# International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 4, No. 4
November 2015



www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 4, No. 4, November 2015


*Research Paper*

# IMPLEMENTATION OF FAST RADIX-10 BCD MULTIPLIER FOR SPEED OPTIMIZED APPLICATIONS

Shaik Naziyan[1]* and N Suresh Babu[2]

*Corresponding Author: Shaik Naziyan ✉ naziyan123@gmail.com

The decimal multiplication is one of the most important decimal arithmetic operations which have a growing demand in the area of commercial, financial, and scientific computing. It has been revived in recent years due to the large amount of data in commercial applications. In this paper, we propose a parallel decimal multiplication algorithm with three components, which are a partial product generation, a partial product reduction, and a final digit-set conversion. First, a redundant number system is applied to recode not only the multiplier, but also multiples of the multiplicand in Signed-Digit (SD) numbers. Furthermore, we present a multi operand SD addition algorithm to reduce the partial product array. We consider the problem of multi operand parallel decimal addition with an approach that uses binary arithmetic, suggested by the adoption of Binary-Coded Decimal (BCD) numbers. This involves corrections in order to obtain the BCD result or a Binary-to-Decimal (BD) conversion. The BD conversion moreover allows an easy alignment of the sums of adjacent columns. We treat the design of BCD digit adders using fast carry-free adders and the conversion problem through a known parallel scheme using elementary conversion cells. Spread sheets have been developed for adding several BCD digits and for simulating the BD conversion as a design tool. In this project Xilinx-ISE tool is used for simulation, logical verification, and further synthesizing.

Keywords: BCD, Signed-digit, Xilinx-ISE, Partial product reduction

## INTRODUCTION

Hardware implementations of decimal arithmetic units have recently gained importance because they provide higher accuracy in financial applications. In this work, we present a combinational decimal multiplier which can be pipelined to reach the desired throughput. The multiply unit is organized as follows: the multiplier is recoded; the partial products are kept in a redundant format; the partial product are accumulated by a tree of redundant adders and the final product is obtained by converting the

[1] M.Tech Student, Department of E.C.E., Chirala Engineering College, Ramapuram Beach Rd, Chirala, Andhra Pradesh 523157, India.

[2] Associate Professor, Department of E.C.E., Chirala Engineering College, Ramapuram Beach Rd, Chirala, Andhra Pradesh 523157, India.

carry-save tree's outputs into Binary-Coded Decimal (BCD) format. With respect to previous implementations of radix-10 multipliers such as the ones in our design is different in the following aspects.The multiplier is combinational (parallel) and not sequential.we recode only the multiplier while in both operands are recoded in –5 to +5.In the partial product generation only multiples of 5 and 2 are required. The accumulation of partial products is done in a tree of radix-10 carry-save adders and counters while in the sequential unit of signed-digit adders are used. We present the standard cells implementation of the multiplier and compare its latency with those of the schemes presented in and Moreover, we compare the delay and the area of the decimal combinational multiplier with those obtained by the implementation of a binary (radix-4) doubleprecision multiplier.For the multiplication p = x · y, we assume that both the multiplicand x and the multiplier y are sign-and-magnitude n-digit fractional numbers in BCD format normalized in [0.1, 1.0). The multiplication shift-and-add algorithm is based on the identity

$$p = x \cdot y = \sum_{i=0}^{n-1} x y_i r^i$$

where for decimal operands r = 10, yi [0, 9] and x is an-digit BCD vector. We consider in the following n = 16. To avoid complicated multiples of x, we recode yi = yHi + yLi with yH {0, 5, 10} and yL {–2, 1, 0, 1, 2} as indicated. With this recoding, we need to precompute only the multiples 5x and 2x, while 10x is obtained by left-shifting x one digit.The negative values "x and "2x are represented in radix-10 radix-complement. Each partial product xyi is positive and sign extension is not necessary. The partial products are accumulated by using an adder tree, and the multiplication is completed by a carry-save to BCD conversion.

In this paper a new technique is presented under which it is briefly shown in the above figures in the form of the block diagram and followed by the architecture and is explained in an elaborative fashion respectively. Here in the implementation of the architecture in relation to that SD multiplier of the radix 10 of the well oriented d digit plays a crucial role in its analysis based perspective respectively. There are some of the sequential steps are used for the implementation of the system and which includes decimal codes of the partial product generation and followed by the SD encoding of the radix 10 strategy and partial product reduction in addition with carry propagate BCD respectively.

