



High Speed Multiplication Using BCD Codes For DSP Applications

Balasundaram¹, Dr. R. Vijayabhasker²

PG Scholar, Dept. of Electronics & Communication Engineering, Anna University Regional Centre, Coimbatore, Tamilnadu, India¹

Assistant Professor, Dept. of Electronics & Communication Engineering, Anna University Regional Centre, Coimbatore, Tamilnadu, India²

Abstract— In digital systems, multiplier is the prominent deciding factor to the overall speed, area and power consumption. The intention of this project is to improve the parallel decimal multiplication. The proposed decimal multiplier uses internally a redundant BCD (Binary Coded Decimal) code. The overloaded BCD or ODDS (Overloaded Decimal Digit Set) representation was proposed to improve the decimal Multi-operand addition, sequential and parallel decimal multiplications. The proposed system goes through three main stages. First, Partial Product Generation (PPG) algorithm uses radix10 recoding that produces a reduced number of partial products. Second, Partial Product Reduction (PPR) algorithm is used to reduce the partial products into two 2d-digit words (A,B). Third, Non- redundant BCD conversion produces final BCD products ($P=A+B$). The parallel decimal multiplier simplifies the implementation and increases the operation speed. The proposed decimal multiplier reduces the overall multiplier area for similar target delays with respect to the fastest implementation. The high speed area efficient decimal multiplication using CSA adder reduces the delay by 12.23% compared to existing system.

Keywords- Carry Save Adder, redundant excess-3 code, Parallel multiplication. Introduction

I. INTRODUCTION

Decimal fixed-point and floating-point formats are important in financial, commercial, and user-oriented computing. Since area and power dissipation are critical design factors in state-of-the-art DFPUs, multiplication and division are performed iteratively by means of digit-by digit algorithms and therefore they present low performance. Moreover, the aggressive cycle time of these processors puts an additional constraint on the use of parallel techniques for reducing the latency of DFP multiplication in high-performance DFPUs.

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 [-3,12]. The general redundant BCD arithmetic is used to (that includes the ODDS, XS-3 and BCD representations to accelerate parallel BCD multiplication in two ways such as Partial Product Generation and Partial Product Reduction.

The design of area efficient high speed with reasonable power consumption 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. The reasons behind the

use of binary data for doing the arithmetic operations in almost all the computer systems is the speed and simplicity of binary arithmetic, efficiency in storing the binary data. But, for the Digital Signal processing and commercial applications, the use of decimal arithmetic is still relevant. But the speed of the operation major concern for the decimal software. Moreover, the commercial databases contain more decimal data than binary data. For the purpose of processing, these decimal data are converted into binary data. And, once the processing is completed, those are again converted back into the decimal format. Cause some delay. Binary Coded Decimal (BCD) on-going research which is carried out in almost everywhere using BCD with reduced area and delay.

The main objectives to improve the performance of BCD multiplication. Other objectives are given below.

- To avoid long carry-propagations in the generation of decimal positive multiplicand multiples.
- To obtain the negative multiples from the corresponding positive ones easily.
- To simplify conversion of the partial products generated in XS-3 to the ODDS representation for efficient partial product reduction.



The rest of the paper is organized as follows. Section II discusses about various literatures regarding the different multiplication and arithmetic. Section III describes the concept of existing Section IV reports the experimental results. Finally, Section V concludes the paper.

II. RELATED WORK

Vazquez A, Antelo et al [2], explain basic implementation of the decimal processor on the FPGA decimal operations can be accelerated by a processor on a FPGA board connected to the computer by a standard bus without an advanced input/output interface. The largest group of this connected-by-bus accelerators is probably constituted by Graphics Processing Units, or GPUs. Field programmable Gate-Arrays (FPGAs) are chips in which the hardware can be programmed by the user at logic gate level to implement any processor. By connecting FPGAs to the CPU via a standard bus, can implement custom made accelerators at low cost. FPGA based accelerators can exploit data parallelism and might suffer from the communication bandwidth.

The main drive for decimal units is the need in financial transaction and accounting for correctly rounded decimals that binary arithmetic cannot guarantee. To overcome this loss of precision, financial applications implement decimal arithmetic operations run 100–1000 times slower than the corresponding binary operations. Alternatively, Decimal Floating-Point (DFP) can be directly implemented in hardware and run of magnitudes faster than the software computation.

