

Design and Implementation of Fast Radix- 10 Multiplication by using Redundant BCD Codes

M. Venkata Subbarao¹, G. M Anitha Priyadashini²

¹PG SCHOLAR Dept. of ECE Srikakulam, AP, (India)

²Assoc.Professor Dept. of ECE Srikakulam, AP, (India)

ABSTRACT

Multiplication is the basic operation in any signal processing systems and financial applications, all these applications requires multiplication to be performed in a faster and efficient manner on a silicon chip. This paper describes the algorithm and architecture of a BCD parallel multiplier. The design exploits two properties of redundant BCD codes to speed up its computation. Namely, the redundant BCD excess-3 code (XS-3) and the overloaded BCD representation (ODDS). In addition to this, number of new techniques are used in order to reduce significantly latency, area and for implementation on FPGA compared to existing implementations. Also the redundancy in XS-3 code is utilized for generating multiplicand multiples in a simple, faster and a carry-free way. Implemented design has three stages. Partial product generation, reduction and final conversion to BCD. For to implement the design in hardware the partial product reduction architecture is modified here to use a bank of ripple carry adder trees.

Keywords:Radix 10, BCD codes, Multipliers, VHDL

I. INTRODUCTION

Multiplication of decimal numbers plays a vital role in many user-oriented applications like finance and commercial and where rounding and conversion of numbers those inherent to binary representations in floating-point cannot be tolerated⁴. This is the cause for decimal operations to become more popular in the recent years. The existence of various microprocessors like Fujitsu Sparc X and IBM power microprocessor^{6,9} families that are oriented to mainframes and servers include fully IEEE 754-2008 Decimal Floating Point Units (DFPUs) for 16-digit decimal and 32-digit decimal formats. Also, the standard IEEE 754-2008 for Floating-Point Arithmetic which includes specifications and format for decimal multiplication have created a path for decimal hardware^{7,10,11} in a significant manner. Again, division and multiplications are performed by using digit by digit algorithm in iterative manner because of which it adds to low performance. The use of aggressive cycle time in Processors will employ additional constraints on parallel techniques in order to reduce the latency employed by parallel design⁷. Thus, an efficient algorithm is required to develop a DFPU to perform multiplication and to have a most regular VLSI layouts which allows pipelining effectively. The hardware implementation for BCD is more widely used as fixed point values than the binary num-bers for easy conversion between user representation and the machine. However the use of decimal numbers has its own disadvantages. Decimal number is represented using four bit binary number ranging from [0–9], since the digits from 10 to 15

are left out in BCD it raises the complications than the binary number system and adds to the delay and penalties in terms of area required for arithmetic units.

A variety of designs had been proposed for designing a BCD multiplier to improve its performance of multiplication operation. The carry save format 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 adders or a tree configuration of adders. The use of decimal signed-digit representation relays on digit set of redundant values $\{-x, \dots, 0, \dots, x\}$, $5 \leq x \leq 9$, to allow carry free addition. The available radix-10 with signed-digit and carry save arithmetics 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.

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 $\{-5, -4, \dots, 0, \dots, 4, 5\}$. 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).

II. 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:

$$Z = -S_z \times 10^n + \sum_{x=0}^{n-1} Z_x \times 10^x$$

Where n being number of digits, Z_i ; $[m - e, 1 - e]$ is the i th digit with range from $0 \leq m \leq e$, and $9 + e \leq m \leq 24 - 1$ ($=15$) and S_z 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 $[0, 11]$ 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,

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

The implementation of the term $(9 - Z_i)$ 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.

III. DESIGN METHODOLOGY

The design methodology for a $p \times q$ -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.

A. 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 Y_k of a BCD number is recoded in SD radix-10 recoder to generate a 5-bit one-hot code represented by $Y_{bk} = \{Y_{1k}, Y_{2k}, Y_{3k}, Y_{4k}, Y_{5k}\}$. The obtained Y_{bk} is used as a select lines for a 5:1 multiplexer to select a respective multiplicand multiples $\{1X, 2X, 3X, 4X, 5X\}$ Y_{bk} also include 1-bit value called signed bit, which is used to control the negation of a selected number. select lines for a 5:1 multiplexer to select a respective multiplicand multiples $\{1X, 2X, 3X, 4X, 5X\}$ Y_{bk} also include 1-bit value called signed bit, which is used to control the negation of a selected number.

