Improved Radix-10 Multiplication of Decimal 128 Calculations

Vikram Singh Yadav[1] , Prof. Suresh Gawande[2]
[1]M-Tech Research, [2]Research Guide and HOD

Department of Electronics and Communication Engineering, Bhabha Research Engineering Institute, Bhopal

Abstract - The radix-10 multiplication is explained in this paper and the performance improved using novel proposed approach is also shown here. The upcoming computation technology needed fast calculations of the high performance decimal numbers and to achieve this goal we need efficient decimal multipliers. The PPG is performed using signed-digit radix-10 reduction of the multiplier and a simplified set of multiplicand multiples. The reduction of partial products is performed in a tree structure based on a proposed algorithm decimal multi operand carry-save addition which uses a specific decimal-coded number systems. We further detail these techniques and it significantly improves the area and delay of the existing work.

Keywords: Decimal 128 number, BCD, Radix-10, Fast Multiplication.

I. INTRODUCTION

A variety of designs had been proposed for designing a BCD multiplier to improve its performance of multiplication operation. The carry save format10 used in BCD multiplication represents a radix-10 operand using a carry value at its each decimal position. The use of carry-free accumulation of partial products resulted from BCD operands requires a series of BCD digit adders10 or a tree configuration of adders1. The use of decimal signed-digit representation11,15 relays on digit set of redundant values {-x,….,0,….,x}, $5 \leq x \leq 9$, to allow carry free addition. The available radix-10 with signed-digit and carry save arith-metics offers improvements in performance. However this type of VLSI implementation techniques will result in irregular layouts than the existing binary carry save adders and compressor trees.

Various redundant BCD codes namely, 4221, 5211 and 4211 can be used in BCD arithmetic as their 4-bit decimal codes satisfy the sum of weights of bits to be equal to 9, so that all the combinations of a 4-bit number can be used for a decimal [0–9] number representation. This property of redundancy can be used to speed up decimal operations without altering the data path width. Also the main advantage of using redundant numbers is in getting complement of a number, which is obtained by just inverting the bits of a 4-bit representation. A disadvantage of such codes is that it is constrained to digit bounds, due to which obtaining the multiple 3X is not possible in a carry-free way.

The design methodology for a p × q-digit BCD multiplier is shown in Figure 1. This design accepts inputs that is conventional decimal such as redundant BCD operands M, N and produces a BCD product after generating a partial products array (redundant). The following three stages of operation has to be performed to obtain the decimal product: (1) Partial product generation (PPG) along with generation of multiplicand multiples coded in excess-3 and recoding of multiplier operand. (2) Partial Product Reduction (PPR) after recoding of partial

Products to ODDS code and (3) Conversion to non-redundant BCD product of p + q digits. The detailed explanation of proposed architecture is given in the subsequent sections.

Decimal Partial Product Generation:

A signed-digit radix-10 recoding is used in the proposed BCD multiplier. This type of design reduces the number

of partial products generated so that the area of the physical layout required is reduced. Generation of partial products include recoding of multiplier digit to signed-digit radix-10, calculation of multiples in excess-3 code and generation of ODDS partial products. Figure 2 shows architecture for generating partial products. The multiplicand digit Yk of a BCD number is recoded in SD radix-10 recoder to generate a 5-bit one-hot code represented by Ybk = {Y1k, Y2k, Y3k, Y4k, Y5k}. The obtained Ybk is used as a select lines for a 5:1 multiplexer to select a respective multiplicand multiples {1X, 2X, 3X, 4X, 5X} Ybk also include 1-bit value called signed bit, which is used to control the negation of a selected number.

Multiplicand Multiples:

Set of multiplicand multiples are generated by coding in excess-3 format given by NX ; { 1X, 2X, 3X, 4X, 5X } with digits NXi in the range [–3, 12], defined by [NXi] = NXi + 3, ; [0, 15] it takes two steps:

• Digit Mapping of multiplicand digits and

• Assimilation of the carries between adjacent digits.