III. RADIX-10 PARALLEL DECIMAL MULTIPLIER

A. SD Radix-10 Architecture

The Radix-10 architecture for d-digit BCD decimal fixed-point parallel multiplication is based on the techniques for partial product generation and reduction respectively. The code (4221) and (5211) is used instead of BCD to represent the partial product is the main feature of this architecture. This improves the reduction of decimal partial product with respect to other proposals, in terms of latency and area is expected. The architecture of the d-digit SD radix -10 multiplier consists of the following stages, Generation of decimal partial products coded in (4221), reduction of partial products and a final BCD carry-propagate addition.

B. Partial Product Generation

The generation of the d+1 partial product is performed by an encoding of the multiplier into d SD radix-10 digits. Each SD radix-10 digit controls a level of 5:1 muxes, which selects a positive multiplicand multiple (0, X, 2X, 3X, 4X, 5X) coded in (4221). To obtain each partial product a level of XOR gates inverts the output bits of the

5:1 muxes when the sign of the corresponding SD radix-10 digit is negative. Before being reduced the d+ 1 partial product, coded in (4221), are aligned according to their decimal weights. Each p-digit column of the partial product array is reduced to two decimal digits using one of the decimal digit p: 2 CSA trees. The number of digits to be reduced for each column varies from p=d+1 to p= 2. Thus, the d+ 1 partial product are reduced to two 2d digit operands S and H coded in (4221).

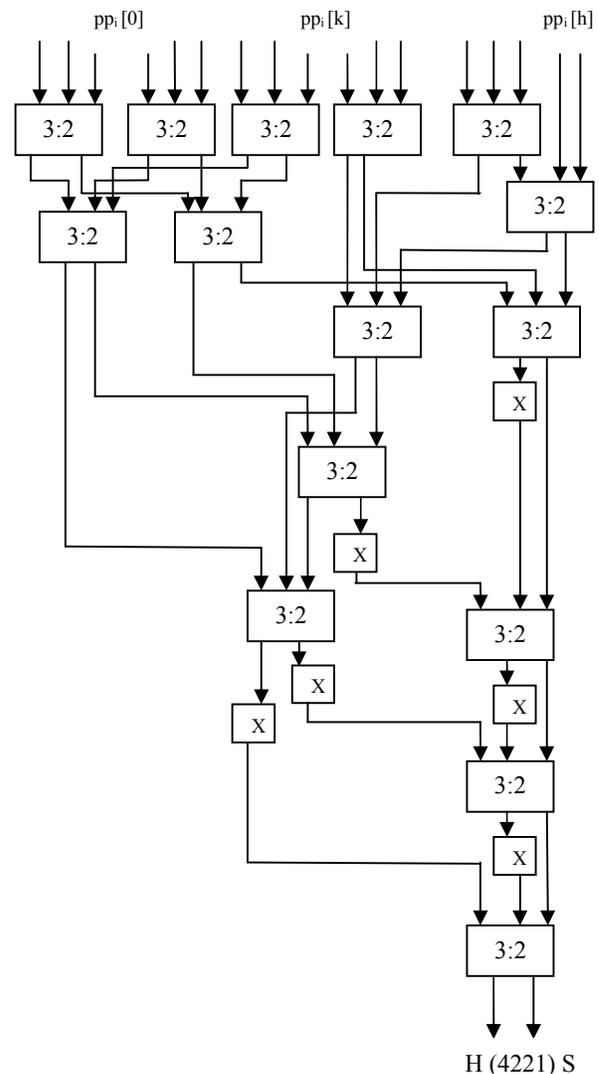


Fig. 1. Binary P:2 CSA Tree

The final product is a 2d-digit BCD word given by $P=2H +S$. Before being added, S and H need to be processed. S is recoded from (4221) to BCD excess-6. The $H \times 2$ multiplication is performed in parallel with the recoding of S. This $\times 2$ blocks uses a (4221) to (5421) digit recoder and a 1-bit wired left shift to obtain the operand 2H coded in BCD shows in Figure 3. For the final BCD carry-propagate addition uses a quaternary tree (Q-T) adder based



on conditional speculative decimal addition. It has low latency and requires less hardware than other alternatives.

C. Partial Product Reduction

The partial product arrays generated by the SD radix-10 encoding each column of p digits is reduced to two digits by means of a decimal digit $p:2$ CSA tree shown in Figure 1. The decimal carries are passed between adjacent digit columns and decimal coding method used for decimal carry-save addition.