B. Decimal Partial Product Reduction

Existing PPR Tree

Decimal PPR tree is used here to reduce the number of partial products generated. The obtained partial products in the PPG step are maintained in the form an array as shown in Figure 4. Where S_k is the sign-bit encoding, O is the ODDS digit in the range $[0, 15]$, H_k is the ten's complement encoding given by: $Y_{Sk} + \{0, 3, 7\}$ and F is the ODDS digit in range $[0, 15]$. The high level architecture of the PPR consists of three parts: (1) A regular binary Carry save adder tree to compute an estimation of decimal partial product sum in a binary carry save form (S, C) , (2) Sum correction block to count the carries generated between digit columns, and (3) A decimal digit 3:2 compressor which increments the carry-save sum according to the carries count to obtain the final double-word product (A, B) where A being represented with excess-6 BCD digits and B being represented with BCD digits. The PPR tree can be viewed as adjacent columns of h ODDS digits each, h being the column height and $h \leq d+1$. Figure 1 shows the high level architecture for existing PPR unit.

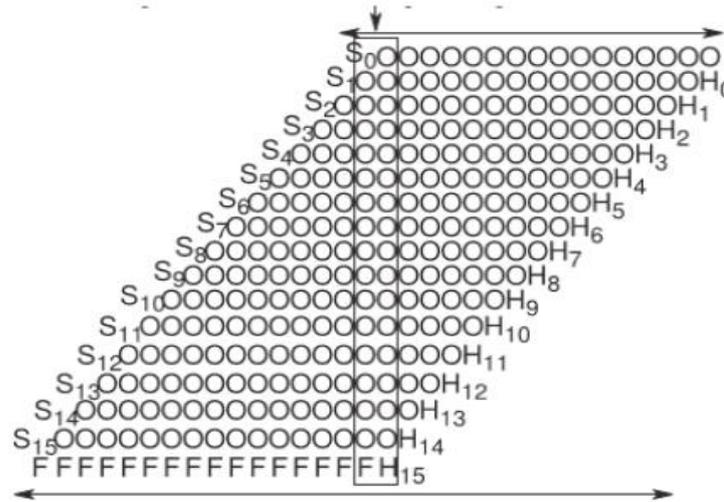


Fig 1. partial Product Array

Proposed PPR Tree with Ripple Carry Adder

In the proposed PPR tree, the partial product reduction architecture is designed using a ripple carry adder as shown in Figure 6. The partial products generated are reduced using a tree of ripple carry adders after aligning. The number of levels used in the tree is $\log_2(2P)$ ranging from $4p + 4$ -bit to $6p$ -bit. The resultant of the two operand is passed to the level down adder and the ripples through the adder length. Since the carry-ripple chain overlap, signal bits are propagated as much as $8p$ bits in addition with $\log_2(2p)$ levels. Mathematical expression of critical path for such a design is given by

$$t_{pr} = t_{c-path} \times 2p + t_{s-path} \times (\log_2(2p) - 1)$$

Where t_{c-path} is delay for carry chain of 4bits (1digit) and t_{s-path} is the delay for sum path of 4bits (1digit) BCD adder. Since the critical path delay only depends on number of levels and path, the required hardware resources is lowered and the delay is low, therefore speed has enhanced.

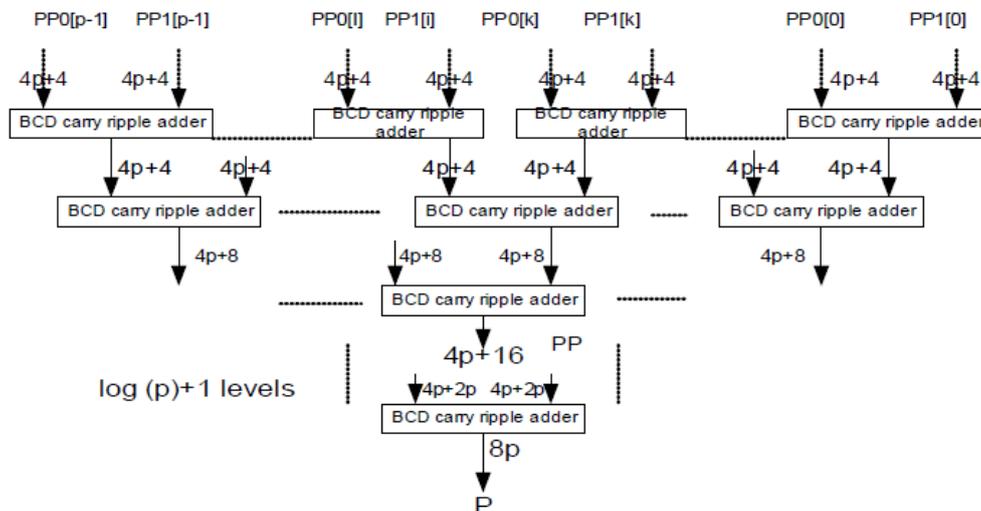


Fig 2. Proposed PPR tree for fast BCD multiplication.