II. SYSTEM MODEL

The ODDS (overloaded BCD or overloaded decimal digit set) representation was proposed to improve the decimal operation in parallel and sequential decimal multiplication. These codes represent the redundant radix-10 digit Xi. The use of ODDS code has various advantages in hardware implementation namely: (1) it allows us to generate simple and complex multiples (3X, 4X, 5X…) and to perform addition in a carry-free way, (2) Unlike in simple BCD there is no need of additional hardware to correct the invalid 4-bit combination and (3) Since digits are represented in binary format, there is no need for extra hardware other than binary arithmetic circuits.

In this paper the algorithm, architecture and FPGA realization of BCD multiplier which focuses on improving the multiplication operation using parallel architecture by utilizing the redundant property of two decimal representations: that is, redundant BCD excess-3 (XS-3) code and the ODDS. The minimal number of digits is used for recoding of the decimal numbers. The signed-digit radix-10 digits used here are in the range of {–5,–4,…., 0,…,4,5}. Main issue with this set of digits is obtaining of multiples without long carry propagation. In this proposal acceleration to the multiplication operation is given in two steps: Partial Product Generation (PPG) and Partial Product Reduction (PPR) which are explained in coming sections.

Redundant BCD Representations:

The proposed BCD multiplier uses a redundant number internally to simplify the implementation and to increase the speed of operation. The radix-10 ten's complement has been used which is described below:

$$Z = -S_z * 10^n \sum_{x=0}^{n-1} Zx \; * \; 10x$$

Where n being number of digits, *Zi* ; [m - e, 1 - e] is the *i*th digit with range from 0 ≤ m ≤ e, and 9 + e ≤ m ≤ 24 – 1 (=15) and *Sz* is the sign bit. Parameter e is the excess value which takes the range from 0 (for non-excess).

In this work, signed digit radix-10 is used for a BCD multiplier to compute the multiplicand multiples. The main issue with the multiplication (mainly for 3X) is to avoid the carry propagation for a longer path. The range of digits obtained after generating multiple 3X is in between hence maximum carry that is obtained will be of just one digit. The nine's complement of a positive decimal number is given by the equation,

$$-10^n + \sum_{i=0}^{n-1} (9 - Z_i) * 10^i$$

The implementation of the term (9 – *Zi*) has more complexity as the number exceeds 9. So the implementation can be made simpler by taking excess-3 value of nine's complement which is obtained by just taking bit complement of excess-3. Table 1 shows how to get the nine's complement.

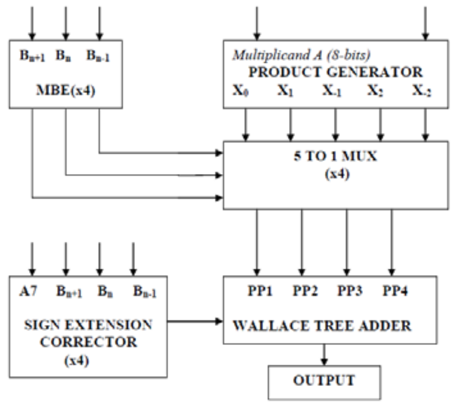

Fig. 2.1. Architecture of designed Booth Multiplier in the Project.

III. PROPOSED ARCHITECTURE

For faster multiplication of radix-10 numbers with BCD this work proposed an architecture shown below. The area and delay of proposed architecture is quite better than the previous architecture. Here we have working on decimal 128 muliplication which was suggested by the authors of [1] and the architecture is designed for SPARTAN 6 FPGA devices.

The sysnthesis outcomes are shown in the result section of the paper.

In Fig. 3.1 the stages of radix-10 architecture is shown. The stages are also explained below:

Stage 1) Decimal partial product generation. A SD radix-10 recoding of the BCD multiplier has been used. This recoding produces a reduced number of partial products that leads to a significant reduction in the overall multiplier area [16].

Stage 2) Decimal partial product reduction. In this stage, the array of d þ 1 ODDS partial products are reduced to two 2d-digit words (A, B).

Stage 3) Conversion to (non-redundant) BCD. We consider the use of a BCD carry-propagate adder [16] to perform the final conversion to a non-redundant BCD product P =A+ B.

