FPGA in Decimal Arithmetic

Subjects: Computer Science, Hardware & Architecture

Contributor: Mário Véstias

Decimal operations are executed with slow software-based decimal arithmetic functions. For the fast execution of decimal operations, dedicated hardware units have been proposed and designed in FPGA. Decimal addition and multiplication is found in most decimal-based applications and so its design is very important for fast execution. This entry describes recent solutions for decimal multiplication and addition in FPGA.

decimal multiplication FPGA decimal addition

1. Introduction

Financial and commercial applications like accounting, banking, tax calculation, insurance and currency conversion require a large amount of data computing. Therefore, they are typically executed in high-performance computing platforms. These applications run over large databases of numbers which, in many cases, are represented in decimal format [1]. The last revision of the IEEE standard for floating-point arithmetic [2] includes specific definitions and rules for decimal operations and three different formats: decimal 32, decimal 64 and decimal 128, with 7, 16 and 34 coefficient digits.

Most general-purpose processors only have binary arithmetic units. So, the fastest solution to run decimal operations would be to convert decimal numbers to binary before being processed and then convert the result to decimal. The problem is that not all decimal numbers can be represented exactly as binary numbers with a finite number of bits. So, to avoid errors created from binary calculation that could lead to unwanted result deviations, arithmetic operations must be done directly over decimal numbers.

Executing decimal operations with binary arithmetic hardware without converting data to binary requires software algorithms for decimal arithmetic. Software-based decimal arithmetic is very slow compared to binary arithmetic implemented in hardware. However, the fast increase of commercial and financial transactions requires fast decimal arithmetic computing to meet real-time requirements and exact computations. Decimal addition and multiplication are fundamental arithmetic operations used in many applications. Therefore, fast decimal multipliers are important to obtain fast decimal-based applications.

FPGAs (Field Programmable Gate Array) are a good alternative for the execution of decimal arithmetic with dedicated hardware modules.

2. Decimal Addition in FPGA$w+z+6 \leq 15$

Decimal addition is implemented by direct manipulation of decimal numbers or using a binary adder followed decimal correction.

Some of the first decimal adders [3][4] were based on 1-digit BCD. Others [5] use a direct BCD carry look-ahead adder or a carry-select technique to conditionally sum 6 in binary to do the decimal correction [6][7].

Other decimal addition approach considers intermediate representations that reduce the complexity of decimal addition but requires converters from and to BCD of the intermediate representation. For example, [8][9] consider a redundant BCD representation to achieve carry free operations.

To take advantage of binary arithmetic circuits, some decimal addition solutions consider binary adders followed by number correction [10][11][12][13][14][15][16][17].

Considering also binary-based decimal addition, some works use binary to BCD conversion [18] and BCD to binary conversion [19] to use binary adders. The main problem of this solution is the overhead of binary-BCD conversions [20].

All previous solutions can be implemented in FPGA. However, some works were proposed specifically for decimal addition in FPGA. Decimal adders wer designed for FPGAs with 4-input LUTs [21] and with 6-input LUTs [16]. Decimal adders with binary
adders followed by correction stages were also implemented in 6-input LUT FPGAs [22][23][24].

Multioperand decimal addition is a particular case of decimal addition where techniques like carry-save addition can be applied efficiently [6][25]. In [25] a new BCD adder was proposed to efficiently implement multioperand addition. In [4][7][22] the tree structure was identified as the most efficient approach for multioperand addition in FPGA.

In [26] a decimal adder was proposed that considers an excess-6 representation to avoid carry propagation of addition. This adder is used in the proposed multipliers to implement the adder tree. It also serves as the base for a novel decimal adder/subtractor necessary for the design of the partial product generators.

3. Decimal Multiplication in FPGA

Processors with dedicated decimal hardware multipliers implement them with iterative algorithms [27][28] to reduce the size of the arithmetic unit. However, iterative algorithms are slow compared to parallel implementations due to its iterative nature. for
fast execution, parallel decimal multiplication consists of partial product generation for each multiplier digit followed by partial product addition. Partial product generation of a N×N multiplication can be implemented with N×N small digit by digit multipliers or N digit by multiplicand multipliers. A digit by digit multiplier can be implemented with logic or with look-up tables [29][30][31]), for fast and compact design. However, given the quadratic number of digit by digit

multipliers necessary to implement a multiplication, these solutions are viable only for small operand sizes. The proposal in [4] considered recoding of operands to simplify digit by digit multiplication for partial product generation. However, the performance and area of the decimal multiplier based on digit by digit multiplication is still worst than a multiplier with a partial product for each multiplier digit.