C. Final Conversion to BCD

The selected architecture for conversion is a 2d-digit hybrid parallel prefix or carry-select adder, the BCD quaternary tree adder. 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 $[A_i]+B_i+0$ and $[A_i]+B_i+1$. 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 it is required to analyze the signal arrival profile from the PPR tree, to optimize the area for the minimum delay adder.

V. RESULTS

Multiplication unit is projected and applied in verilogHDL. Its model output is shown in figure. Consider X_i and Y_i are two decimal numbers, where $I \in (0,15)$ and P_i is the final output sum.

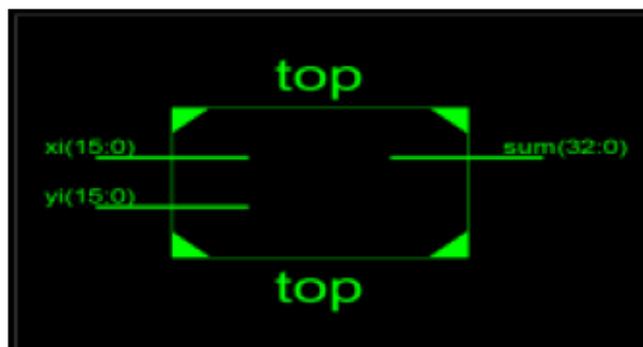


Fig 3 . RTL diagram

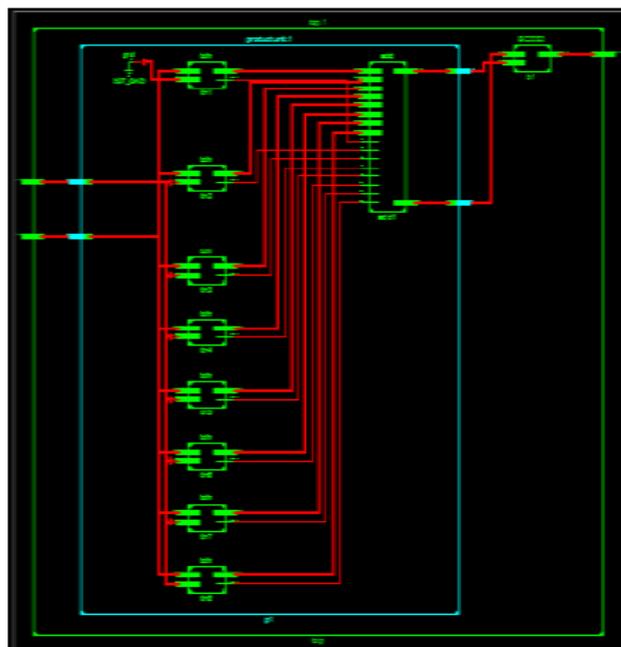


Fig 4. RTL Schematic

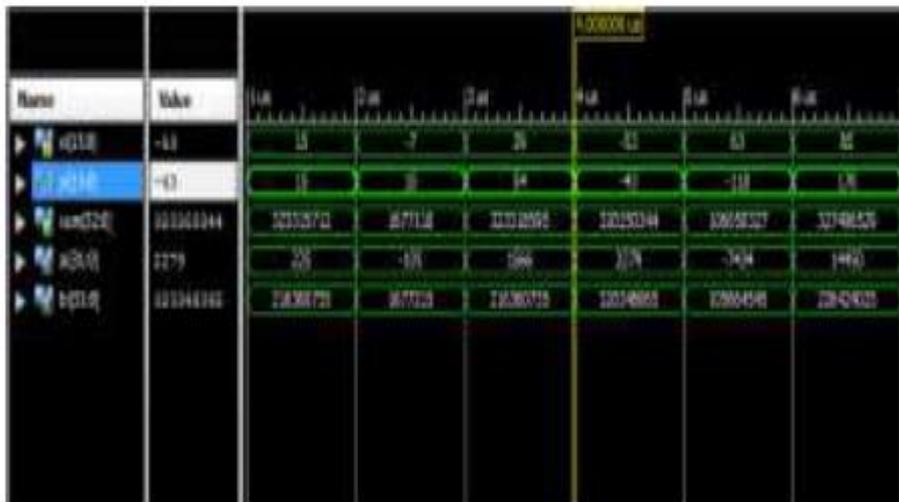


Fig 5 . Output Wave form

VI. CONCLUSION

The combinational parallel BCD multiplier is presented in this paper with various architectural modifications. Parallel multiplication yields benefits of implementing the design in FPGA along with optimization of area and delay. The improvement of the multiplier purely depends on the use of recoding techniques like excess-3, ODDS representation and redundant codes. Which helps in quick generation of partial products, carry-free computation of precomputed terms, generating multiples in easier way just by bit inversion of positive ones.

