Design and Evaluation of Decimal Array Multipliers

Saeid Gorgin[1], Ghassem Jaberipur[1,2], Behrooz Parhami[3]

[1] Dept. of Electrical & Computer Engr., Shahid Beheshti Univ., Tehran, Iran; gorgin@sbu.ac.ir
[2] School of Computer Sci., Inst. for Research in Fundamental Sci. (IPM), Tehran, Iran; jaberipur@sbu.ac.ir
[3] Dept. of Electrical & Computer Engr. Univ. of California, Santa Barbara, USA; parhami@ece.ucsb.edu

Abstract

Hardware support for decimal arithmetic has become an important focal point, both in the research arena and in commercial processor developments. Like their binary counterparts, decimal multipliers can be designed in a variety of ways, offering area and speed trade-offs. Pipelined array multipliers support high throughput, making them attractive in multiply-intensive applications. We propose two different architectures for decimal array multipliers based on (1) precomputed multiples and (2) decimal digit-multipliers. We compare the VLSI area and delay parameters of the resulting array multiplier designs with each other and with those of binary array multipliers covering the same range of inputs.

Keywords—Array multiplier, Binary-coded decimal; Computer arithmetic; Decimal calculation; Pipelining.

1. Introduction

Several decades of binary computation may well prove to be a historical curiosity, sandwiched between centuries of decimal arithmetic in mechanical calculating devices and the omnipotent machines of the future that will not be limited by cost considerations of early digital computers or today's binary-friendly electronic circuits. Switching to all-decimal arithmetic in computers will eliminate one of the key sources of problems in numerical computation: the conversion and reconversion errors. It is certainly a more faithful way of dealing with arithmetic, which is decimal in the world outside computers.

Limited hardware support for decimal arithmetic is already in place [1-5], following decades of slow software simulation on binary computers [6]. It is now quite feasible to replace sequential decimal multipliers [7-9] by much faster parallel realizations [10-12].

Pipelining of computer arithmetic circuits is a longstanding practice for achieving high throughput. In this regard, array multipliers are not only particularly suitable for pipelined computations, but they also lead to enhanced circuit regularity. This regularity, when combined with shorter interconnects, translates to greatly improved performance [13].

In this paper, we offer two decimal array multiplier designs for conventional binary-coded-decimal (BCD) operands: one is based on selecting the partial products from among precomputed "easy" multiples of the multiplicand [14], and the other on generating the partial products via BCD digit-multiplier cells [15]. Table I lists symbols and abbreviations used throughout this paper for easy reference.

Table I Symbols and abbreviations used

Symbol/Term	Meaning
PPG	Partial product generation
BPP	BCD partial product
BCD	Binary-coded decimal
BDM	BCD digit-multiplier
BCD-FA	BCD full adder
PPR	Partial product reduction
●	Decimal carry in [0, 1]
▬	BCD digit in [0, 9]
X^i	BCD digit X of weight 10^i
P_j^i	Digit of weight 10^i in the jth BPP

Here is a roadmap for the rest of the paper. Brief descriptions of the two partial product generation (PPG) methods and the related basic reduction cells for the corresponding array multipliers are provided in Section 2. Section 3 is devoted to two architectures for decimal array multipliers. Area and delay comparisons between the two architectures and with binary array multipliers that cover the same range of input values are offered in Section 4. Finally, conclusions are drawn in Section 5.

2. Decimal Partial Product Generation

A $k \times k$ decimal multiplier computes the $2k$-digit product by accumulating k BCD partial products (BPPs). Each BPP constitutes a multiple of the multiplicand, where the multiplication factor ranges from 0 to 9, corresponding to a decimal digit. A particular BPP can be generated via k BCD digit-multipliers (BDMs) or, more commonly, via precomputation of the required multiples and selection, using a multiplexer (mux). A mux is shown as a trapezoidal box in our logic diagrams.