Here the partial products of the d+1 data bits are generated followed by the help of the multiplier based encoding basis on behalf of the SD digits of the radix 10 strategy with the proper addition of the leading bit respectively. 5:1 mux level can be controlled by the help of the SD digit radix 10 strategy in which it includes the design based specification of the positive coding of the multiple multiplicand in a well oriented fashion respectively. For getting the individual partial product in which it includes the gates of XOR based level under which the output bits are getting inverted by the muxes of the 5:1 until the negativity is attained under the SD digit of the radix 10respectively. Before the codes of the partial products gets reduced the alignments of thecodes plays a crucial role and is done in an effective manner on behalf of the weight based decimal point respectively.

Digital recorder implementation For the purpose of the computation oriented multiples of

the multiplicand under which the recoders of the digit are used for the operation reduction of the computation of the partial product is a major concern respectively. Here the BCD based recoder implementation on behalf of the logical digits there is a straight forwards strategy excess 6 scenario where there is an inclusive of the mapping of the certainty of the decimal digits is a major concern respectively. As per the analysis of the various sub sets there is a well efficient presentation of the desired properties under which by the inclusive of the codes of the non redundant decimal is a major concern and plays a crucial role in a well efficient manner respectively.

In this work, we focus on the improvement of parallel decimal multiplication by exploiting the redundancy of two decimal representations: the ODDS and the redundant BCD excess-3 (XS-3) representation, a self-complementing code with the digit set. We use a minimally redundant digit set for the recoding of the BCD multiplier digits, the signed-digit radix-10 recoding ,that is, the recoded signed digits are in the set f 0; 1; 2; 3; 4; 5g. For this digit set, the main issue is to perform the 3 multiple without long carry-propagation (note that 2 and 5 are easy multiples for decimal and that 4 is generated as two consecutive 2 operations). We propose the use of a general redundant BCD arithmetic (that includes the ODDS, XS-3 and BCD representations) to accelerate parallel BCD multiplication in two ways:

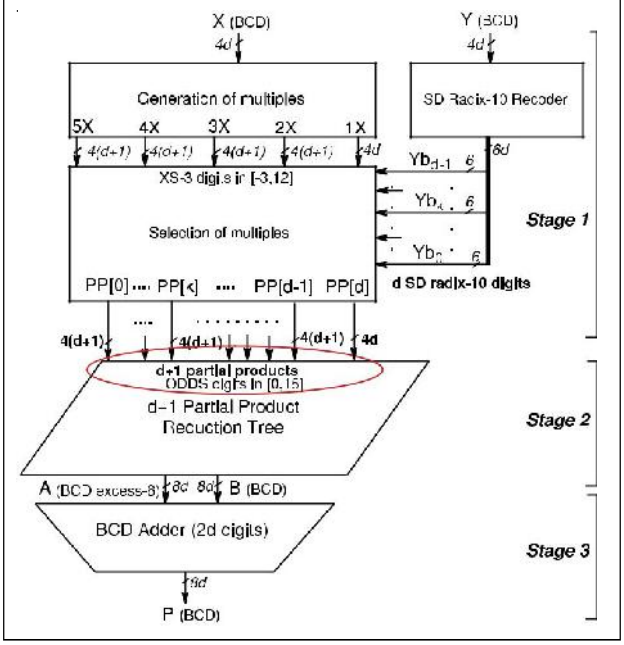
**Partial Product Generation (PPG):** By generating posi-tive multiplicand multiples coded in XS-3 in a carry-free form. An advantage of the XS-3 representation over non-redundant decimal codes (BCD and 4221/5211) is that all the interesting multiples for deci-mal partial product generation, including the 3X multiple, can be implemented in constant time with an equivalent delay of about three XOR gate levels. Moreover, since XS-3 is a self-complementing code, the 9's complement of a positive multiple can be obtained by just inverting its bits as in binary.

**Partial Product Reduction (PPR):** By performing the reduction of partial products coded in ODDS via binary carry-save arithmetic. Partial products can be recoded from the XS-3 representation to the ODDS representation by just adding a constant factor into the partial product reduction tree. The resultant partial product reduction tree is implemented using regular structures of binary carry-save adders or compressors. The 4-bit binary encoding of ODDS operands allows a more efficient mapping of decimal algorithms into binary techniques. By contrast, signed-digit radix-10 and BCD carry-save redundant representations require specific radix-10 digit adders.