REFERENCES

- [1.] Vazquez A, Antelo E, Bruguera JD. Fast Radix-10 multiplication using redundant BCD codes. IEEE Transactions on Computers. 2014 Aug; 63(8):1902–14.
- [2.] Aswal, Perumal MG, Prasanna GNS. On basic financial decimal operations on binary machines.
- [3.] IEEE Transactions on Computers. 2012 Aug; 61(8):1084–96.
- [4.] Cowlshaw MF, Schwarz EM, Smith RM, Webb CF. A decimal floating-point specification. Proc 15th IEEE Symp Computer Arithmetic; 2001 Jun. p. 147–54.
- [5.] Cowlshaw MF. Decimal floating-point: Algorithm for computers. Proc 16th IEEE Symp Comput Arithmetic; 2003 Jul. p. 104–11.
- [6.] Carlough S, Schwarz E. Power6 decimal divide. Proc 18th IEEE Symp Appl-Specific Syst, Arch, Process; 2007 Jul. p. 128–33.
- [7.] Carlough S, Mueller S, Collura A, Kroener M. The IBM zEnterprise-196 decimal floating point accelerator. Proc 20th IEEE Symp Comput Arithmetic; Tubingen 2011 Jul 25-27. p. 139–46.
- [8.] Dadda L. Multioperand parallel decimal adder: A mixed binary and BCD approach. IEEE Trans Comput. 2007 Oct; 56(10):1320–8.
- [9.] Dadda L, Nannarelli A. A variant of a Radix-10 combinational multiplier. Proc IEEE Int Symp Circuits Syst; 2008 May. p. 3370–3.

- [10.] Eisen L, Ward JW, Tast H-W, Mading N, Leenstra J, Mueller SM, Jacobi C, Preiss J, Schwarz EM, Carlough SR. IBM POWER6 accelerators: VMX and DFU. IBM J Res Dev. 2007 Nov; 51(6):p. 663–84.
- [11.] Erle MA, Schulte MJ. Decimal multiplication via carry save addition. Proc IEEE Int Conf Appl-Specific Syst, Arch, Process; 2003 Jun 24-26. p. 348–58.
- [12.] Erle MA, Schwarz EM, Schulte MJ. Decimal multiplication with efficient partial product generation. Proc 17th IEEE Symp Comput Arithmetic; 2005 Jun 27-29. p. 21–8.
- [13.] Faraday Tech. Corp. 90nm UMC L90 standard performance low-K library (RVT) [Online]. 2004..
- [14.] GorginS, Jaberipur G. A fully redundant decimal adder and its application in parallel decimal multipliers. Microelectron J. 2009 Oct; 40(10):1471–81.
- [15.] Han L, Ko S. High speed parallel decimal multiplication with redundant internal encodings. IEEE Trans. Comput. 2013 May; 62(5):956–68.
- [16.] Kenney RD, Schulte MJ, Erle MA. High-frequency decimal multiplier. Proc IEEE Int Conf Comput Des VLSI Comput Process; 2004 Oct 11-13. p. 26–9.
- [17.] Kenney RD, Schulte MJ. High-speed multioperand decimal adders. IEEE Trans Comput. 2005 Aug; 54(8):953–63.



M. Venkata Subbarao pursuing his M.Tech in the department of Electronics and Communication Engineering (VLSI), Sri Venkateswara College of Engineering & Technology, Etcherla, Srikakulam, A.P., India. Affiliated to Jawaharlal Nehru Technological University, Kakinada. Approved by AICTE, NEW DELHI. He obtained his B.Tech(ECE) from RVR&JC College of Engineering, Guntur.



Mrs.G.M.Anitha Priyadarshini received her B.Tech degree in Electronics & Communication Engineering from Sri Venkateshwara University in 1997, She then received her M.Tech degree in Digital Systems & Computer Electronics from JNTUH.And Pursuing her Ph.D from Rayalaseema University ,Kurnool.She Entered into Teaching field in 2006 as Lecturer and latter promoted as Assistant professor, Associate Professor.Presently,she is working as Assoc.Professor in Department of ECE,Sri Venkateshwara College of Engineering and Technology (A.P,India).she has Published more than 10 papers in National /International Journal/Conference.