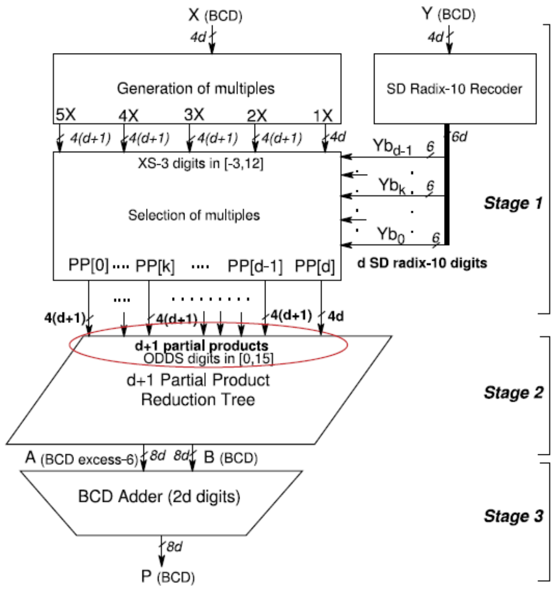

Fig. 3.1. Combinational SD radix-10 architecture.

The partial product generation stage comprises the recoding of the multiplier to a SD radix-10 representation, the calculation of the multiplicand multiples in XS-3 code and the generation of the ODDS partial products.

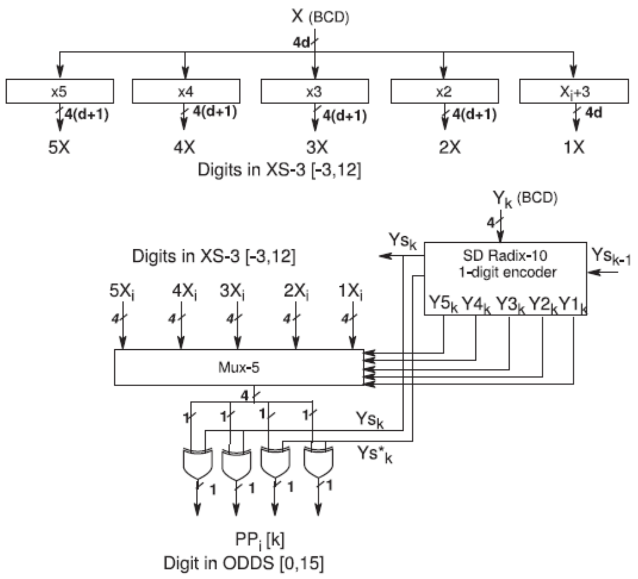

Fig. 3.2. SD radix-10 generation of a partial product digit.

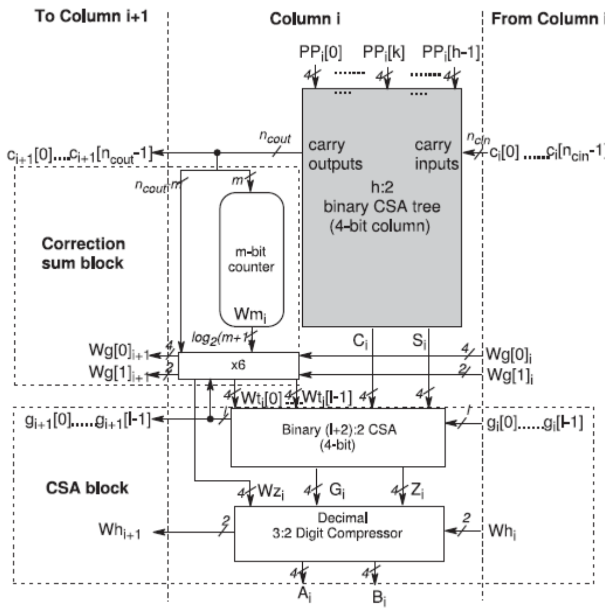

Fig. 3.3. High-level architecture of the proposed decimal PPR tree (h inputs, 1-digit column).

Decimal 128 Implementation:

The maximum height of the partial product array by the 34x34-digit BCD multiplier is h=35. The proposed implementation for the maximum height columns of the PPR tree is shown in Fig. 8. The binary 35 : 2 CSA tree is built of a first level of three 9 : 2 and one 8 : 2 compressors, and a second level of one 8 : 2 compressor. The number of carries transferred to the next column of the binary CSA tree is 33.