## REDUNDANT BCD REPRESENTATIONS

The proposed decimal multiplier uses internally a redun-dant BCD arithmetic to speed up and simplify the imple-mentation. This arithmetic deals with radix-10 ten's complement integers of the form.The value of $Z_i$ depends on the decimal representation parameterized by (l; m; e). We use a 4-bit encoding to repre-sent digits $Z_i$. This allows us to manage decimal operands in different representations with the same arithmetic, such as on the other hand, the binary value of the 4-bit vector representation of $Z_i$ is given by them

Figure 1: Combinational SD Radix-10 Architecture




In contrast to our previous SD radix-10 implementations , 3X is obtained in a reduced constant time delay (3 XOR-gate delays) by using the XS-3 representation. Moreover, a negative multiple is generated from the corre-spondent positive one by a bitwise XOR operation. Consequently, the latency is reduced and the hardware implementation is simplified. The scheme proposed in also produces 3X in constant time but using redundant signed-digit BCD arithmetic.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). Our proposal relies on a binary carry-save adder tree to perform carry-free additions of the deci-mal partial products. The array of d þ 1 ODDS partial prod-ucts can be viewed as adjacent digit columns of height h d þ 1. Since ODDS digits are encoded in binary, the rules for binary arithmetic apply within the digit bounds, and only carries generated between radix-10 digits (4-bit columns) contribute to the decimal correction of the binary sum. That is, if a carry out is produced as a result of a 4-bit (modulo 16) binary addition, the binary sum must be incremented by 6 at the appropriate position to obtain the correct decimal sum (modulo 10 addition).

Two previous designs implement tree structures for the addition of ODDS operands. In the non specula-tive BCD adder, a combinational logic block is used to determine the sum correction after all the operands have been added in a binary CSA tree, with the maximum num-ber of inputs limited to 19 BCD operands. By contrast, in our method the sum correction is evaluated concurrently with the binary carry-save additions using columns of binary counters. Basically we count the number of carries per decimal column and then a multiplication by 6 is per-formed (a correction by 6 for each carry-out from each column). The result is added as a correction term to the output of the binary carry-save reduction tree. This improves significantly the latency of the partial product reduction tree. Moreover, the proposed architecture accepts an arbitrary number of ODDS or BCD operand inputs. Some of PPR tree structures presented in (the area-improved PPR tree) also exploit a similar idea, but rely on a custom designed ODDS adder to perform some of the stage reductions. Our proposal aims to provide an optimal reuse of any binary CSA tree for multioperand decimal addition, as it was done in for the 4221 and 5211 decimal codings

**Conversion to (Non-Redundant) BCD:** We consider the use of a BCD carry-propagate adder to perform the final conversion to a non-redundant BCD product P ¼ Aþ B. The proposed architecture is a 2d-digit hybrid parallel prefix/ carry-select adder, the BCD Quaternary Tree

adder (see Section 6). The sum of input digits $A_i$, $B_i$ at each position i has to be in the range ½0; 18 & so that at most one decimal carry is propagated to the next position i þ 1 . Furthermore, to generate the correct decimal carry, the BCD addition algorithm implemented requires $A_i$ þ $B_i$ to be obtained in excess-6. Several choices are possible. We opt for representing operand A in BCD excess-6 ($A_i$ 2 ½0; 9 &, ½$A_i$ & ¼ $A_i$ þ e, e ¼ 6), and B coded in BCD ($B_i$ 2 ½0; 9 &, e ¼ 0).

The overloaded BCD (or ODDS—overloaded decimal digit set) representation was proposed to improve decimal multioperand addition [18], and sequential [17] and parallel [12], [13] decimal multiplications. In this code, each 4-bit binary value represents a redundant radix-10 digit $X_i$ 2 ½0; 15&. The ODDS presents interesting properties for a fast and efficient hardware implementation of decimal arithme-tic: (1) it is a redundant decimal representation so that it allows carry-free generation of both simple and complex decimal multiples (2X, 3X, 4X, 5X, 6X, ...) and addition, (2) since digits are represented in the binary number system, digit operations can be performed with binary arithmetic, and (3) unlike BCD, there is no need to implement addi-tional hardware to correct invalid 4-bit combinations. A disadvantage with respect to signed-digit and self-comple-menting codes, is a slightly more complex implementation of 9's complement operation for negation of operands and subtraction.