Figure 1 contains a high-level description of a BDM, where the heavy horizontal bars represent BCD digits. Functionally, a BDM multiplies its two 4-bit BCD inputs and provides the resulting two-digit product at output. The design details are available elsewhere [15]. Figure 2 depicts 4 × 4 decimal PPG, where each of the four BPPs, generated via four BDMs, is represented as a doubled-up BCD number, composed of a 4-digit BCD number with the appropriate decimal alignment and a second left-shifted 4-digit BCD number.

Fig. 1 Decimal digit-multiplier (BDM).

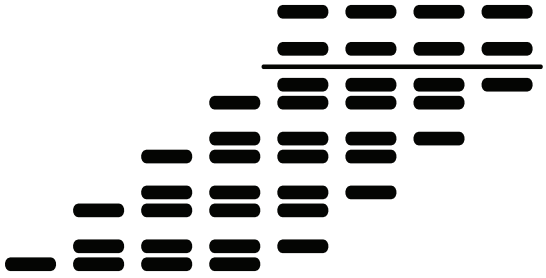

Fig. 2 A scheme for 4 × 4 PPG via 16 BDMs.

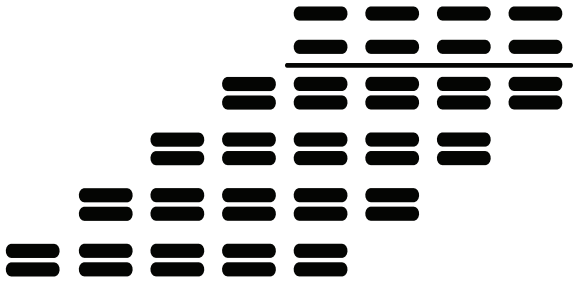

Fig. 3 A 4 × 4 PPG scheme using precomputed multiples.

The method of precomputed multiples is based on directly computing some of the required multiples (e.g., 2, 4 and 5 [Rich55]) as single BCD numbers and forming the others as doubled-up BCD numbers (*viz.*, 3 = 2 + 1, 6 = 5 + 1, 7 = 5 + 2, 8 = 4 + 4, 9 = 5 + 4). Figure 3 depicts this PPG method as applied to 4 × 4 BCD multiplication. Note that for the multiples 0, 1, 2, 4, and 5, an all-0s entry is used as the second component of the associated doubled-up BCD number for the sake of uniformity. Other sets of easy multiples have also been utilized in the design of parallel BCD multipliers [10], [12].

Similar to the practice for binary tree and array multipliers, a common way of computing the desired decimal product is to first compress the partial products matrix into two BCD numbers and then employ a conventional BCD adder to finish the process. The basic partial product reduction (PPR) cell, depicted in Fig. 4a, is a BCD full adder (BCD-FA) [16]. However, just as utilizing (4; 2)-compressors leads to binary multipliers with more regular layouts [13], a merged realization of two BCD-FAs, acting as a BCD (3; 1)-compressor (Fig. 4b), produces more regular decimal designs.

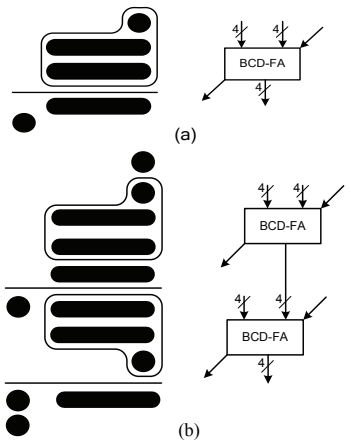

Fig. 4 (a) BCD-FA; (b) BCD (3; 1)-compressor.

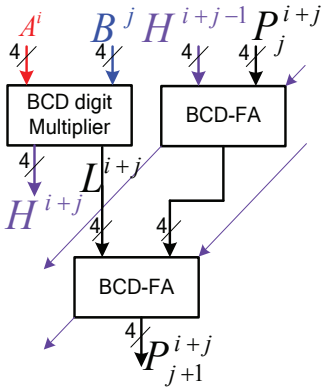

Fig. 5 Combined PPG/PPR cell.

