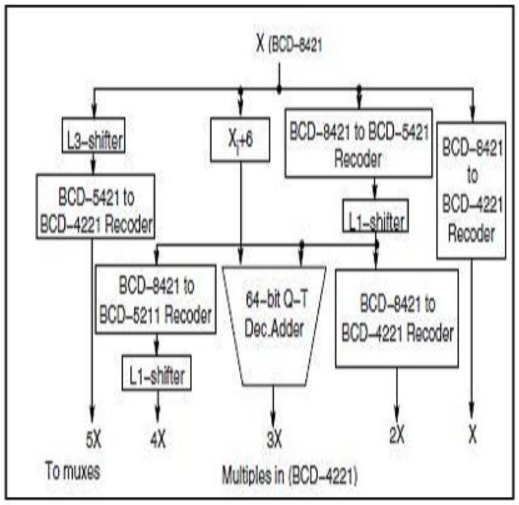
X (BCD-8421)
L3-shifter
X+6
BCD-8421 to BCD-5421 Recoder
BCD-8421 to BCD-4221 Recoder
BCD-5421 to BCD-4221 Recoder
L1-shifter
BCD-8421 to BCD-5211 Recoder
64-bit Q-T Dec.Adder
BCD-8421 to BCD-4221 Recoder
L1-shifter
5X
4X
3X
2X
X
To muxes
Multiples in (BCD-4221)

Fig 3 .Generation of multiplicand multiples for SD radix–10

Multiple 4X: It is obtained as 2X × 2, where the 2X multiple is coded in (4221). The second ×2 operation is implemented as a digit recoding from (4221) to code (5211), followed by an L1shift. The design of the (4221) to (5211) digit recoders is described in Section 4.3. The ×2 operation, with input operands coded in (4221) or (5211), is also implemented in the decimal CSA trees used for partial product reduction.

Multiple 5X:It is obtained by a simple L3shift of the (4221) recoded multiplicand, with resultant digits coded in (5211). Then, a digit recoding from (5211) to (4221) is performed (see Section 4.3). Fig. 4 shows an example of this operation.

Multiple 3X: It is evaluated by a carry - propagate addition of BCD multiples X and 2X in a d-digit BCD adder. The latency of the partial product generation for the SD radix-10 scheme is constrained by the generation of 3X.

Method for Decimal Carry-Save Addition

Fig.4 shows the implementation of a decimal 3:2 CSA for digits coded in (4221) using a 4-bit binary 3:2 CSA. The weight bits in Fig. 9a are placed in brackets above each bit column. The 4-bit binary 3:2 CSA adds three decimal digits (Ai, Bi, Ci), coded in (4221), and produces a decimal sum digit (Si) and a carry digit Hi Coded in (4221), such that

$$Ai + Bi + Ci = Si + 2 \times Hi.$$

In order to obtain (2H)i, Hi is first recoded to Wi using the (4221) to (5211s) digit recorder. The output of the digit recorder (Wi) is then left shifted by 1 bit position (L1shift[Wi]). A decimal carry output wi,3 is passed to the next significant digit position, while a decimal carry in wi-1,3 comes from the previous. Since the recoder is placed in the carry path, a full adder implementation with a fast carry output.

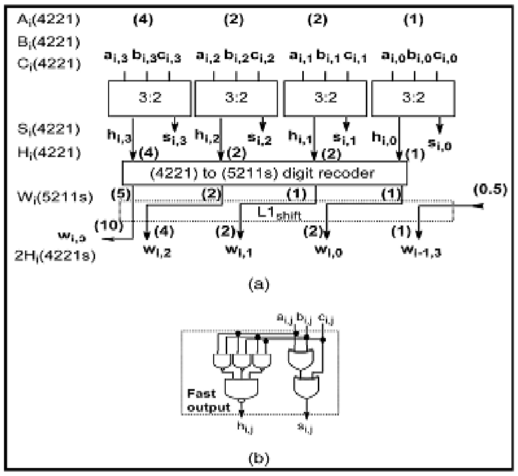

Fig.4 Proposed decimal 3:2 CSA for operands coded in (4221). (a) Decimal digit (4-bit) 3:2 CSA (b) Full Adder

V.Simulation Results

The decimal multiplier design was simulated in Modelsim 6.6c and synthesized

Volume 4, Issue 2 JULY 2015

using Xilinx ISE 12.2i which was mapped on to Virtex-6 FPGA. The simulation results of 16-bit decimal multiplier are shown in fig 5.