The approach followed to implement a 1×N multiplier is to determine the decimal multiples of the multiplier. A direct approach to a design a decimal multiplier based on multiples generates all multiples of the multiplicand. Then, selects the required multiples according to the multiplier digits. The generated multiples are then shifted and added to generate the final product. While simple, the method requires a large multiplexer with all multiplies for each multiplier digit and the generation of all multiples from A to 9A. Knowing that the generation of some multiples are not carry-free, this solution degrades the performance of the multiplier.

Therefore, authors started to consider only a limited set of multiples. In [32] only multiples A, 2A, 4A, 5A are used, since they can be generated without carry propagation (multiple 4A is generated from 2A in sequence as 2×2A). The other multiples are obtained by adding two of these multiples. Since multiple 4A cannot be generated in a single carryfree step, it has been removed from the set of base multiples in [7]. The other multiples are obtained by adding a multiple from the set {0, 5X, 10X} and a multiple from the set {-2X, -X, 0, X, 2X}. For fast selection of multiples, digits of the multiplier are first recoded, but the solution requires a large multiplexer for each multiplier digit.

Since then, other sets of multiples and encodings were considered. In [14] two different decimal encodings (4221 and 5211) are used to generate and reduce the partial products with two different architectures. In one the architectures the multiplier is recoded into a signed-digit (SD) set [-5, 5], while in the other the multiplier is encoded as A = YU5 +YL like in [7], where $Y^U \in \{0, 1, 2\}$ and $Y^L \in \{-2, -1, 0, 1, 2\}$. Signed-digit (SD) recoding of the multiplier in the set [-5, 5] was adopted by several authors for the implementation of a decimal multiplier [33][34]. The base architecture generates multiples {0, X, 2X, 3X, 4X, 5X}. These are selected for each partial product and the output is complemented to obtain the negative of the multiple. The partial products are then reduced with a partial product reduction module. Different representations are used to improve the generation of complements and the decimal addition. The radix-5 algorithm proposed in [14] was followed by [35] but using an hybrid 8421–5421 representation.

In [15] a decimal multiplier is proposed using a redundant decimal addition algorithm based on a weighted bit-set encoding. The method generates double BCD (Binary-Coded Decimal) numbers using decimal multiples 2X, 4X, and 5X. The redundant decimal adder is used to reduce the generated 2n BCD partial products to a redundant number in the range of [0, 15]. The final redundant product is then converted to BCD encoding.

The special case of constant decimal multiplication was considered in [36]. Constant decimal multiplication is widely used in economic and financial applications. The authors address this problem to design a solution with smaller area, power and delay compared to constant decimal multiplication implemented with a general decimal multiplier.

The work proposes a new redundant digit set in {0, 18} and a 3:1 compressor. The results show an improvement in the area up to 89%.

Partial products are then added is a step known as partial product reduction using decimal adders. Partial product reduction can be designed with an adder tree or with a multioperand adder. An adder tree successively reduces pairs of partial products until a final result. Multioperand addition takes into account that multiple partials have to be reduced into a single value. In [13] three techniques were proposed for multioperand decimal addition. Two of the approaches consider speculative addition that speculates about BCD correction values which are corrected while adding the operands. The other technique uses a binary adder that produces a binary sum which is then corrected. This last technique achieved the best area-delay results. A mixed binary and BCD multioperand addition was proposed in [37]. Digits in a column are all added in binary, converted to decimal and finally added with decimal adders.

In [7][32] the adder tree is implemented with decimal carry look-ahead adders. In [38] partial products are recoded to 4221. This codification simplifies addition since it avoids he correction step. The method reduces three partial products to two equally weighted 4221 decimal digits. These two operands are then converted to BCD add added to generate the final result.

A different approach for decimal multiplication considers binary multipliers as the base arithmetic unit [19][20][39][40]. This permits using binary multipliers that are faster and may already be available in the system. Also, it implements both binary and decimal multiplication in a single module. The method first converts the BCD operands of the multiplication to binary. The converted operands are then multiplied using the binary multiplier. The binary product is then converted to BCD. The main drawback of the binary-based method is the large overhead introduced by the converters [18][19]. A balanced solution was proposed in [20] that subdivides the multiplier and the multiplicand into smaller blocks and applies the method to each of these sub-blocks. The partials are then aligned and added using decimal adders to generate the final product.