The BDM approach of Fig. 2 leads to slower PPG than that of Fig. 3 [15]. However, the former yields greater regularity for efficient VLSI realization. Figure 5 depicts a combined PPG/PPR cell composed of one BDM and two BCD-FAs, representing Eqns. 1 and 2, where a superscript denotes the power-of-10 weight of a particular decimal position.

$$A^i \times B^j = 10 \times H^{i+j} + L^{i+j} \quad (1)$$

$$P_{j+1}^{i+j} = P_j^{i+j} + H^{i+j-1} + L^{i+j} \quad (P_0^0 = L^0) \quad (2)$$

The inputs A^i and B^j are the corresponding digits of the multiplicand and multiplier, whose product is represented by a pair of BCD digits H (high) and L (low). The output P_j^{i+j} constitutes the $(i + j)$th digit of the jth accumulated BPP. The same arrangement would not be possible with the PPG approach of Fig. 3.

3. Decimal Array Multiplier Architectures

Two decimal array multiplier architectures, based on the basic cells of Figs. 4b and 5, are depicted in Figs. 6 and 7, respectively.

Each mux block in Fig. 6 selects the two components of a doubled-up BCD multiple from among precomputed multiples based on the appropriate BCD digit of the multiplier. The boundary reduction cells in Fig. 6 are somewhat simpler than those in the middle. For example, the top and left cells are single BCD-FAs (the top five with no carry-in) and the input/output relationship of the four bottom cells is shown in Fig. 8, where the required reduction logic is composed of a binary half-adder and a BCD-FA. Similarly, the boundary cells of Fig. 7 are simpler than those in the middle.

The bottom cells in both Figs. 6 and 7 constitute a ripple-carry BCD adder that would be replaced by a fast adder in a parallel multiplier [17]. In a pipelined realization, however, use of a fast adder may not be helpful and may prove detrimental to both performance and cost-effectiveness. In pipelined realizations of the designs in Figs. 6 and 7, we can incorporate two cascaded bottom cells within a single pipeline stage. The latency of the resulting merged cell would be slightly less than that of one of the nonboundary cells.

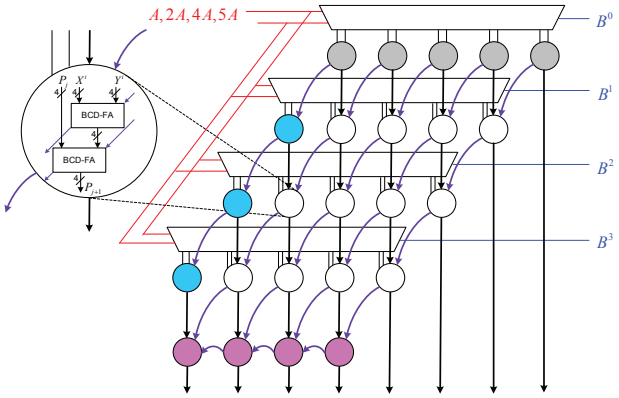

Fig. 6 A 4 × 4 array multiplier with precomputed multiples.

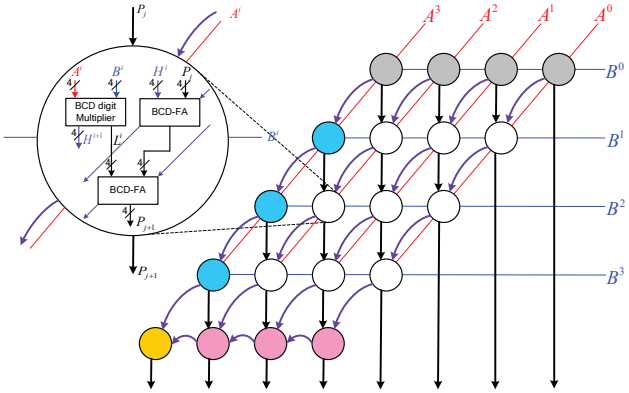

Fig. 7 A 4 × 4 array multiplier with BDMs.

