IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 3, Issue 1, Feb-Mar, 2015
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

New Low Energy/Power Parallel Decimal Multiplier on FPGA and ASIC

Amin Malekpour[1], Sadegh Nejatzadeh[2], Farzaneh Aghajari[3], Saeid Malekpour[4]

[1]Computer Engineering, Sharif University of Technology, Tehran, Iran

[2]Visiting professor of Computer Engineering, Azad University of Yasuj, Yasuj, Iran

[3]Computer Engineering, Payam-noor University of Shiraz, Shiraz, Iran

[4]Visiting professor of Computer Engineering, Azad University of Yasuj, Yasuj, Iran

Abstract

Decimal multiplication is believed as a frequent operation with inherent complexity in implementation which is becoming more and more important every day. Most commercial, financial, scientific, and internet-based applications need their data to be precise, while binary number system cannot ensure preciseness. In this paper a new architecture for parallel decimal multiplication is presented. Evaluation results for 16-digit operands show that the proposed architecture has 8.7% less energy/power consumption and improves area on ASIC and has 8.1% less energy consumption and improves power, delay and area on FPGA.

Keywords: *computer arithmetic, decimal multiplier, ASIC, FPGA, synthesis.*

1. Introduction

Due to the dramatic advances in VLSI technology, hardware realization of many complex functions is possible now [6]. Since there is an intensive request for decimal computations in various critical applications such as scientific, commercial, financial and Internet-based applications [4], design and implementation of decimal arithmetic algorithms draws attention of scientists and practitioners.

On account of the importance of decimal arithmetic, IEEE has been added new feature of decimal to IEEE 754 Standard for Floating-Point Arithmetic (IEEE 754-2008) [5].

A disadvantage of binary arithmetic is the lack of preciseness. For example, the value 0.2 cannot be stored in memory precisely because it cannot be converted into binary representing in exact value as in decimal. This drawback can be suffering in calculations containing very small numbers. To storing data in decimal format, conventional binary algorithms should be changed.

Decimal multiplication is one of the widely used algorithms in throughout the arithmetic systems. It is more complicated than binary multiplication because in decimal multiplication, a considerably bigger digit set is being dealt with [11]. Decimal multiplication is a complex, time consuming, and high frequent operation. Therefore, it has been realized by iterative hardware algorithms in some recent announced processors (e.g. IBM POWER6 [8] and IBM z10 [9] and IBM z900 [10]). However, there are several promising points for exploiting of parallel decimal multiplication algorithms in hardware. The hardware complexity of an iterative multiplier is much lower than parallel multiplier. However, since decimal operations are computationally intensive, the lower latency of a parallel multiplier makes it more attractive for decimal computations [7].

Several algorithms have been proposed for decimal multiplications. They mostly iterate through the multiplier and add the corresponding multiplicand digits to a register successively.

The hardware algorithm of decimal multiplication is composed of three steps: partial product generation (PPG), partial product reduction (PPR), and final carry propagating addition. There are various techniques for implementing PPG and PPR steps.

In [2], the architecture of a radix-10 combinational multiplier is presented. The PPG is generated in such a way that only multiples 2 and 5 of the multiplicand are required, and their accumulation is done by using radix-10 CSAs and counters.[2]

In [3], new algorithm for decimal carry-save multi-operand addition that uses a novel BCD-4221 recoding for decimal digits is presented. It also presents some schemes for fast generation of partial products. The recoding of the BCD-8421 multiplier operand into minimally redundant signed-digit radix-10, radix-4 and radix-5 representations using new re-coders, reduces the complexity of partial product generation [3].

In the PPG part of [13], the unsigned multiples of multiplicand X, 2X, 4X, and 5X are used. For the PPR part, two reduction schemes are proposed: delay-optimized and area-optimized. The former scheme uses a novel carry-free adder (ODDS adder) and its restricted input varieties to achieve a VLSI-friendly recursive partial product reduction tree, while the latter scheme employs binary 4:2 compressors augmented with +0/6 correction logic. Finally in last step, for both architectures, a redundant to non-redundant converter is used to convert ODDS products to BCD digits.