## DECIMAL PARTIAL PRODUCT REDUCTION

The PPR tree consists of three parts: (1) a regular binary CSA tree to compute an estimation of the decimal partial product sum in a binary carry-save form (S, C), (2) a sum correction block to count the carries generated between the 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), 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 (Figure 4), and h d þ 1.

Figure 5 shows the high-level architecture of a column of the PPR tree (the ith column) with h ODDS digits in [0, 15] (4 bits per digit). Each digit column of the binary CSA tree (the gray colored box in Figure 5) reduces the h input digits and $n_{cin}$ input carry bits, transferred from the previous corresponding dot-diagram representation for h ¼ 17 (m ¼ 14) in Figure 6b. An efficient implementation is obtained by representing the digit of $W_i$ 6 with l ODDS digits, $Wt_i$½0&; ...; $Wt_i$½l 1&), being l ¼ 1 for Decimal64, and l ¼ 2 for Decimal128.

Figure 2: Decimal Partial Product Array Generated for d ¼ 16 (16 16-Digit Multiplier)

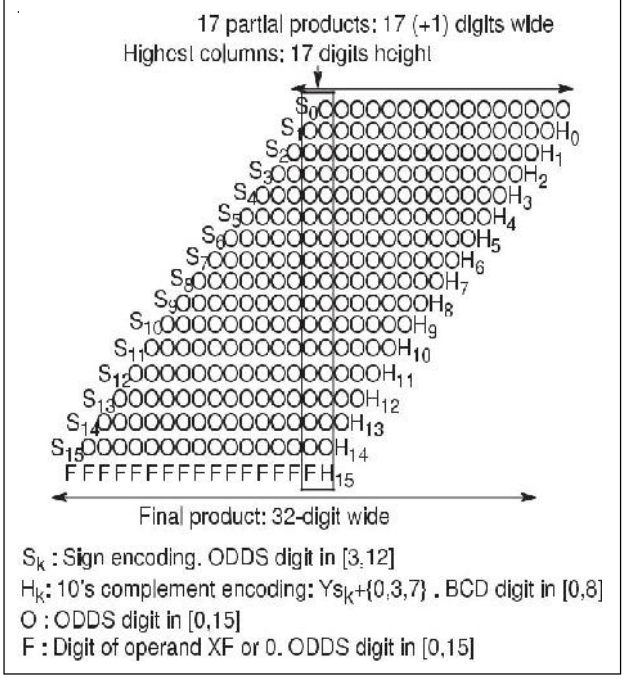
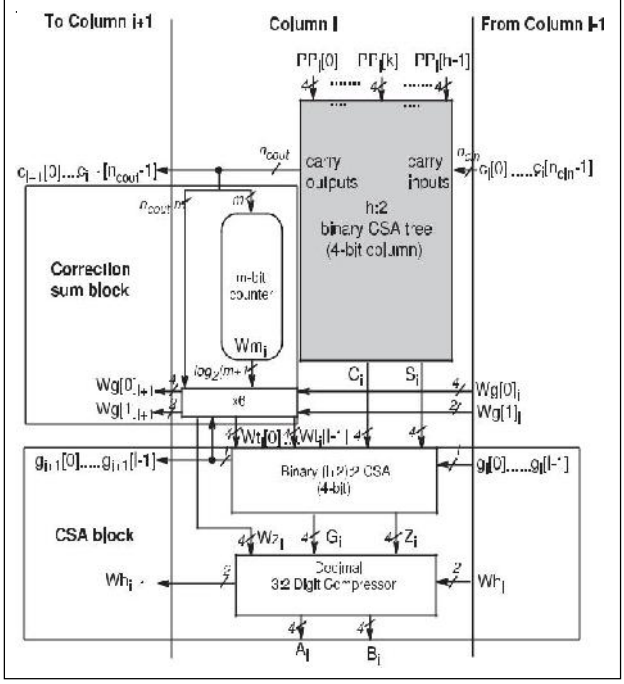


Figure 3: High-Level Architecture of the Proposed Decimal PPR Tree (h Inputs, 1-Digit Column)




## FINAL CONVERSION TO BCD

The selected architecture is a 2d-digit hybrid parallel prefix/ 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 imple-ments ½$A_i$& þ $B_i$ þ 0 and ½$A_i$& þ $B_i$ þ 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.