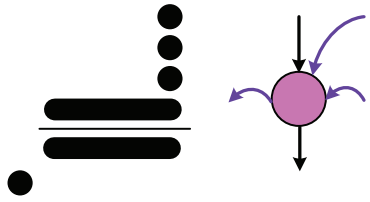

Fig. 8 The simplified bottom boundary cells of Fig. 6.

4. Delay and Area Comparisons

The architecture of Fig. 7 is readily seen to be more regular than that of Fig. 6; it is also more amenable to pipelined realization. Table II shows assumed component delays in terms of fan-out-of-4 NOT gates (FO4) and layout areas in units of 2-input NAND-gate (NAND2), based on Logical Effort analysis [18]. All the basic cells required for our two proposed architectures, and those needed for a similar binary array multiplier used for comparison, are listed in Table II. The PPG entries constitute the collection of components that would be responsible for generating the precomputed multiples, decoding the multiplier digit, and selecting the BPP (in doubled-up BCD form) in Fig. 6.

The equations that follow represent the area (A) and delay (D) of a k-bit BCD array multiplier based on architectures of Figs. 6 and 7. Because our focus is on pipelined realization, the delay figures are derived by identifying the slowest pipeline stage.

$$A_{\text{EasyPPG}} = k \times \text{Generator} + k \times \text{Decoder} + (k^2 + k) \times \text{Selector}$$

$$D_{\text{EasyPPG}} = D_{\text{Generator}} + D_{\text{Selector}}$$

$$A_{\text{Fig.6}} = A_{\text{EasyPPG}} + (2k^2 + k) \times \text{BCDFA} + (k - 1) \times \text{HA}$$

$$D_{\text{Fig.6}} = D_{\text{EasyPPG}} + 2 \times D_{\text{BCD.FA}} + D_{\text{Latch}}$$

$$A_{\text{Fig.7}} = k^2 \times \text{BDM} + (2k^2 - 2k + 1) \times \text{BCDFA}$$

$$D_{\text{Fig.7}} = D_{\text{BDM}} + D_{\text{BCDFA}} + D_{\text{Latch}}$$

Table III shows our derived delay and area parameters for 3-digit pipelined decimal array multipliers (i.e., for $k = 3$) based on the equations just provided and Table II. The corresponding parameters for a 10-bit binary array multiplier, which covers almost the same range of input values (1024 different values in binary, versus 1000 in decimal), are also supplied for comparison.

Table II Delay and area parameters for basic cells

		Delay (FO4)	Area (NAND2)
Half-adder (HA)		1.35	5.
Full adder (FA)		2.63	13.
Latch		1.44	4.88
BCD-FA		4.16	98.
BDM		8.90	166.12
PPG	**Generator**	3.91	66.37
	Decoder	1.35	36.
	Selector	1.44	31.5

Table III Array multiplier delay and area comparisons

	Delay (FO4)	Delay ratio	Area (NAND2)	Area ratio
Binary	4.94	1.00	1228	1.00
Decimal, Fig. 6	15.11	3.05	2753	2.24
Decimal, Fig. 7	14.50	2.93	2769	2.25

The factor of about 3 delay penalty for decimal array multiplication compared with the binary version is rather discouraging. However, there are two redeeming factors. First, some of the performance loss due to decimal arithmetic is regained from the elimination of conversion and reconversion overheads. Second, in some applications, the gain in accuracy may justify the reduced speed. For example, applications that currently tolerate the excessive overhead of software implementations (with or without partial hardware support) will certainly have no problem with a factor-of-3 throughput reduction. Finally, this is only a first attempt at designing decimal array multipliers; better designs may emerge over time.

Note that the "decimal" penalties discussed above are assessed with respect to pipelined binary array multiplication. The throughput of a pipelined decimal array multiplier is considerably better than that of other decimal multiplication techniques. A quantitative assessment of this advantage is left to future work.

5. Conclusions

We have proposed decimal array multiplier designs based on two different methods for partial product generation. The design utilizing BCD digit-multipliers shows a 4% advantage in the pipeline clock cycle; it is also more regular and hence better-suited to VLSI realization. The regularity and higher clock frequency are achieved at negligible area cost (≈ 0.5%). We compared the performance of 3-digit decimal array multipliers, accepting inputs in [0, 999], with each other and with that of a 10-bit binary array multiplier, having input operands in [0, 1023]. Not surprisingly, the area and delay penalties are rather significant, with high throughput being the main advantage compared with other decimal multipliers.