In [14], two parallel decimal multipliers are presented which are improved designs of architecture in [3]. In the PPG part of radix-5 design, multiples -X, 2X, -2X, 5X, and 10X are generated. Then a mixed (4221/5211) partial products are generated. In the PPG part of radix-10 design, multiples X, 2X, 3X, 4X, and 5X and d + 1 partial products are generated. In order to reduce these decimal partial products in both designs, decimal adder trees have been used. Finally, both architectures in [14] have used a quaternary tree adder (proposed by the same authors in [15]) for redundant to non-redundant converting

In this paper, the proposed BCD multiplier in [1] which is called Jaberipur multiplier in this paper is implemented and synthesized. Then some changes are driven in its PPG and PPR parts which lead to achieve a new parallel decimal multiplier improving the delay, area and energy/power consumption on FPGA and area and energy/power consumption on ASIC. Practical reports about the correctness of logical and physical features of implemented design are addressed to make the design process clear.

The rest of this paper is as follow. In Section 2, we describe parallel decimal multiplier in [1] (Jaberipur multiplier). In Section 3, we present the proposed multiplier architecture. Implementation methods and experimental results are presented in Section 4, and finally, section 5 concludes the paper.

2. Jaberipur Parallel Decimal Multiplier

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 systems based on binary signals. These issues complicate the generation and reduction of partial products [6]. In this section different parts of Jaberipur multiplier are described.

2.1 Partial Product Generation (PPG)

In this design, multiples of 0, X, 2X, 5X, $(8X_l+8X_h)$, $(9X_l+9X_h)$ are generated where X is multiplicand. The proper selector signals are derived from the multiplier digits. An abstract view of PPG for one digit-slice is depicted in Figure 1. This PPG method avoids to 4X generation and negation logic at the cost of increases hardware complexity of the PPG in order to generates 8X and 9X. The critical path of PPG in this design consists of computation of 8X followed by selection of multiples. In this part two selected multiples (outputs of s1 and s2) are added by means of decimal adder. The PPG module generates matrix of partial products consisting of BCD digits and redundant digits. This matrix is sent as input to PPR part.

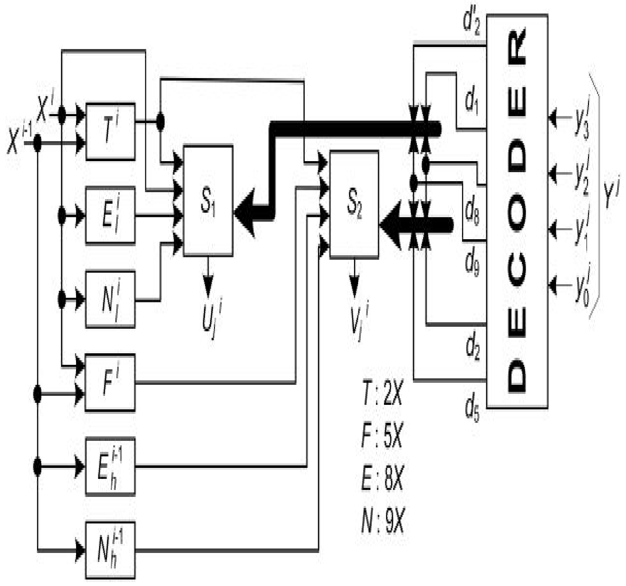

Figure 1 Selecting part of partial product generation in Jaberipur multiplier

2.2 Partial Product Reduction *(PPR)*

Partial product reduction is a special case of parallel multi-operand addition. A parallel method for PPR uses a reduction tree that leads to a redundant intermediate representation of product. In this design, if multiplier and multiplicand are considered as 16 digits of BCD, the matrix which is sent by PPG part will be a 32×32 matrix. In order to reduce the partial product in [1], a 6-level reduction tree is used. An abstract view of PPR for the deepest digit-slice of the 32-to-2 BCD reduction tree is depicted in Figure 2. In the PPR part of [1], the 16 BCD-FAs in the rightmost column of Figure 2 (first level of the PPR), do not use any carry-in and the other BCD-FAs use only half of the generated carries in the PPR. In order to convert the unused carries to BCD digits, a 9:4 counter and a 6:3 counter are used in this design. The carryout of the very last BCD-FAs is routed to a multiplexer that selects between b and b+1. As Figure 2 shows, the increment operation is performed off the critical path and in two logic levels.