The selected architecture is a 2d-digit hybrid parallel prefix/carry-select adder, the BCD Quaternary Tree adder [16]. The delay of this adder is slightly higher to the delay of a binary adder of 8d bits with a similar topology. The decimal carries are computed using a carry prefix tree, while two conditional BCD digit sums are computed out of the critical path using 4-bit digit adders which implements *[Ai]+[Bi]+0* and *[Ai]+[Bi]+1*. These conditional sums correspond to each one of the carry input values. If the conditional carry out from a digit is one, the digit adder performs a -6 subtraction. The selection of the appropriate conditional BCD digit sums is implemented with a final level of 2 : 1 multiplexers.

To design the carry prefix tree we analyzed the signal arrival profile from the PPRT tree, and considered the use of different prefix tree topologies to optimize the area for the minimum delay adder.

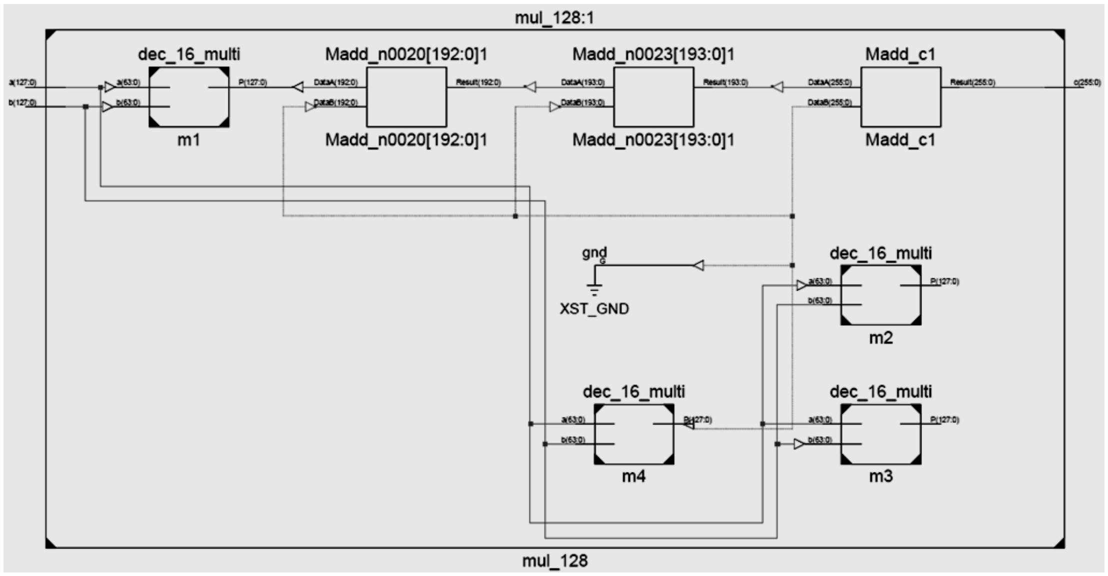

Fig. 3.4 RTL Schematic of Proposed Architecture

IV. SYNTHESIS RESULTS

The proposed architecture shown in the previous section and synthesized on XILINX 13.1. The FPGA device used is SPARTAN 6. The device utilization summary and the timing summary where delay is shown is given in below Table 2 and 3 respectively.

The comparison of area and delay from previous work is also shown in Table 1.

Architecture	Delay	Area
Previous [1]	56ns	120600
Proposed	22.125ns	40836

Table 2: Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	25523	2400	1063%
Number of fully used LUT-FF pairs	0	25523	0%
Number of bonded IOBs	512	102	501%
Number of DSP48A1s	25523	2400	1063%