Further research can be undertaken to improve the performance and reduce the decimal-arithmetic penalty. We would like to examine the use of alternate reduction cells, such as those based on decimal encodings other than standard BCD [11]. Fully redundant decimal array multipliers can also be envisioned [19].

References

[1] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough, "The IBM z900 Decimal Arithmetic Unit," *Proc. 35th Asilomar Conf. Signals, Systems, and Computers*, Vol. 2, pp. 1335–1339, 2001.

[2] S. Shankland, "IBM's POWER6 Gets Help with Math, Multimedia," *ZDNet News*, October 2006.

[3] J. Friedrich, *et al.*, "Design of the POWER6 Microprocessor," *Proc. Int'l Solid-State Circuits Conf.*, pp. 96–97, 2007.

[4] C. F. Webb, "IBM z10: The Next-Generation Mainframe Microprocessor," *IEEE Micro*, Vol. 28, No. 2, pp. 19–29, 2008.

[5] E. M. Schwarz, J. Kapernick, and M. Cowlishaw, "Decimal Floating-Point Support on the IBM z10 Processor," *IBM J. Research and Development*, Vol. 53, No. 1, 2009.

[6] M. F. Cowlishaw, "Decimal Floating-Point: Algorism for Computers," *Proc. 16th IEEE Symp. Computer Arithmetic*, pp. 104–111, 2003.

[7] M. A. Erle and M. J. Schulte, "Decimal Multiplication via Carry-Save Addition," *Proc. Conf. Application-Specific Systems, Architectures, and Processors*, pp. 348–358, 2003.

[8] R. D. Kenney, M. J. Schulte, and M. A. Erle, "A High-Frequency Decimal Multiplier," *Proc. IEEE Int'l Conf. Computer Design*, pp. 26–29, 2004.

[9] M. A. Erle, E. M. Schwartz, and M. J. Schulte, "Decimal Multiplication with Efficient Partial Product Generation," *Proc. 17th IEEE Symp. Computer Arithmetic*, pp. 21–28, 2005.

[10] T. Lang and A. Nannarelli, "A Radix-10 Combinational Multiplier," *Proc. 40th Asilomar Conf. Signals, Systems, & Computers*, pp. 313–317, 2006.

[11] A. Vazquez, E. Antelo, and P. Montuschi, "A New Family of High-Performance Parallel Decimal Multipliers," *Proc. 18th IEEE Symp. Computer Arithmetic*, pp. 195–204, 2007.

[12] G. Jaberipur and A. Kaivani, "Improving the Speed of Parallel Decimal Multiplication," *IEEE Trans. Computers*, to appear (doi: 10.1109/TC.2009.110).

[13] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford, 2nd ed., 2010.

[14] R. K. Richards, *Arithmetic Operations in Digital Computers*, Van Nostrand, 1955.

[15] G. Jaberipur and A. Kaivani, "Binary-Coded Decimal Digit Multipliers," *IET Computers & Digital Techniques*, Vol. 1, No. 4, pp. 377–381, 2007.

[16] M. Schmookler and A. Weinberger, "High Speed Decimal Addition," *IEEE Trans. Computers*, Vol. 20, No. 8, pp. 862–866, 1971.

[17] A. Vazquez and E. Antelo, "Conditional Speculative Decimal Addition," *Proc. 7th Conf. Real Numbers and Computers*, pp. 47–57, 2006.

[18] I. F. Sutherland, R. F. Sproull, and D. F. Harris, *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufmann, 1999.

[19] S. Gorgin and G. Jaberipur, "Fully Redundant Decimal Arithmetic," *Proc. 19th IEEE Symp. Computer Arithmetic,* pp. 145-152, 2009.

Note: This paper was produced in final form on 2009/09/09