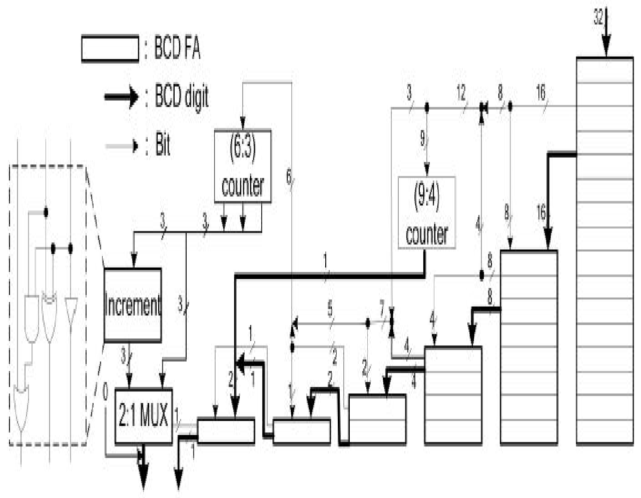

Figure 2 Partial product reduction of Jaberipur multiplier

2.3 Final Carry Propagating Addition (FCPA)

Decimal Addition requires a correction logic that forces an overhead to both delay and area. In some designs, this addition is done with carry save addition to cut the carry chain which has a more complex structure than usual decimal adders [12].
The last step in the Jaberipur multiplier requires a 32-digit BCD (128 bits) adder. This adder is a 32-digit conditional speculative adder, which provides input carry of each digit via a quaternary tree [6].

3. Improving the Jaberipur Multiplier

In order to improve Jaberipur multiplier, some structural contributions are applied to PPG and PPR parts of Jaberipur multiplier. As it is mentioned in section 2.1, in the last part of PPG, the outputs of two selectors S_1 and S_2 are added by means of decimal adders and the PPG output is a matrix consisting of BCD digits and redundant digits.

In our proposed design, after generating and selecting multiples in PPG part, they will sent as a matrix which is made of BCD-digits to the PPR part directly (instead of adding outputs of selectors S_1 and S_2 by decimal adders). And in PPR section of our proposed design, before using reduction tree in [1], the omitted decimal adder in PPG is used in order to reform the input matrix to the form which could be used by proposed PPR section in [1].

Our contribution is to omit decimal adders in PPG part of Jaberipur multiplier, use them in PPR part and adapt these two parts in order to present a new architecture which improves delay and area on FPGA and area and power on ASIC.

4. Implementation and Experimental Results

In this section, initially we will talk about our flow of implementation and synthesis of parallel decimal multipliers. After that we will present our experimental results

4.1 Top-Down VHDL Description

We have modeled parallel decimal multipliers with RTL VHDL. We used top-down methodology to manage the complexity of hardware. In this scheme, known blocks are modeled in RTL and other blocks (unknowns) are described with behavioral description at each level. In the next step, unknown blocks of the designs are described in detail gradually until whole of designs is described in RTL. This strategy permits simulating the whole system in each level of the design hierarchy.

After describing the multipliers, we did pre-synthesize simulation using ModelSim toolbox. In the next step, we synthesized all designs by means of LEONARDO to find our design's synthesis problems. Finally, we did post-synthesize simulation in which we tested all the synthesized designs with a random test generator using a large number of test vectors in order to check the correctness of all designs. It is worthwhile to note that all designs were modeled as a mix of sequential and concurrent VHDL codes.

4.2 Logic Synthesis and Experimental Results

Logic synthesis is an important phase in hardware designs [6]. For ASIC synthesis, the multipliers have been synthesized by TSMC 130nm technology in MAGMA toolbox and we used xc4vsx35 from Virtex4 family for FPGA synthesis on ISE toolbox.

We have estimated energy/power, delay, and area of parallel decimal multipliers for 16-digit (64bit) BCD input operands. Table 1 and Table 2 show the estimations of design characteristics after logic synthesis. As Table 1 shows, the power consumption of our proposed architecture is about 8.7 % better than Jaberipur multiplier and proposed design has less hardware complexity on ASIC.