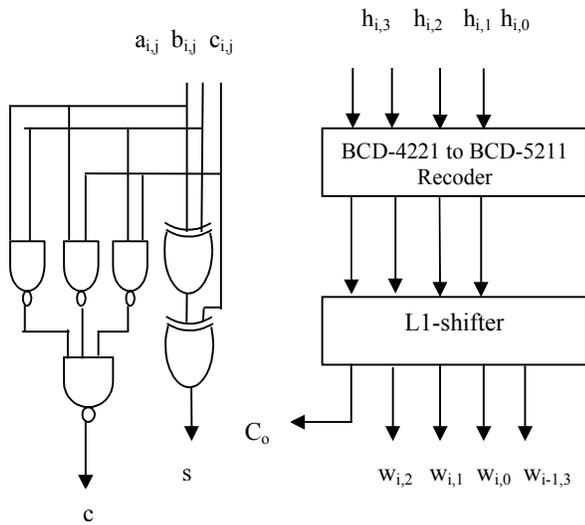


Fig. 2. Scheme of $\times 2$ for BCD-4221

To perform the decimal coding instead of BCD for an efficient implementation of decimal carry-save addition with binary CSAs or full adders use (4221) and (5211). The use of these codes avoids the need for decimal corrections and need to focus on the $\times 2$ decimal multiplication shown in Figure 2. The Decimal $p:2$ CSA Trees for Digits Coded in (4221) Operands Long carry propagation because of that area and delay is more in this system.

IV. IMPLEMENTATION

The algorithm and architecture of a BCD parallel multiplier that exploits some properties of two different redundant BCD codes to speed up its computations are redundant BCD excess-3 code (XS-3) and the overloaded BCD representation (ODDS). Proposed techniques are developed to reduce significantly the latency and area of previous representative high performance implementations.

A. Partial Product Generation

Partial products are generated in parallel using a signed-digit radix-10 recoding of the BCD multiplier with the digit set $[-5,5]$ and a set of positive multiplicand

multiples (0X, 1X, 2X, 3X, 4X, 5X) coded in XS-3 encoding has several advantages.

First, it is a self-complementing code the negative multiplicand multiple can be obtained by just inverting the bits of the corresponding positive one. The available redundancy allows a fast and simple generation of multiplicand multiples in a carry-free way. Finally, the partial products can be recoded to the ODDS representation by just adding a constant factor into the partial product reduction tree.

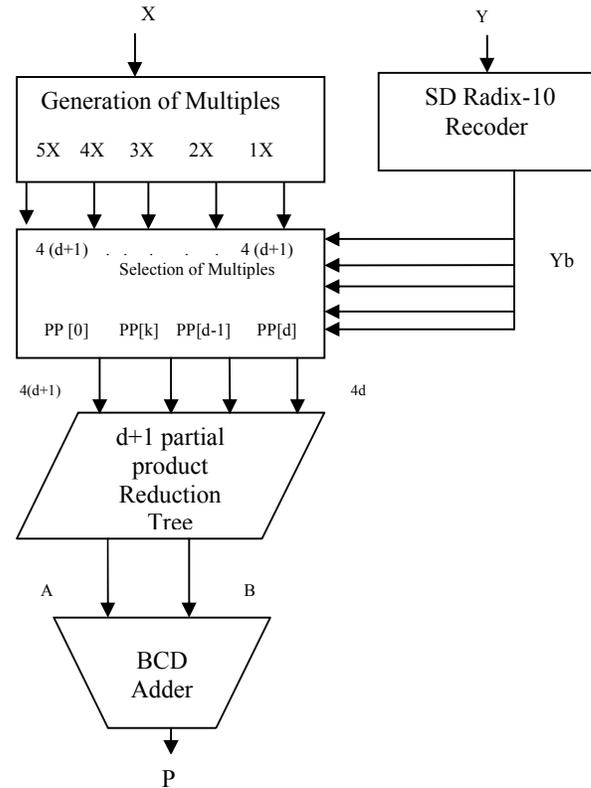


Fig. 3. Combinational SD Radix-10 Architecture

The ODDS uses a similar 4-bit binary encoding as non-redundant BCD techniques explains binary carry-save adders and compressor trees, can be adapted efficiently to perform decimal operations. A variety of redundant decimal formats and arithmetic have been proposed to improve the performance of BCD multiplication.

The BCD carry-save format represents a radix-10 operand using a BCD digit and a carry bit at each decimal multiplication area and power dissipation are critical design factors in DFPU. Multiplication and division are performed iteratively by means of digit-by-digit algorithms for reducing the latency of DFP multiplication in high-performance DFPU.