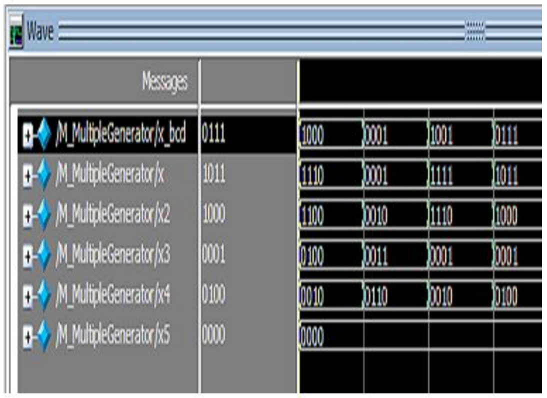
Wave
Messages
/M_MultipleGenerator/x_bcd 0111
1000 0001 1001 0111
/M_MultipleGenerator/x 1011
1110 0001 1111 1011
/M_MultipleGenerator/x2 1000
1100 0010 1110 1000
/M_MultipleGenerator/x3 0001
0100 0011 0001 0001
/M_MultipleGenerator/x4 0100
0010 0110 0010 0100
/M_MultipleGenerator/x5 0000
0000

Fig 5 Waveforms of Multiplicand multiples

VI. CONCLUSION

In this paper, we have presented one technique to implement decimal parallel multiplication in hardware. We propose the SD encoding for the multiplier that lead to fast parallel and simple generation of partial products. We have developed a decimal carry-save algorithm based on (4221) and (5211) decimal encoding for partial product reduction. We have proposed architecture for decimal SD radix-10 multiplication.

VII. REFERENCES

[1] IEEE standard for floating–point arithmetic. IEEE Standards Committee, Oct. 2006. Available thttp://754r.ucbtest.org/drafts/754r.pdf.
[2] F. Y. Busaba, T. Slegel, S. Carlough, C. Krygowski, and J. G.Rell. The design of the fixed point unit for the z990 microprocessor. InProc. ACM Great Lakes14th Symposium on VLSI, pages 364–367, Apr. 2004.
[3] M. F. Cowlishaw. Decimal floating-point: Algorism for computers. InProc. IEEE16th Symposium on Computer Arithmetic, pages 104–111, July 2003.
[4] M. A. Erle and M. J. Schulte. Decimal multiplication via carry-save addition. InProc. IEEE Int’l Conference on Application-Specific Systems, Architectures, and Processors, pages 348–358, June 2003.
[5] M. A. Erle, E. M. Schwarz, and M. J. Schulte. Decimal multiplication with efficient partial product generation. In Proc. IEEE17th Symposium on Computer Arithmetic, pages 21–28, June 2005.
[6] R. D. Kenney and M. J. Schulte. High-speed multioperand decimal adders.IEEE Trans. on Computers, 54(8):953–963, Aug. 2005.
[7] R. D. Kenney, M. J. Schulte, and M. A. Erle. High-frequency decimal multiplier. InProc. IEEE Int’l Conference on Computer Design: VLSI in Computers and Processors, pages 26–29, Oct. 2004.
[8] T. Lang and A. Nannarelli. A radix-10 combinational multiplier. InProc.40th Asilomar Conference on Signals, Systems, and Computers, pages 313–317, Oct. 2006.
[9] R. H. Larson. High-speed multiply using four input carrysave adder. IBM Tech. Disclosure Bulletin, 16(7):2053–2054, Dec. 1973.
[10] N. Ohkubo and M. Suzuki. A 4.4 ns CMOS 54x54–bit multiplier using pass-transistor multiplexer. IEEE Journal of Solid State Circuits, 30(3):251–256, Mar. 1995.
[11] T. Ohtsuki. Apparatus for decimal multiplication. U.S.Patent No. 4,677,583, June 1987.
[12] M. Schmookler and A. Weinberger. High speed decimal addition.IEEE Trans. on Computers, C-20(8):862–866, Aug.1971.