Table 1 ASIC synthesis reports

Design	Delay (ns)	Power (mW)	Energy (pJ)	Area (cells)
[1]	6.1	15	91.5	34084
Proposed design	6.1	13.7	83.57	34032

Table 2 FPGA synthesis reports

Design	Delay (ns)	Power (mW)	Energy (pJ)	#slices
[1]	51.6	273	14086.8	14243
Proposed esign	50	259	12950	14193

As can be seen in Table 2, our structural changes in Jaberipur multiplier make it about 3.2% faster with less hardware complexity and about 5.2% less power consumption on FPGA.

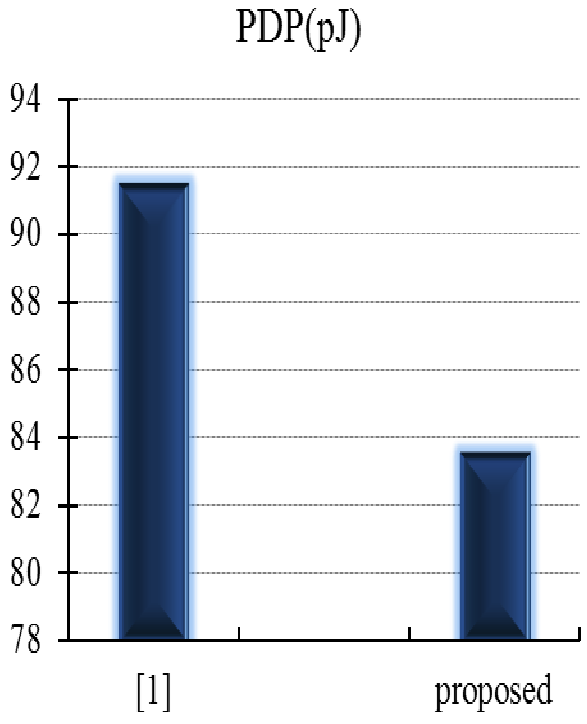

Figure 3 Energy comparison on ASIC

The energy comparison of Jaberipur multiplier and our proposed design on ASIC and FGPA are depicted in Figure 3 and Figure 4, respectively. The energy efficiency is one of the most important features of today's digital systems [7]. As Table 1 and Figure 3 illustrate, our proposed design is about 8.7% better than Jaberipur multiplier in term of energy consumption on ASIC.

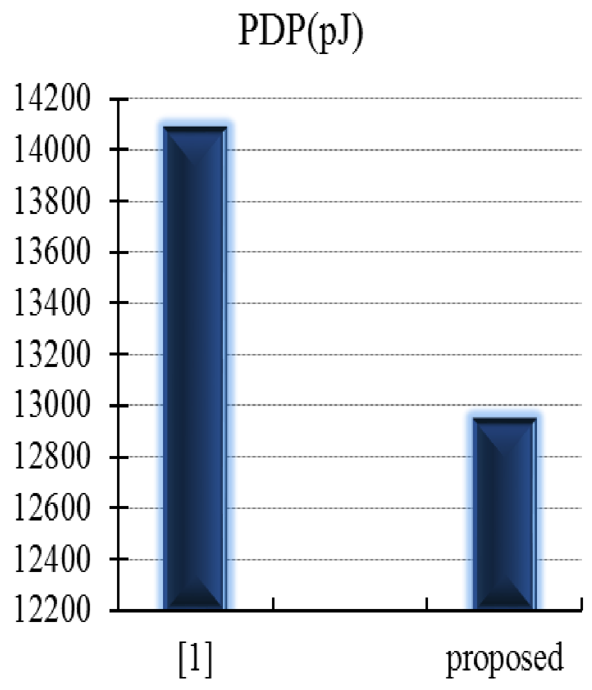

Figure 4 Energy comparison on FPGA

As can be seen in Table 2 and Figure 4, our proposed design is about 8.1% better than Jaberipur multiplier from energy consumption point of view.

The Floor-plans of Jaberipur and our proposed parallel decimal multiplier are depicted in Figure 5 and Fiqure 6, respectively.