B. Sign Digit Radix-10 Generation

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. The SD radix-10 encoding produces d SD radix-10 digits Y_b_k $[-5, 5]$, with $k = 0, \dots, d-1$, Y_{d-1} being the MSD (most significant digit) of the multiplier shown in Figure 3.

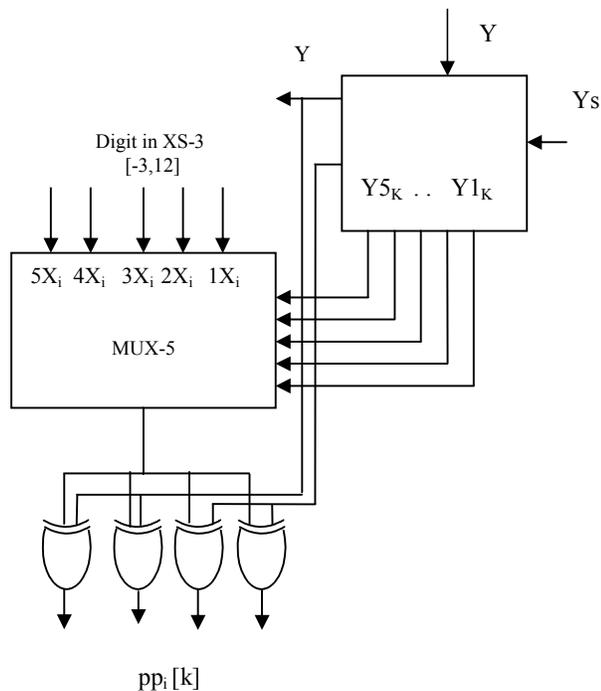


Fig. 4. SD radix-10 Generation of Partial Product Digit

Each digit Y_b_k is represented with a 5-bit hot-one code ($Y_{1k}, Y_{2k}, Y_{3k}, Y_{4k}, Y_{5k}$) to select the appropriate multiple $\{1X, \dots, 5X\}$ with a 5:1 mux and a sign bit Y_{s_k} that controls the negation of the selected multiple shown in figure 4. The negative multiples are obtained by ten's complementing the positive ones. This is equivalent to taking the nine's complement of the positive multiple and then adding 1. As we have shown in Section 2, the nine's complement can be obtained simply by bit inversion. This needs the positive multiplicand multiples to be coded in XS-3, with digits in $[-3,12]$. The d least significant partial products $PP[d-1], \dots, PP[0]$ are generated from digits Y_b_k by using a set of 5:1 muxes. The xor gates at the output of the mux invert the multiplicand multiple, to obtain its 9's complement, if the SD radix-10 digit is negative ($Y_{s_k} = 1$). On the other hand, if the signals ($Y_{1k}, Y_{2k}, Y_{3k}, Y_{4k}, Y_{5k}$) are all zero then $PP[k] = 0$, but it has to be coded in XS-3 bit encoding 0011. The two least significant bits to 1 the input to the XOR gate is $Y_{s_k} * Y_{b_k}$ is zero denotes the Boolean OR operator, where Y_{b_k} is zero equals 1 if all the signals ($Y_{1k}, Y_{2k}, Y_{3k}, Y_{4k}, Y_{5k}$) are zero. In addition, the

partial product signs are encoded into their MSDs. The generation of the most significant partial product $PP[d]$ is described and only depends on $Y_{s_{d-1}}$.

C. Partial Product Reduction

PPR tree consists of three parts a regular binary CSA tree to compute an estimation of the decimal partial product sum in a binary carry-save form (S, C). A sum correction block to count the carries generated between the digit columns and 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 see figure 4.3 and $h_{-d} + 1$.

Finally addition of digits G_i, Z_i, W_{zi} of the column, $G_i + Z_i + W_{zi} \in [0, 45]$. We have designed a decimal 3:2 digit compressor that reduces digits W_{zi}, G_i and Z_i to two digits A_i, B_i . The final BCD product by using a single BCD carry propagate addition $P = A+B$, which is the last step in the multiplication. It required that $A_i + B_i \in [0, 18]$ to reduce the delay of the final BCD carry-propagate adder operand A is obtained in excess-6, so that we compute $[A_i] = A_i + e$ in excess $e = 6$. The output digits sum $[A_i] + B_i \in [6,24]$.