Most works on decimal multiplication target ASICs, but several architectures have been proposed for FPGA and coarse-grained reconfigurable computing [41]. Any of the previous architectures can be directly mapped to FPGA. However, a careful adaptation of the design leads to a more efficient architecture since logic functions in FPGAs are implemented with look-up tables. In [42] a parallel implementation of a multiplier was mapped in Virtex-4 FPGA from Xilinx. The architecture obtains the partial products using digit by digit multiplication with a binary multiplier followed by binary to BCD conversion [43]. The work in [22] described previously was mapped on a 6-input LUT FPGA.

A new optimization of the multiplication algorithm was considered in [44] where the application of the Karatsuba-Ofman algorithm reduces the area of the parallel decimal multipliers on FPGA at the cost of an increase in delay. A BCD multiplier using the atomic 1×1 digit multiplier was proposed in [45]. The effort of the work is on the partial product reduction unit. The two-digit partial products of all 1×1 digit multiplications are correctly aligned to generate

the complete partial products. The partial products are then reduced with a mix of binary decimal compressors and decimal adders.

Recently, a new decimal multiplier [46] improved the area of the best previous decimal multipliers on FPGA by about 20%. The solution considers a new decimal adder based on a mixed BCD/excess-6 representation and a 5221 recoding of the multiplier digits. Partial products are obtained from the addition of a multiple in the set {0, 2X, 5X, 2X+5X} and a multiple in the set {X, 2X}.

Two novel decimal multipliers on FPGA with different area/performance tradeoffs with both multipliers improving the area and performance of state-of-the-art multipliers were proposed in [47]. Both methods use a new adder/subtractor based on the excess-3 representation of multiples. Two different sets of multiples are considered: {0, X, 2X, 5X, 10X} and {2X, 4X, 5X}. Partial products are obtained by the addition or subtraction of two multiples of the sets. The method permits a very efficient generation of multiples, which considerably reduces the required resources.

References

1. Tsang, A.; Olschanowsky, M. A Study of Database 2 Customer Queries. Technical report, IBM Santa Teresa Laboratory, San Jose, USA, 1991.
2. IEEE Standards Committee. 754-2008 IEEE Standard for Floating-Point Arithmetic. 2008, pp. 1–58.
3. Veeramachaneni, S.; Kirthi Krishna, M.; Avinash, L.; P, S.R.; Srinivas, M. Novel, High-Speed 16-Digit BCD Adders Conforming to IEEE 754r Format. VLSI, 2007. ISVLSI ’07. IEEE Computer Society Annual Symposium on, 2007, pp. 343–350. doi:10.1109/ISVLSI.2007.71.
4. Erle, M.A.; Schwarz, E.M.; Schulte, M.J. Decimal Multiplication with Efficient Partial Product Generation. Proceedings 17th IEEE Symposium on Computer Arithmetic, 2005, pp. 21–28.
5. Bayrakci, A.; Akkas, A. Reduced Delay BCD Adder. Application-specific Systems, Architectures and Processors, 2007. ASAP. IEEE International Conf. on, 2007, pp. 266–271. doi:10.1109/ASAP.2007.4429991.
6. Bioul, G.; Vázquez, M.; Deschamps, J.P.; Sutter, G. Decimal addition in FPGA. Proceedings of V Southern Programmable Logic Conference, 2009, pp. 101–108.
7. Lang, T.; Nannarelli, A. A radix-10 combinational multiplier. Proceedings IEEE 40th International Asilomar Conference onSignals, Systems, and Computers, 2006, pp. 313–317.
8. Shirazi, B.; Yun, D.; Zhang, C. RBCD: Redundant Binary Coded Decimal Adder. IEE Proceedings 1989, 136.