As we mentioned before, in our proposed design, the decimal adders are omitted from PPG and are used in PPR section. As Figure 5 and Fiqure 6 show, our proposed multiplier has less design complexity. In our proposed design, the length of connector wires and empty spaces among different parts are decreased and design regularity is increased which results in better delay, area and energy/power consumption on FPGA and better area and energy/power consumption on ASIC.

5. Conclusions

Decimal multipliers are complex modules that are widely used in decimal computer arithmetic. In order to synthesize parallel decimal multipliers on ASIC, we have used MAGMA toolbox with TSMC 130nm technology file and for FPGA synthesis, we used xc4vsx35 from Virtex4 family on ISE toolbox.

In this paper, we have improved one of the best architectures proposed for decimal multiplication. We applied some changes to PPG and PPR parts of parallel decimal multiplier in [1] which make it faster with less energy/power consumption and less hardware complexity on FPGA and ASIC. Our experimental results show about 8.7% improvements in term of energy/power consumption on ASIC and about 8.1% improvements in term of energy consumption on FPGA.

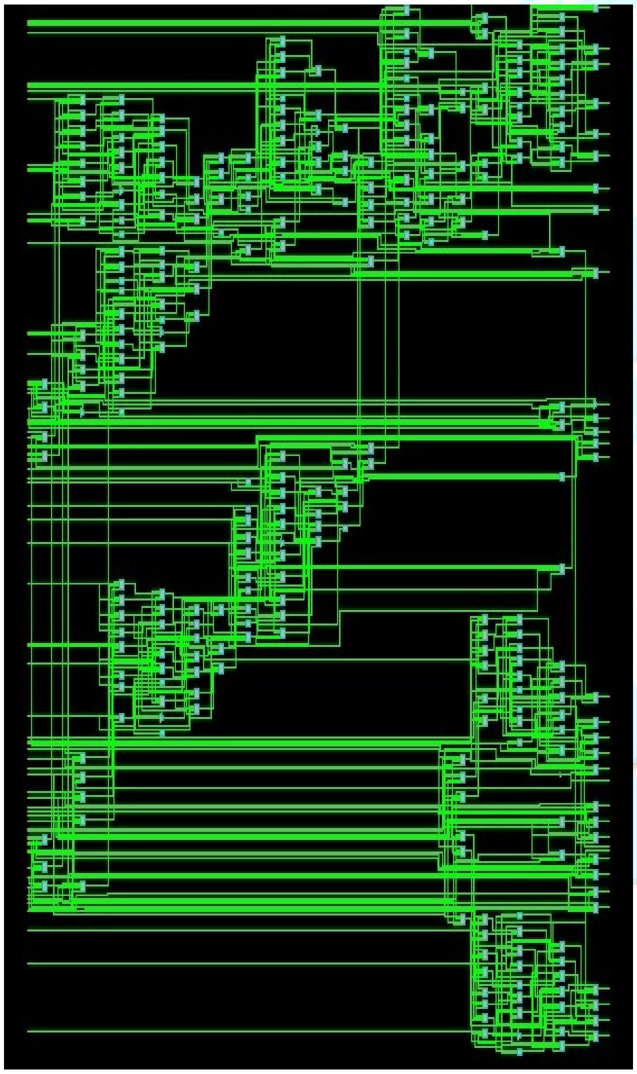
Figure 5 Floor plan of Jaberipur multiplier