D. Decimal 64 Implementation

The maximum number of carries transferred between adjacent columns of the binary 17:2 CSA tree is 15. These carries are labeled $C_{i+1}[0], \dots, C_{i+1}[14]$ (output carries) and $C_i[0], \dots, C_i[14]$ (input carries). The binary 17:2 CSA tree is built of a first level composed of a 9:2 compressor and a 8:2 compressors, and a second level composed of a 4:2 compressor. To balance the delay of the 17:2 CSA tree and the bit counter, $m = 14$ has been chosen. The 14-bit counter produces the 4-bit digit W_{mi} . The computation of $W_{mi} * 6$ deserves a more detailed description. The 4-bit digit $W_{mi} = W_{i,3}, W_{i,2}, W_{i,1}, W_{i,0}$, with $W_{mi,j}$ being the bits of the digit, is conveniently represented as,

$$W_{mi} = W_{g[0]_{i+1}} \times 2 + W_{mi,0} \quad (1)$$

$$W_{g[0]_{i+1}} = \sum_{j=1}^3 W_{mi,j} \times 2^{j-1} \quad (2)$$

W_{mi} has been split into two parts, the 3 most significant bits of $W_{g[0]_{i+1}}$ and least-significant bit, $W_{mi,0}$. Then, results in

$$W_i = W_{g[0]_{i+1}} \times 2 + W_{mi,0} + C_{i+1}[14] \quad (3)$$



Digit W_{ti} is obtained by the concatenation of the most-significant bit of $Wg[0]i+1 \times 2$ and LSB of $Wg[0]i$, A row of decimal 3:2 digit compressors is used to reduce the 3-operand partial product sum (G, Z, Wz) to two BCD operands (A, B), with A represented in excess-6.

E. Decimal 128 Implementation

The maximum height of the partial product array by the 34 x34-digit BCD multiplier is $h = 35$. The optimal value for parameter m is $m = 31$. Therefore, the addition of these carries has been split into two parts. First, a 31-bit counter evaluates W_{mi} , the 5-bit sum of the 31 fastest carries. Then, the two slowest carries, $C_{i+1}[31]$ and $C_{i+1}[32]$, are added to W_{mi} into a second 5-bit counter. digits $S_i, C_i, Wt[0]i, Wt[1]i$, are reduced to two digits $G_i; Z_i \in [0, 15]$ using a 4-bit binary 4:2 CSA. Finally, the three digits G_i, Z_i, Wz_i are reduced to two excess-6 BCD digits A_i and B_i by using the decimal digit 3:2 compressor shown in figure 3.

It reduces the overall critical path latency, area and improving speed of parallel decimal multiplication to avoid long carry-propagations which Reduces the number of partial products generation.

V. SIMULATION OUTPUTS

A. 16x16 (64) Digit Multiplication

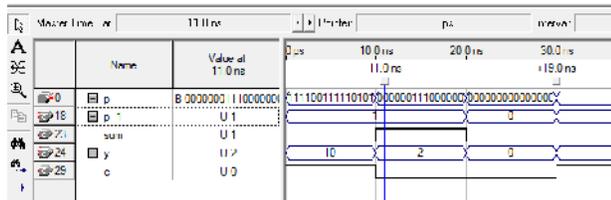


Fig. 5. 16x16 Digit Multiplication

64 decimal multiplication is designed and implemented in Verilog HDL. Its simulation output is shown in Figure 5. Let a_i and b_i are two decimal 4 bit numbers, p is the carry counter output selection inputs add the two decimal number and produce the output sum.

B. 34x34 (128) Digit Multiplication

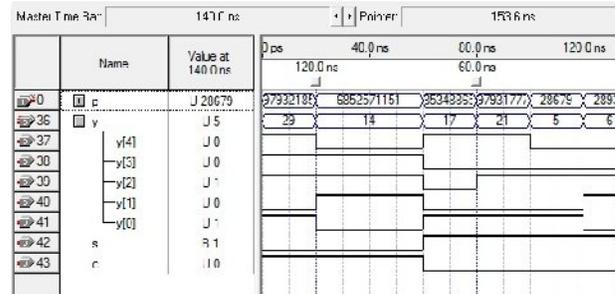


Fig. 6. 34x34 Decimal Multiplication

Decimal-128 multiplication implemented in Verilog HDL. Its simulation output is shown in Figure 6. Let p is a partial product s and c are two decimal numbers, y is the carry counter output selection of inputs add the two decimal number and produce the output.