9. Yehia, K.; Fahmy, H.; Hassan, M. A redundant decimal floating-point adder. Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on, 2010, pp. 1144–1147. doi:10.1109/ACSSC.2010.5757583.
10. Fischer, H.; Rohsaint, W. Circuit Arrangement for Adding or Subtracting Operands in BCD-Code or Binary-Code. United States 5146423, 1992.
11. Grupe, U. Decimal Adder. United States 3935438, 1976.
12. Haller,W.; Krauch, U.;Wetter, H. Combined Binary/Decimal Adder Unit. United States 5928319, 1999.
13. R.D. Kenney; M.J. Schulte; High-Speed Multioperand Decimal Adders. *IEEE Transactions on Computers* **2005**, *54*, 953-963, 10.1109/tc.2005.129.
14. Alvaro Vazquez; Elisardo Antelo; Paolo Montuschi; Improved Design of High-Performance Parallel Decimal Multipliers. *IEEE Transactions on Computers* **2009**, *59*, 679-693, 10.1109/tc.2009.167.
15. Saeid Gorgin; Ghassem Jaberipur; A fully redundant decimal adder and its application in parallel decimal multipliers. *Microelectronics Journal* **2009**, *40*, 1471-1481, 10.1016/j.mejo.2009.07.002.
16. Vazquez, A.; Antelo, E. A High-Performance Significand BCD Adder with IEEE 754-2008 Decimal Rounding. ComputerArithmetic, 2009. ARITH 2009. 19th IEEE Symposium on, 2009, pp. 135–144. doi:10.1109/ARITH.2009.30.
17. Morteza Dorrigiv; Ghassem Jaberipur; Low area/power decimal addition with carry-select correction and carry-select sum-digits. *Integration* **2014**, *47*, 443-451, 10.1016/j.vlsi.2014.01.004.
18. Osama Al-Khaleel; Zakaria Al-Qudah; Mohammad Al-Khaleel; Christos Papachristou; High performance FPGA-based decimal-to-binary conversion schemes for decimal arithmetic. *Microprocessors and Microsystems* **2013**, *37*, 287-298, 10.1016/j.micpro.2013.01.002.
19. Mahmood Fazlali; Hadi Valikhani; Somayeh Timarchi; Hadi Tabatabaee Malazi; Fast architecture for decimal digit multiplication. *Microprocessors and Microsystems* **2015**, *39*, 296-301, 10.1016/j.micpro.2015.01.004.
20. Véstias, M.; Neto, H. Parallel Decimal Multipliers using Binary Multipliers. Proceedings IEEE 6th Southern Programmable Logic Conference, 2010, pp. 73–78.
21. Yixiong, G.; Jun, D.; Na, L.; Jun, Y. Notice of Retraction A Research and Design of Decimal Floating Multiplier Based on FPGA. Knowledge Discovery and Data Mining, 2010. WKDD ’10. Third International Conference on, 2010, pp. 314–319. doi:10.1109/WKDD.2010.44.
22. Vázquez, A.; de Dinechin, F. Efficient implementation of parallel BCD multiplication in LUT-6 FPGAs. Proceedings of 2010 International Conference on Field-Programmable Technology (FPT), 2010, pp. 126–133.