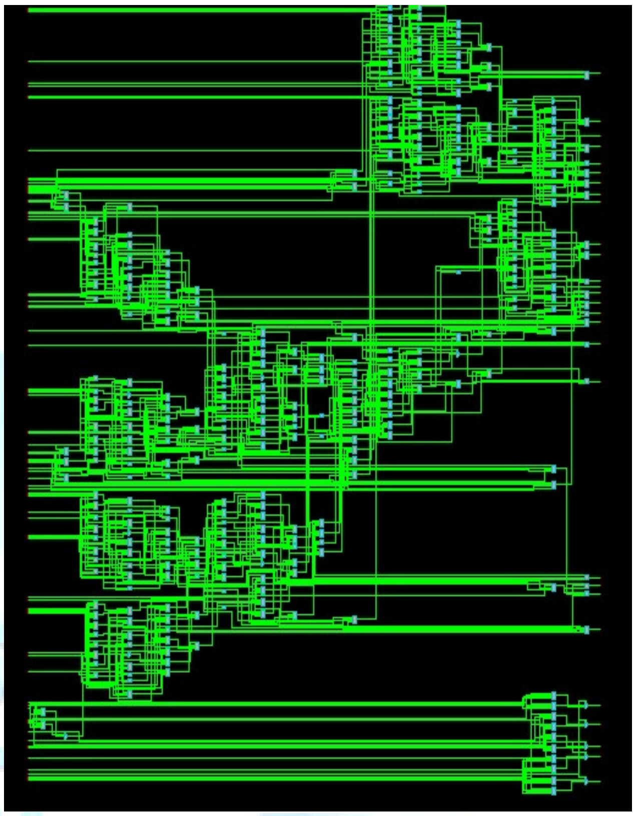
Figure 6 Floor plan of proposed design

References

[1] G. Jaberipur and A. Kaivani, “Improving the speed of parallel decimal multiplication”, IEEE Transactions on Computers, vol. 58, No. 11, Nov. 2009, pp. 1539-1552.

[2] T. Lang and A. Nannarelli, “A radix-10 combinational multiplier”, In Proc. 40th Asilomar Conf. Signals, Systems, and Computers, Oct. 2006, pp. 313-317.

[3] Va´zquez, E. Antelo, and P. Montuschi, “A new family of high-performance parallel decimal multipliers” In Proc. 18th IEEE Symp. Computer Arithmetic, June 2007, pp. 195-204.

[4] M. F. Cowlishaw, “Decimal floating-point algorism for computers”, In Proc. 16th IEEE Symposium on Computer Arithmetic, June 2003, pp. 104-111.

[5] Institute of Electrical and Electronics Engineers, In IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008, August 2008.

[6] A. Malekpour, A. Ejlali, S. Gorgin, “A Comparative Study of Energy/Power Consumption in Parallel Decimal Multipliers”, Microelectronics Journal, vol. 45, No. 6, 2014, pp. 775-780.

[7] A. Malekpour and A. Ejlali, “ Improving the Energy/Power Consumption of Parallel Decimal Multipliers”, Indian Journal of Science and Technology, vol. 7, No. 3, 2014, pp. 276-281.

[8] Eisen. L, Ward. J.W, Tast. H.-W, Mading. N, Leenstra. J, Mueller. S.M, Jacobi. C, Preiss. J, Schwarz. E.M, Carlough. S.R, "IBM power 6 accelerators: VMX and DFU", IBM Journal Research and Development, Vol. 51, No. 6, Nov. 2007, pp. 633-684.

[9] F. Webb, "IBM z10: The Next-Generation Mainframe Microprocessor," In IEEE Micro, Vol. 28, No. 2, March 2008, pp. 19-29.

[10] F. Busaba, C. Krygowski, W. Li, E. Schwarz, and S. Carlough, "The IBM z900 decimal arithmetic unit", Proceedings of the 35th Asilomar Conference on Signals, Systems and Computers, Vol. 2, IEEE Computer Society, Nov. 2001, pp. 1335-1339.

[11] M. A. Erle and B. J. Hickmann, "Decimal floating-point multiplication", In IEEE Transactions on Computers, Vol. 58, No. 7, 2009, pp. 902-916.

[12] M. A. Erle, M. J. Schulte, and B. J. Hickmann, "Decimal floating-point multiplication via carry-save addition", In Proceedings of the 18th IEEE Symposium on Computer Arithmetic, 2007, pp. 46-55.

[13] S. Gorgin and G. Jaberipur, "A fully redundant decimal adder and its application in parallel decimal multipliers", Microelectronics Journal, vol. 58, No. 11, Nov. 2009, pp. 1539-1552.

[14] A. V´azquez, E. Antelo, and P. Montuschi, "Improved design of high-performance parallel decimal multipliers", IEEE Transactions On Computers, vol. 59, No. 5, May 2010, pp. 679-693.

[15] A. V´azquez and E. Antelo, "Conditional speculative decimal addition", Proc. of 7th Conference on Real Numbers and Computers (RNC 7), July 2006, pp. 47–57.