Table 3: Timing Summary

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

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 994740777074063360 / 256
-------------------------------------------------------------------------
Delay:               22.125ns (Levels of Logic = 266)  Source:            b<50> (PAD)
  Destination:       c<255> (PAD)
    ----------------------------------------
Total                     22.125ns (12.989ns logic, 9.136ns route)
                                       (58.7% logic, 41.3% route)
```

V. CONCLUSION AND FUTURE SCOPE

The proposed architecture for decimal 128 number is shown in the proposed section of this paper and the synthesis results after implementing on the Spartan 6 FPGA device is also shown in the previous section, and it can be concluded that the proposed architecture has better area utilization and delay profile than the existing[1] work. But it can faster if we modified the architecture in future to implement higher decimal number for calculations and the device utilized will also play very important role to speed up higher decimal number like 256 bit etc.

REFERENCES

[1] Vazquez, A.; Antelo, E.; Bruguera, J.D., "Fast Radix-10 Multiplication Using Redundant BCD Codes," in *Computers, IEEE Transactions on* , vol.63, no.8, pp.1902-1914, Aug. 1 2014.

[2] Wang, Zhuo; Han, Liu; Seok-Bum Ko, "Design and implementation of a Radix-100 division unit," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* , vol., no., pp.1239-1242, 20-23 May 2012.

[3] Minchola Guardia, C.E., "Implementation of a fully pipelined BCD multiplier in FPGA," in *Programmable Logic (SPL), 2012 VIII Southern Conference on* , vol., no., pp.1-6, 20-23 March 2012.

[4] Kornerup, P.; Lefevre, V.; Louvet, N.; Muller, J.-M., "On the Computation of Correctly Rounded Sums," in *Computers, IEEE Transactions on* , vol.61, no.3, pp.289-298, March 2012.

[5] Lang, T.; Nannarelli, A., "Comments on 'improving the speed of decimal division'," in *Computers & Digital Techniques, IET* , vol.6, no.6, pp.370-371, November 2012.

[6] Ercegovac, M.D.; McIlhenny, R., "Shared implementation of radix-10 and radix-16 square root algorithm with limited precision primitives," in *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on* , vol., no., pp.345-349, 4-7 Nov. 2012.

[7] Emami, S.; Dorrigiv, M.; Jaberipur, G., "Radix-10 addition with radix-1000 encoding of decimal operands," in *Computer Architecture and Digital Systems (CADS), 2012 16th CSI International Symposium on* , vol., no., pp.139-144, 2-3 May 2012.

[8] Ercegovac, M.D.; McIlhenny, R., "Shared implementation of radix-10 and radix-16 division algorithm with limited precision primitives," in *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on* , vol., no., pp.1828-1832, 6-9 Nov. 2011.

[9] Khan, K.M.N.H.; Ali, M.L.; Islam, S., "A new technique for high speed decimal logarithm computation of decimal floating-point number," in *Computer and Information Technology (ICCIT), 2011 14th International Conference on* , vol., no., pp.208-212, 22-24 Dec. 2011.

[10] Baesler, M.; Voigt, S.; Teufel, T., "FPGA Implementations of Radix-10 Digit Recurrence Fixed-Point and Floating-Point Dividers," in *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on* , vol., no., pp.13-19, Nov. 30 2011-Dec. 2 2011.

[11] Faraday Tech. Corp. (2004). 90nm UMC L90 standard performance low-K library (RVT). [Online]. Available: *http://freelibrary.faraday-* tech.com/

[12] S. Gorgin and G. Jaberipur, “A fully redundant decimal adder and its application in parallel decimal multipliers,” Microelectron. J., vol. 40, no. 10, pp. 1471–1481, Oct. 2009.

[13] S. Gorgin and G. Jaberipur. (2013, May). “High speed parallel decimal multiplication with redundant internal encodings,” IEEE Trans. Comput. vol. 62, no. 5, [Online]. Available: *http://doi.ieeecomputersociety.* org/10.1109/TC.2013.160

[14] L. Han and S. Ko, “High speed parallel decimal multiplication with redundant internal encodings,” IEEE Trans. Comput., vol. 62, no. 5, pp. 956–968, May 2013.

[15] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754(TM)-2008 IEEE Comput. Soc., Aug. 2008.

[16] A. Vazquez, E. Antelo, and P. Montuschi, “A new family of high performance parallel decimal multipliers,” in Proc. 18th IEEE Symp. Comput. Arithmetic, Jun. 2007, pp. 195–204.