23. G. Bioul; M. Vazquez; J. P. Deschamps; Gustavo Sutter; High-Speed FPGA 10's Complement Adders-Subtractors. *International Journal of Reconfigurable Computing* **2010**, *2010*, 1-14, 10.1155/2010/219764.
24. Gao, S.; Al-Khalili, D.; Chabini, N. An improved BCD adder using 6-LUT FPGAs. New Circuits and Systems Conference(NEWCAS), 2012 IEEE 10th International, 2012, pp. 13–16. doi:10.1109/NEWCAS.2012.6328944.
25. Vázquez, A.; de Dinechin, F. Multi-operand decimal adder trees for FPGAs. Research report rr-7420, INRIA, 2010.
26. Horácio Neto; Mário Véstias; Decimal addition on FPGA based on a mixed BCD/excess-6 representation. *Microprocessors and Microsystems* **2017**, *55*, 91-99, 10.1016/j.micpro.2017.10.004.
27. Véstias, M.P.; Neto, H.C. Revisiting the Newton-Raphson Iterative Method for Decimal Division. 2011 21st International Conference on Field Programmable Logic and Applications, 2011, pp. 138–143. doi:10.1109/FPL.2011.33.
28. Véstias, M.P.; Neto, H.C. Iterative decimal multiplication using binary arithmetic. 2011 VII Southern Conference on Programmable Logic (SPL), 2011, pp. 257–262. doi:10.1109/SPL.2011.5782658.
29. Larson, R.H. High-Speed Multiply Using Four Input Carry-Save Adder. IBM Technical Disclosure Bull 1973, 16, 2053–2054.
30. Ueda, T. Decimal Multiplying Assembly and Multiply Module. United States 5379245, jan, 1995.
31. Encarnación Castillo; Antonio Lloris; Diego P. Morales; Luis Parrilla; Antonio García; Guillermo Botella; A new area-efficient BCD-digit multiplier. *Digital Signal Processing* **2017**, *62*, 1-10, 10.1016/j.dsp.2016.10.011.
32. Erle, M.A.; Schulte, M.J. Decimal multiplication via carry-save addition. Proceedings 14th IEEE International Conference onApplication Specific Systems, 2003, pp. 348–358.
33. Saeid Gorgin; Ghassem Jaberipur; Sign-Magnitude Encoding for Efficient VLSI Realization of Decimal Multiplication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2016**, *25*, 75-86, 10.1109/tvlsi.2016.2579667.
34. Xiaoping Cui; Wenwen Dong; Weiqiang Liu; Earl E. Swartzlander; Fabrizio Lombardi; High Performance Parallel Decimal Multipliers Using Hybrid BCD Codes. *IEEE Transactions on Computers* **2017**, *66*, 1994-2004, 10.1109/tc.2017.2706262.
35. Ming Zhu; Yingtao Jiang; Mei Yang; Tianding Chen; On high-performance parallel decimal fixed-point multiplier designs. *Computers & Electrical Engineering* **2014**, *40*, 2126-2138, 10.1016/j.compeleceng.2014.08.013.

36. Sara Sadat Hoseininasab; Hooman Nikmehr; Architectures for multiple constant decimal multiplication. *Computers & Electrical Engineering* **2019**, *75*, 31-45, 10.1016/j.compeleceng.2019.01.024.
37. Luigi Dadda; Multioperand Parallel Decimal Adder: A Mixed Binary and BCD Approach. *IEEE Transactions on Computers* **2007**, *56*, 1320-1328, 10.1109/tc.2007.1067.
38. Vázquez, A.; Antelo, E.; Montushi, P. A New Family of High-Performance Parallel Decimal Multipliers. Proceedings IEEE 18th Symposium on Computer Arithmetic, 2007, pp. 195–204.
39. Neto, H.; Véstias, M. Decimal Multiplier on FPGA using Embedded Binary Multipliers. Proceedings IEEE Field Programmable Logic and Applications, 2008, pp. 197–202.
40. Sasidhar Mukkamala; Pradeep Rathore; Rangababu Peesapati; Decimal multiplication using compressor based-BCD to binary converter. *Engineering Science and Technology, an International Journal* **2018**, *21*, 1-6, 10.1016/j.jestch.2018.01.003.
41. Samaneh Emami; Mehdi Sedighi; An optimized reconfigurable architecture for hardware implementation of decimal arithmetic. *Computers & Electrical Engineering* **2017**, *63*, 18-29, 10.1016/j.compeleceng.2017.08.018.
42. Sutter, G.; Todorovich, E.; Bioul, G.; Vázquez, M.; Deschamps, J.P. FPGA Implementations of BCD Multipliers. Proceedings IEEE International Conference on Reconfigurable Computing and FPGAs, 2009, pp. 36–41.
43. Jaberipur, G.; Kaivani, A. Binary-coded decimal digit multipliers. IET Computer Digital Techniques 2007, 1, 377–381.
44. Véstias, M.; Neto, H. Parallel Decimal Multipliers and Squarers Using Karatsuba-Ofman's Algorithm. 15th Euromicro Conference on Digital System Design, 2012, pp. 782–788.
45. Shuli Gao; Dhamin Al-Khalili; J. M. Pierre Langlois; Noureddine Chabini; Efficient Realization of BCD Multipliers Using FPGAs. *International Journal of Reconfigurable Computing* **2017**, *2017*, 1-12, 10.1155/2017/2410408.
46. Mário Véstias; Horácio Neto; Improving the area of fast parallel decimal multipliers. *Microprocessors and Microsystems* **2018**, *61*, 96-107, 10.1016/j.micpro.2018.05.015.
47. Mário Véstias; Horácio Neto; Decimal Multiplication in FPGA with a Novel Decimal Adder/Subtractor. *Algorithms* **2021**, *14*, 198, 10.3390/a14070198.