## RESULTS

### RTL Schematic

The RTL SCHEMATIC gives the information about the user view of the design. The internal blocks contains the basic gate representation of the logic.

Figure 4: RTL Schematic

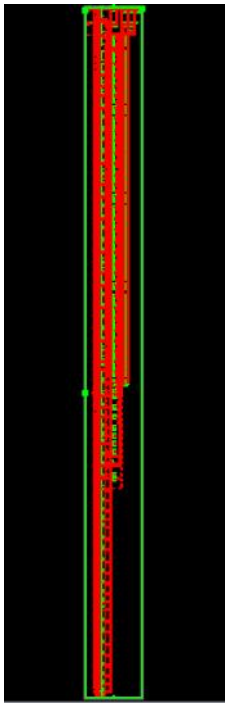


Figure 5: Technology Schematics

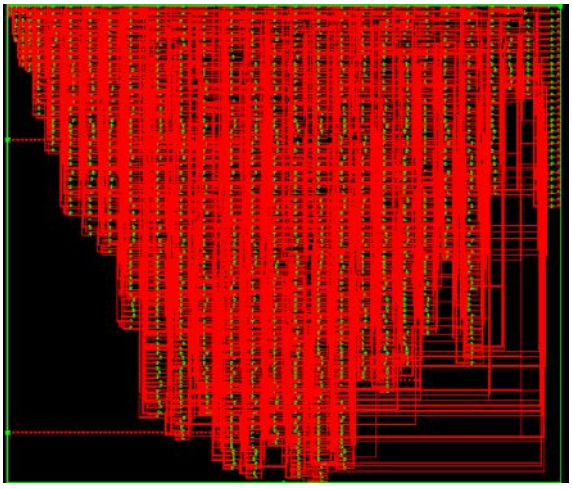
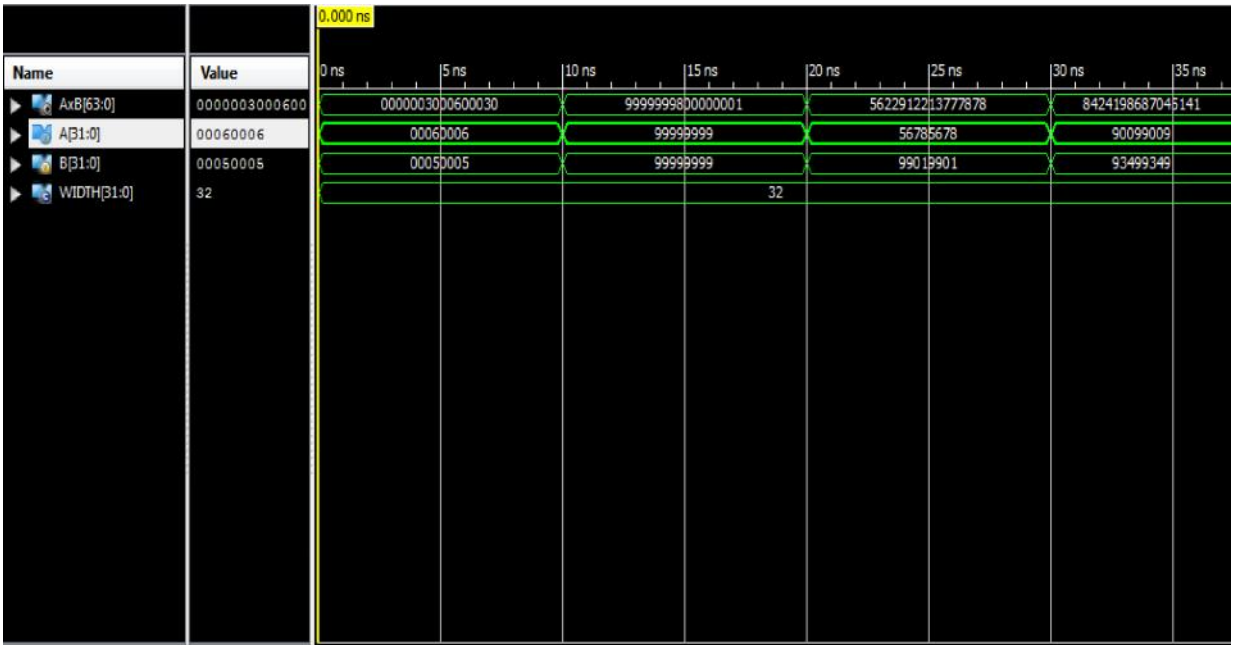