C. Performance Comparison

TABLE I
DECIMAL 64 DIGIT MULTIPLICATION

PARAMETERS	Existing			Proposed		
	No. of LUT	Power (mw)	Delay (ns)	No. of LUT	Power (mw)	Delay (ns)
PPG	4	56.77	8.353	1	56.28	8.153
PPR	22	324.89	11.812	19	323.39	11.511
16*16	186	326.62	27.181	183	324.62	24.1

The existing parallel decimal multiplication and the high speed area efficient compared in terms of area, delay and power and its performances were tabulated in Table 1. From the obtained results, it is clear that parallel decimal multiplication reduced area, increased delay and power than the existing.

TABLE II
128 DECIMAL MULTIPLICATIONS

PARAMETERS	Existing			Proposed		
	No. of LUT	Power (mw)	Delay (ns)	No. of LUT	Power (mw)	Delay (ns)
PPG	31	325.39	14.511	31	323.39	11.511
PPR	44	326.62	12.864	39	324.41	12.812
34*34	193	358.93	20.283	189	325.93	12.223



The high speed area efficient decimal multiplication using CSA adder reduces the delay by 12.23% compared to existing shown in Table 2. The existing module CSA adder uses each carry multiply by 2 module, it requires additional hardware and increases the power. The proposed method to speedup the operation by reduce the number of partial product.

VI. CONCLUSION

The high speed and area efficient decimal multiplier using CSA. The existing and proposed implemented and their results were compared. From the obtained results, it is clear that the proposed decimal performs in terms of reduced area and delay because of carry save adder. Compared to the conventional method, the proposed method reduces the delay by 12.23%. Implementing this high speed area efficient binary coded decimal in Finite Impulse Response (FIR) filter can be considered, which is extensively used as high speed DSP application and can be implement in FPGA.

REFERENCES

- [1]. Alvaro Vazques, Elisardo Antelo and Javier Bruguera (2014), 'Fast Radix-10 Multiplication Using Redundant BCD Codes', IEEE Vol. 63, No.8, pp.325-338.
- [2]. Carlough S and Schwarz E (2007), 'Power Six Decimal Divide', Proc. 18th IEEE Symp. on Application-Specific Systems, Architectures, and Processors, Vol. 89, No. 8, pp. 28–133.
- [3]. Dadda L (2007), 'Multioperand and Parallel Decimal Adder: A Mixed Binary and BCD Approach', IEEE Transactions on Computers, Vol. 56, No. 10, pp. 1320–1328.
- [4]. Dadda L and Nannarelli A (2008), 'A Variant of a Radix-10 Combinational Multiplier', IEEE Int. Symposium in Circuits and Systems, ISCAS 2008, Vol. 37, No. 2, pp. 3370–3373.
- [5]. Erle M.A, Schwarz E.M and M. J. Schulte (2005), 'Decimal Multiplication With Efficient Partial Product Generation', Proc. 17th IEEE Symposium on Computer Arithmetic, Vol. 73, No. 4, pp. 21–28.
- [6]. Erle M.A and M. J. Schulte (2003), 'Decimal Multiplication Via Carry-Save Addition', Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors, Vol. 51, No.7, pp. 348–358.
- [7]. Gorgin S and Jaberipur G (2013), 'High Speed Parallel Decimal Multiplication with Redundant Internal Encodings', IEEE Transactions on Computers, Vol.45, No 160, pp. 232-249.
- [8]. Han L and Ko S (2013), 'High Speed Parallel Decimal Multiplication with Redundant Internal Encodings', IEEE Transactions on Computers, Vol. 62, No. 5 pp. 956–968.
- [9]. G.Jaberipur, and A. Kaivani (2009), 'Improving the Speed of Parallel Decimal Multiplication', IEEE Transactions on Computers, Vol. 58, No. 11, pp.39–52.
- [10]. Vazquez A,Antelo E and Montuschi P (2010), 'Improved Design of High-Performance Parallel Decimal Multipliers', IEEE Transactions on Computers, Vol. 59, No. 5, pp. 679–693.

BIOGRAPHY



Mr.S.Balasundaram Received BE Degree in (ECE) From Sri Nandhanam College of Engineering and Technology, Tirupattur in 2011. Currently He is Doing ME (VLSI Design) in Anna University Regional Centre, Coimbatore. His General Area of Interest are VLSI Design, VLSI Testing and Verification.



Dr.R.Vijayabhasker completed U.G. in EEE, PG in Power Electronics and Drives, Ph.D in Electrical Engg. Working as Assistant Professor, ECE Dept. Anna University Regional Centre. Fields of Interest are VLSI Design, VLSI Signal Processing, DSP. He has published his works in various Annexure I and II journals.