Figure 6: Waveform




These basic gate realization is purely depend upon the corresponding FPGA selection and the internal database information.

### Technology Schematic

The Technology Schematic gives the information about the chip view of the design. This mainly consists of LUTs, input buffers, output buffers, D-Flipflop components. Internally Look Up Tables (LUTs) contains the corresponding logic boolean equations, its schematic representation, k-map representation and its truth table representation.

### Waveform

In the waveform which is shown above, *A* and *B* signals represents the inputs which we are applying to the design. Similarly *A* x *B* is the output signal for the design. The constant WIDTH signal here is 32. To obtain the required outputs force the inputs logic with the required values. All the signals which are shown in the above waveform are in DECIMAL mode. Here in the waveform the multiplication operation is performed between the inputs *A* and *B* and the corresponding result is stored in the signal *A* x *B*.

## CONCLUSION

In this paper we have presented the algorithm and architecture of a new BCD parallel multiplier. The improvements of the proposed architecture rely on the use of certain redundant BCD codes, the XS-3 and ODDS representations. Partial products can be generated very fast in the XS-3 representation using the SD radix-10 PPG scheme: positive multiplicand multiples (0X, 1X, 2X, 3X, 4X, 5X) are pre computed in a carry-free way, while negative multiples are obtained by bit inversion of the positive ones. On the other hand, recoding of XS-3 partial products to the ODDS representation is straightforward. The ODDS representation uses the redundant digit-set [0, 15] and a 4-bit binary encoding (BCD encoding), which allows the use of a binary carry-save adder

tree to perform partial product reduction in a very efficient way. The HDL is developed based on the verilog language. The RTL is simulated and synthesized in the XILINX ISE.

## REFERENCES


1. Aswal A, Perumal M G and Prasanna G N S (2012), "On Basic Financial Decimal Operations on Binary Machines", *IEEE Trans. Comput.*, Vol. 61, No. 8, pp. 1084-1096.
2. Carlough S and Schwarz E (2007), "Power6 Decimal Divide", in Proc. 18th IEEE Symp. Appl.-Specific Syst., Arch., Process., July, pp. 128-133.
3. Carlough S, Mueller S, Collura A and Kroener M (2011), "The IBM zEnterprise-196 Decimal Floating Point Accelerator", in Proc. 20th IEEE Symp. Comput. Arithmetic, July, pp. 139-146.
4. Cowlishaw M F (2003), "Decimal Floating-Point: Algorism for Computers", in Proc. 16th IEEE Symp. Comput. Arithmetic, July, pp. 104-111.
5. Cowlishaw M F, Schwarz E M, Smith R M and Webb C F (2001), "A Decimal Floating-Point Specification", in Proc. 15th IEEE Symp. Comput. Arithmetic, June, pp. 147-154.
6. Dadda L (2007), "Multioperand Parallel Decimal Adder: A Mixed Binary and BCD Approach", *IEEE Trans. Comput.*, Vol. 56, No. 10, pp. 1320-1328.
7. Dadda L and Nannarelli A (2008), "A Variant of a Radix-10 Combinational Multiplier", in *Proc. IEEE Int. Symp. Circuits Syst.*, May, pp. 3370-3373.
8. Eisen L, Ward J W, Tast H-W, Mading N, Leenstra J, Mueller S M, Jacobi C, Preiss J, Schwarz E M and Carlough S R (2007), "IBM POWER6 Accelerators: VMX and DFU", *IBM J. Res. Dev.*, Vol. 51, No. 6, pp. 663-684.
9. Erle M A and Schulte M J (2003), "Decimal Multiplication via Carry-Save Addition", in *Proc. IEEE Int. Conf Appl.-Specific Syst., Arch., Process*., June, pp. 348-358.


International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com